Annotating and Automatically Tagging Constructions of Causal Language

Jesse Dunietz

CMU-CS-18-100

February 2018

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Jaime Carbonell *(Co-chair)*
Lori Levin *(Co-chair)*
Eduard Hovy
Nianwen Xue, Brandeis University

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the National Eye Institute under grant number R01EY022247 and by the US Army Research Development and Engineering Command Acquisition Center under grant number W911NF12C0020.

Keywords: Natural Language Processing, Semantics, Construction Grammar, Causality, Computational Linguistics, Linguistic Annotation

Abstract

Automatically extracting relationships such as causality from text presents a challenge at the frontiers of natural language processing. This thesis focuses on annotating and automatically tagging causal expressions and their cause and effect arguments.

One popular paradigm for such tasks is SHALLOW SEMANTIC PARSING—marking relations and their arguments in text. Efforts to date have focused on individual propositions expressed by individual words. While this approach has been fruitful, it falters on semantic relationships that can be expressed by more complex linguistic patterns than words. It also struggles when multiple meanings are entangled in the same expression. Causality exhibits both challenges: it can be expressed using a variety of words, multi-word expressions, or even complex patterns spanning multiple clauses. Additionally, causality competes for linguistic space with phenomena such as temporal relations and obligation (e.g., *allow* can indicate causality, permission, or both).

To expand shallow semantic parsing to such challenging relations, this thesis presents approaches based on the linguistic paradigm known as CONSTRUCTION GRAMMAR (CxG). CxG places arbitrarily complex form/function pairings called CONSTRUCTIONS at the heart of both syntax and semantics. Because constructions pair meanings with arbitrary forms, CxG allows predicates to be expressed by any linguistic pattern, no matter how complex.

This thesis advocates for a new "surface construction labeling" (SCL) approach to applying CxG: given a relation of interest, such as causality, we annotate just the words that consistently signal a construction expressing that relation. Then, to automatically tag such constructions and their arguments, we need not wait for automated CxG tools that can analyze all the underlying grammatical constructions. Instead, we can build on top of existing tools, approximating the underlying constructions with patterns of words and conventional linguistic categories.

The contributions of this thesis include a CxG-based **annotation scheme and methodology** for annotating explicit causal relations in English; an **annotated corpus** based on this scheme; and three methods for **automatically tagging** causal constructions. The first two tagging methods use a pipeline architecture based on tentative pattern-matching to combine automatically induced rules with statistical classifiers. The third method is a transition-based deep neural network. The thesis demonstrates the promise of these methods, discusses the tradeoffs of each, and suggests future applications and extensions.

Acknowledgments

Acknowledgments sections are often cheery affairs, suggesting that the final work is the inevitable accumulation of countless small contributions from the author's dozens of kind friends and family. I write this section partly as an expression of gratitude for such contributions. But perhaps at the expense of cheeriness, I feel compelled to express my gratitude from a more reflective angle, even a pensive one.

This thesis nearly didn't happen. Like so many other graduate students, I nearly abandoned the project several times. The fact that I am nonetheless graduating with a Ph.D. is a testament to the quality and quantity of the advice, emotional support, love, and generosity that pushed me through the obstacles I encountered. So to all the people I am about to mention, I say thank you—not just for helping with the substance of this thesis, but for keeping my spirits high, my ideas flowing, my trajectory stable, and my sanity more or less intact.

The highest order of gratitude is reserved for my patient advisors, **Jaime Carbonell** and **Lori Levin**, who have constantly stood ready to offer feedback and guidance to a student who needed a great deal of both. At the same time, they gave me the freedom to find my own way in research, put up with my intellectual wandering, and indulged my unusual side projects. I thank them and the other members of my committee, **Nianwen Xue** and **Eduard Hovy**, for their technical ideas and feedback throughout the thesis process. I especially thank Ed for critiques that fueled many major improvements to Chapters 6, 7, and 8.

I've also been fortunate to work with several other mentors and senior collaborators. **Dan Gillick** conceived and supervised my internship project at Google in the summer of 2013, resulting in a paper that has thus far garnered more citations than anything in this thesis. **Miriam Petruck** at UC Berkeley shared her expertise in annotation, lexical semantics, and frame semantics with Lori and me, helping us connect our efforts with the work of Chuck Fillmore and the FrameNet project. Miriam taught me a great deal about language, and Dan gave me my first on-the-ground training as a language technologies engineer. I am indebted to both.

I would be remiss if I did not thank one more mentor: **Tai-Sing Lee**, my advisor in an earlier academic life (read: my first year of grad school). Although my research ultimately took a different turn, I think often about the scientific and computational lessons I learned in that first year.

Then there are all the esteemed researchers whom I've had the privilege to learn from more informally. Within CMU, their ranks include **Bob Frederking**, **David Mortensen**, **Alan Black**, **Carolyn Rosé**, and **Taylor Berg-Kirkpatrick**. **Chris Dyer** and **Miguel Ballesteros** introduced me to the key ideas that led to Chapter 8. I'm also grateful to the many experts from other institutions whom I encountered at NLP conferences, most notably the annotation masters **Ines Rehbein**, **Heike Zinsmeister**, and **Bonnie Webber**, and the construction grammar crew, including **Adele Goldberg**, **Remi van Trijp**, **Bill Croft**, and **Nancy Chang**. In addition to their invaluable technical insights, these latter groups gave me a much-needed vote of confidence and reminded me why I love NLP.

Of course, just as much critical support came from my peers. A special, heartfelt shout-out goes to **Nathan Schneider**, without whose sage advice and unflagging enthusiasm I would not have managed half the research I did. Additional advice and encouragement came at crucial moments from **Daniel Leeds**, **Alona Fyshe**, **Dana Movshovitz-Attias**, and **Prasanna Kumar**. I benefited from many stimulating conversations with others in Lori's LLab group, including **Chu-Cheng Lin**, **Archna Bhatia**, **Jeffrey Micher**, and **Pat Littell**. Comments from **Todd Snider** and **Shing-hon Lau** elevated this document far above the drafts they reviewed. And I can't count the number of times I leaned on my intrepid housemates, **Ardon Shorr**, **Adona Iosif**, and **Carlton Downey**. Carlton in particular has guided me through many a machine learning pickle.

Adona and **Ardon** deserve enormous recognition for another role: leaping headfirst with me into founding Public Communication for Researchers, a project that infused much meaning into my graduate years. Their passion and professionalism have been an inspiration, and my career would be immeasurably impoverished without them. A thousand thanks likewise go out to **Julian Whitman**, **Kelly Matula**, **Jay Wang**, **Lauren Hayward**, **Mark Cheung**, **Djuna Gulliver**, **Daniel Gingerich**, **Prasanna Kumar**, **Reuben Aronson**, **Kirstin Early**, and all the other students who have led and participated in PCR's programming. Thanks also to the administrators who assisted us along the way—**Suzie Laurich-McIntyre**, **Amy Burkert**, **Joanna Wolfe**, **Korryn Mozisek**, and **Daniel Dickson-Laprade**—as well as everyone in the science communication community who cheered us on.

The annotation work described in Chapters 4 and 5 hinged on the masterful efforts of my annotators: **Ari Klein**, **Peter Boyland**, **Donna Gates**, **Jeremy Dornboos**, **Spencer Onuffer**, **Nora Kazour**, and **Mike Mordowanec**. Artisans like them are the unsung heroes of language technologies. I hereby sing their praises.

The administrative staff of the School of Computer Science have been a true blessing, smoothing out so many wrinkles in the day-to-day life of a graduate student. I'm sure I'm omitting many who work behind the scenes, but many thanks to **AnnMarie Zanger**, **Kate Shaich**, and **Catherine Copetas**, and especially to **Deb Cavlovich**, who acts as the indispensable mom of the entire Computer Science Department.

I owe a substantial debt of gratitude to **Schenley Park**, which has played an outsize role in my thesis. The park's woods were a perfect place for reading papers, reflecting, and restoring calm, and I credit them with getting me through some of the toughest moments of my Ph.D. Additionally, the majority of this document was written (or at least adapted from previous writings) on various park benches in Schenley. I send a hearty thank-you to the **Pittsburgh Parks Conservatory** for maintaining the park in such a wonderful state.

Finally, of course, I can never adequately thank my family, both biological and chosen. **My parents** passed down their deep love of learning and thinking, and have supported me in every way as I've pursued the passions and ambitions they instilled in me. And to my wife, **Marli**: every day of my graduate career since we met has been better for your being there. You've seen me through it all, and I only hope I'll manage to support you as thoroughly and lovingly as you've supported me.

Contents

List of Figures

List of Tables

CHAPTER 1 Overview

1.1 Introduction

Humans write an awful lot of text. As of the completion of this thesis (2018), the Internet contains in excess of 4.4 billion webpages,[1] and it's growing at an ever-faster rate. Our species has produced around 60 million journal articles[2] and more than 130 million books.[3] And that's not to mention the millions of social media posts, the tens of millions of text messages, and the hundreds of millions of emails composed every **minute**.[4] If we want to make good use of all that information—to react to the latest news, to ascertain the scientific consensus about a medication, or even just to remember a spouse's request for groceries on the way home—we're going to need some help from our computers. We need software that can decipher the meaning—the SEMANTICS—of language.

Given that we humans understand language without a second thought, teaching computers to do the same might seem straightforward. But despite decades of effort from the artificial intelligence community, this goal has proved remarkably elusive. Nonetheless, the field of natural language processing (NLP) has made major inroads by breaking down the problem into many smaller subproblems, formalized as TASKS. Each task defines another layer of interpretive information that a computer might want to add to its rudimentary understanding of the text.

One of the most well-studied of these tasks has been SHALLOW SEMANTIC PARSING, henceforth referred to as SEMANTIC PARSING. ("Shallow" here refers to the fact that to perform the task, a parser needs to build representations only of the argument structures of individual words. This contrasts with richer forms of semantic parsing, in which the system must build a full-fledged logical representation of all the information in each sentence—negations, nested relationships, etc.) To perform this task, an automated system combs through a document for instances of predetermined semantic relationships. Consider, for example, Terry Pratchett's depressingly accurate aphorism, "Real stupidity beats artificial intelligence every time." A semantic parser's primary duty would be to produce an analysis like the one in Figure 1.1.

[1] http://www.worldwidewebsize.com/

[2] http://onlinelibrary.wiley.com/doi/10.1087/20100308/. The number was estimated at 50 million in 2010, and the rate of increase at that time was over 1.4 million per year.

[3] http://mentalfloss.com/article/85305/how-many-books-have-ever-been-published

[4] http://aci.info/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic/

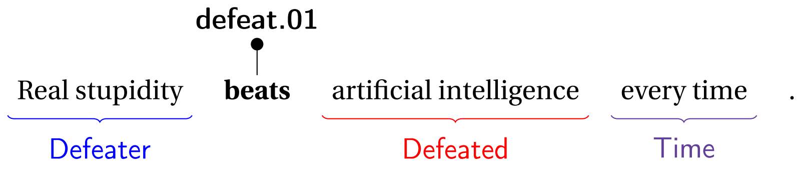


Figure 1.1: A sentence with example shallow semantic parsing annotations.

.

Here, *beats* has been identified as a TRIGGER—a word that expresses a relationship of interest. *Beats* has also been disambiguated: it has been marked as expressing a relationship of defeating (and not, say, mixing vigorously). In addition, the trigger's ARGUMENTS—the spans of text that indicate what is defeating and what is being defeated—have been tagged. (Throughout this thesis, I use the term SPAN to refer not just to a contiguous block of text, but to any collection of words or even characters in a text that are treated as a unit.) The sentence can thus be thought of as expressing a logical statement of the form Defeat(Defeater, Defeated). If possible, a semantic parser might also want to extract additional metadata about the relationship, such as a description of the time of defeat (which in this case is described in terms of its regularity).

In this thesis, I will argue that this paradigm, though fruitful, is limited in two important ways. First, it assumes that the carriers of meaning in a sentence are individual words. This is often true, but not always; some kinds of semantic relationships can be expressed with more varied linguistic forms, ranging from the individual morpheme to syntactic structures that span multiple clauses. For instance, the enablement relationship in *if we are to succeed, we'll need everyone's cooperation* is conveyed by the combination of *if*, *are to*, and *need*. Existing systems' focus on words limits their ability to capture the full range of such expressions. Second, the semantic parsing paradigm assumes that each word should be mapped only to one meaning, when a given word may signify multiple semantic relationships at once. Both challenges are particularly prominent in the type of language the thesis will focus on: CAUSAL LANGUAGE, which describes what enables, facilitates, inhibits, or prevents what. The thesis will demonstrate an approach to semantic parsing for causal language that expands the task to include more varied linguistic triggers, and paves the way for handling multiple simultaneous meanings.

To achieve this, we will need to go one level deeper than parsing techniques, down to the level of the data structures the parsers produce. In most NLP tasks, including semantic parsing, the desired data structures are specified by some ANNOTATION SCHEME—a rigorously defined system of guidelines for layering interpretive information on top of the text. In the previous example, the annotation scheme might dictate that every verb should be marked as a trigger; that each trigger should be tagged with its most relevant definition; and that the subject, the object, and any prepositional phrases should be labeled with the semantic roles they perform with respect to that verb.[5] A parser typically learns to produce these structures by being TRAINED on a large number of example documents annotated by humans—an ANNOTATED CORPUS. Portions of the same corpus are held in abeyance and used to evaluate how well the parser performs on data it has not previously seen.

Thus, each parser's output space and learning ability can only be as rich as the annotation scheme the parser is based on. This means that we can trace the limitations on existing semantic parsers' expressiveness directly to the limitations of their underlying annotation schemes. If we wish to enrich the task of semantic parsing, we will need a richer annotation scheme, as well.

[5]This is a rough approximation of the much more extensive guidelines for PropBank, an annotation scheme and corpus commonly used for semantic parsing; see §3.3 below.

This thesis will therefore consist of an overview followed by three parts. In this chapter and the next (Part 0, if you will), I motivate the need for extending semantic parsing and sketch the overall approach and thesis statement. In Part I, I describe a new approach to annotation. The approach is designed to capture semantic relationships whose linguistic expressions vary widely and may express additional co-present meanings. I describe a detailed annotation scheme and corpus created for causal language. In addition to causality, the scheme captures multiple layers of non-causal meaning that can accompany causal language. Part II then presents several semantic parsers we[6] have built on top of the annotation scheme, and discusses the performance and implications of these systems. Finally, Part III closes with some lessons learned and thoughts on how this approach to semantic parsing can be taken forward.

1.2 Motivation: causal language aids many downstream applications

The promise of semantic parsing is that, by finding instances of known relationships in text, a user (possibly an automated user) can discover fragments of meaning in a set of documents—assertions about real-world entities or facts and their relationships. These fragments can then enhance downstream computational tasks, such as translation, information extraction, and summarization, or they can directly help a human discover relevant information.

To this end, it is particularly useful to discover assertions about a particularly fundamental and ubiquitous type of relationship: cause and effect. Causal relationships pervade our thinking. They underpin our very understanding of our world, and we often want to know what causes, enables, or prevents medical symptoms, political events, interpersonal actions, and the other phenomena we observe. It is unsurprising, then, that causation is also ubiquitous in language. Conrath et al. (2014), for example, found that causation constituted 33% of the relations expressed between verbs in a corpus of French text from the web. Similarly, in the Penn Discourse Treebank (PDTB; Prasad et al., 2008), a corpus of documents with annotations for relationships between clauses and sentences, over 12% of explicit discourse connectives are marked as causal, as are nearly 26% of implicit relationships (i.e., relationships between sentences or clauses that are juxtaposed without an explicit discourse marker). Of course, causality is hard to define precisely, so these numbers should not be taken as gospel, but they demonstrate the frequency with which writers appeal to the intuitive notion of causality.

Recognizing these relations is thus invaluable for many kinds of applications. The most straightforward applications rely on extracting causal relations and storing them as a knowledge resource. For example, one of the biggest challenges for the widely studied task of recognizing textual entailment (RTE)—the task of determining what assertions can be inferred from a text—is that the systems cannot reason about causal relations (Sammons et al., 2011). Likewise, many of the questions posed to question-answering (QA) systems are about causes and effects (see Girju, 2003). Systems that had access to pre-extracted cause/effect pairs would be better prepared for these applications.

Many applications, however, rely not just on causal relations extracted from large, redundant corpora, as is typically done in information extraction, but on correctly analyzing the semantics of specific pieces of causal language. One such example is the question interpretation side of QA: when faced with a question like *what are the reasons for California's fires?*, a QA system needs to map it to the relevant causal relationship. Similarly, semantic interpretation is a concern in paraphrasing, text simplification, and machine translation, where the output of the system must at least approximately match the original meaning of the input. Because causal language is so common, it is essential for

[6]Throughout this document, I use "we" to describe joint work with my advisors and others. I use "I" when discussing the presentation of this thesis, or occasionally to identify a role I in particular played in the research process (e.g., an annotator).

(1) THIS BILL **promotes** *consolidation and cooperation among regulatory agencies*.
(2) SUCH SWELLING can **impede** *breathing*.
(3) WE DON'T HAVE MUCH TIME, **so** *let's move quickly*.
(4) *We did it* **because** WE FELT IT WAS OUR DUTY.
(5) *He died* **from** A BLOCKED ARTERY.
(6) *Making money* is **contingent on** FINDING A GOOD-PAYING JOB.
(7) THIS DECISION **opens the way for** *much broader application of the law*.
(8) **For** *market discipline* **to** *work*, BANKS CANNOT EXPECT TO BE BAILED OUT.
(9) Judy's comments were **SO** OFFENSIVE that *I left*.

Table 1.1: Examples of causal language, reflecting the annotation scheme described in Chapter 4. **Triggers** (connectives, in the scheme's terminology) are shown in bold, CAUSES in blue small caps, and *effects* in red italics.

systems performing these tasks to identify instances of causal language, and what kind of causal relation was intended in each. For example, simplifying example 9 in Table 1.1 into *Judy's comments were very offensive. I left.* would lose an essential part of the meaning, as would turning *cleared the way for* into *led to*.

There are also domains where highly specific causal relationships, expressed only once in a document or corpus, can directly drive human decision-making. For example, in politics, finance, and biology, analysts might be looking for the causes or effects of a particular event or chemical. Causal relationships in particular have been shown to be important for reading comprehension in these domains, as the events and entities are linked together by complex chains of causations, enablements, and preventions (see, e.g., Berant et al., 2014).

1.3 Lexical units with single meanings do not fully capture causal language

Tagging causal relationships is thus an important semantic parsing task, but it is far from straightforward: these relations can take a tremendous variety of linguistic realizations (Wolff et al., 2005). As demonstrated by the examples in Table 1.1, causal relations can be expressed using anything from verbs (1, 2) to conjunctions (3, 4), prepositions (5), adjectives (6), and much more complex expressions. Some of these trickier cases can be handled as multiword expressions (MWEs; 7), but others (8, 9) rely also on the syntactic relations between multiple parts of the sentence.

This wide-ranging variation presents a problem for most existing semantic parsers, which inherit the limitations of their underlying representational schemes. Many of these schemes restrict themselves to tagging the arguments of particular classes of words. The PDTB, for example, which is intended to capture relationships between clauses or sentences, includes only conjunctions and adverbials as linguistic triggers. Likewise, PropBank (Palmer et al., 2005) and VerbNet (Schuler, 2005) focus on representing the arguments of verbs. FrameNet (Baker et al., 1998) includes more varied linguistic triggers, but still requires them to be LEXICAL UNITS (LUs): either words or phrases that behave like words. Most importantly, then, all of these representations share the fundamental simplifying assumption that the basic carrier of linguistic meaning is the word or its equivalent.

Some representations (e.g., PDTB and FrameNet) do allow multi-word expressions (MWEs) as lexical units, and much work has been done on detecting and interpreting MWEs (see Baldwin and Kim, 2010). But even these schemes must overlook essential linguistic elements that encode meanings. In example

(10)	a.	**After** a drink, she felt much better.
	b.	**After** a drink, she went to bed.
(11)	a.	The invention of the telegraph **allowed** the nascent industry to flourish.
	b.	The king **allowed** the nascent industry to flourish.
(12)	a.	**If** you follow me, we might just get out alive.
	b.	**If** he was willing to follow you, he must really trust you.
(13)	a.	A series of kung fu movies **launched** the "kung fu craze" in America.
	b.	Senate Republicans **launched** a judicial coup last February.

Table 1.2: Examples of language that includes an element of causality, but is intertwined with other semantic phenomena. Possible causal triggers are bolded. Each pair shows a causal and non-causal use of the same trigger.

9, for instance, an LU approach would have to treat *so* as encoding the causal relationship,[7] when in fact *so* merely intensifies the adjective; it is the combination of *so* and the finite clause following the adjective that indicates causality.

Causal language also highlights another shortcoming of traditional shallow semantic analysis: generally, each LU token is tagged only with a single semantic relation. This leads to difficulty with borderline cases, where an expression bears elements of multiple relations. Causal language includes many such cases, as exemplified in Table 1.2: causality jockeys for linguistic machinery that is also shared by temporal relations (10), obligation (11), conditional dependence (12), and bringing into existence (13), among others. Under a typical semantic parsing scheme, each of these examples' triggers would have to be neatly categorized as causal, temporal, etc., even though aspects of more than one relation may be present.

Thus, semantic parsing for causal relations calls for a richer representation—one flexible enough to uniformly handle a wide variety of triggers, and ideally rich enough to capture the presence of multiple overlapping semantic relationships.

1.4 A way forward: Construction Grammar (CxG)

Inspiration for such a representation can be found in the principles of CONSTRUCTION GRAMMAR (CxG; Fillmore et al., 1988; Goldberg, 1995). CxG posits that the fundamental units of language are CONSTRUCTIONS: pairings of meanings (or functions) with arbitrarily complex linguistic forms. For example, the construction for the verb *push* pairs the verb stem with the meaning *force to move* (see Figure 1.2[a]).

Some elements of a construction, such as the verb root *push*, are fixed. But a construction may also include open slots that can be filled by other words or constructions—i.e., constructions can be LINGUISTICALLY PRODUCTIVE. Continuing with the same example, *push* takes two linguistic arguments, subject and object, corresponding to the two semantic arguments, the pusher and the pushee.

As in traditional lexical semantics, the forms that are paired with meanings can be words or morphemes. But in CxG, many different kinds of forms can carry meanings. For instance, *so X that Y* is characterized by a single construction (see Figure 1.2[b]), where the form is *so* ⟨*adjective X*⟩ ⟨*finite clausal complement Y*⟩ and the meaning is *X to an extreme that causes Y*. Even passivization (e.g., turning the transitive verb *sing* into a passive like *A song was sung by John*) and the existential *there*

[7]This is in fact exactly what FrameNet does; see the Sufficiency frame.

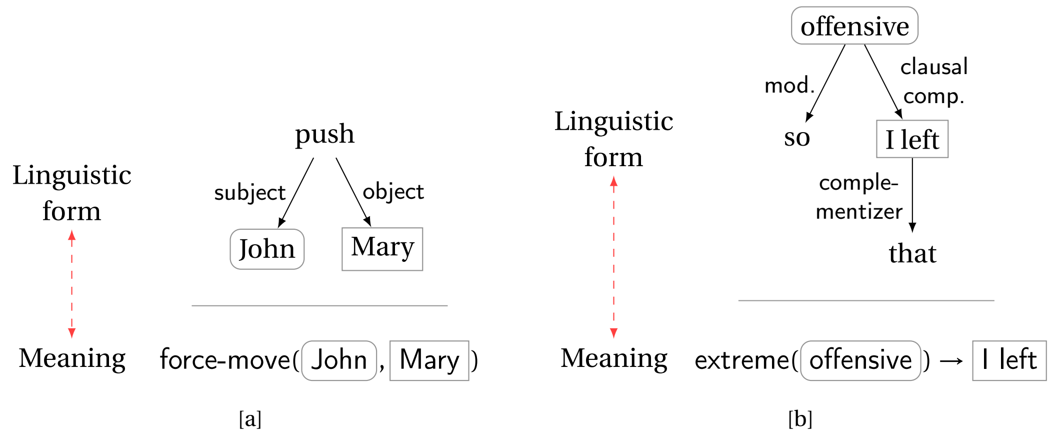


Figure 1.2: Schematic visualizations of the constructions corresponding to the verb *push* and the pattern *so X that Y*. The linguistic forms shown are Universal Dependencies (Nivre et al., 2016) parse tree fragments.

(as in *there are three reasons…*) are considered constructions, to be merged with individual word and phrase constructions to produce the final sentence and its meaning. Essentially, CxG expands the notion of the lexicon to include not just words and their components, but any other linguistic unit or pattern that carries meaning.

This means constructions can include fixed lexical items, syntactic relationships, and even subtler linguistic features such as requirements for modality (the mechanisms used by a language to describe situations that may not be real). Consider, for example, the construction *if X is to Y, Z*, as in *if we are to survive, we must face the truth.* The intended meaning of this construction is that Z is a precondition for X doing Y. On the surface, this is very similar to a normal conditional construction, as in *if you sleep, you'll feel much better*—except that the semantics are reversed. In a normal conditional, the first clause is a precondition for the second (sleeping is a precondition for feeling better). But in our example, it is exactly the opposite: facing the truth is a precondition for surviving.

The cues that we use to recognize which construction is in play are illustrated in Figure 1.3. These include the fixed words (*if, is, to*), the syntactic relationship between X and Y, and the modality of obligation expressed in Z. Indeed, any phrasing that expresses this modality would work equally well: *we will have to face the truth, we need to face the truth*, etc. The surface form that signals the *if X is to Y, Z* construction is a combination of all these elements.

Even as CxG has established a place for itself in linguistics, it has received relatively little attention in the world of NLP. Yet its flexibility makes it potentially invaluable for computational semantics. For example, in machine translation, phrase-level alignments between parallel texts become difficult or impossible when they would break up a construction in one of the languages: an idiomatic phrase like *no one has a finger on the scale* might have been translated as *everyone acts justly*, yielding no sensible way to align the component phrases (see Deng and Xue, 2014). Without construction-level information, alignment must be limited to much coarser syntactic units, dropping some of the meaning. The time is ripe for CxG to be incorporated into such NLP tasks and methods.

In our case, what makes the CxG paradigm useful is that it can anchor semantic interpretations to an enormous variety of surface forms. This enables a new approach to semantic annotation and parsing: traditionally, NLP organizes systems around a single linguistic phenomenon (e.g., verb arguments or discourse relations). But drawing on CxG, we can instead start from a particular type of semantic relation (e.g., causation), and identify all the ways it is expressed across all levels of linguistic structure—all the

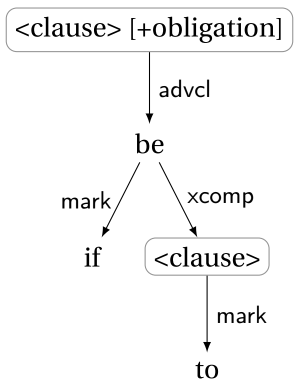


Figure 1.3: Schematic visualization of the linguistic form for the construction *if X is to Y, Z*. The form is shown as a Universal Dependencies tree.

constructions sharing that meaning. We can also identify the semantic domains that compete for the same linguistic structures, including cases where the surface form indicates multiple meanings at once.

Because this CxG-driven approach treats all expressions of a relation uniformly, it is more conducive to semantics-sensitive applications such as information extraction and machine translation. It is also ideal for representing phenomena such as causal language that are linguistically varied and intertwined with other semantic domains. I will therefore employ CxG as the basis of my enriched representation for semantic parsing.

1.5 The "surface construction labeling" approach

There are many reasons why CxG has seen little uptake in NLP. Most obviously, CxG-based systems (e.g., Fluid Construction Grammar [Steels, 2012] and Embodied Construction Grammar [Bergen and Chang, 2005]) have not yet reached the level of maturity and robustness exhibited by conventional linguistic analysis systems. Another factor, though, seems to be the perceived barrier of rebuilding the entire NLP ecosystem based on "constructions all the way down" (see Goldberg, 2006). The NLP community already has mature, well-studied tools that identify words, parse them into syntactic structures, and assign semantic interpretations to pieces of those structures. Until fully constructional systems offer competitive robustness, researchers who need automated linguistic analysis are unwilling to leave the conventional tools aside. And in the short term, developing constructional tools for well-studied tasks seems like stepping backwards and retreading ground, even if doing so holds the potential for long-term advances.

Rather than rewriting the standard NLP pipeline, though, this thesis proposes a middle ground: to borrow the key insights of CxG and apply them **on top of** traditional NLP techniques. We take those central insights to be:

1. Morphemes, words, MWEs, and grammar are **all on equal footing** as "learned pairings of form and function" (Goldberg, 2013); to the extent that they differ, they do so only in their gradations of complexity and/or fixedness.
2. Constructions pair patterns of surface forms **directly with meanings**.

To accomodate these insights, we do not need a full implementation of CxG theory. We need merely to expand the targets for semantic parsing from LUs to the fixed portions of constructions—the words[8] that consistently appear when the construction is in use. Semantic interpretations can then be tied to instances of constructions, rather than to lexical units. This **"surface construction labeling" (SCL) approach** to NLP bestows automatic taggers with much of the representational flexibility of constructions, while still allowing the systems to rely on the usual robust part-of-speech taggers, syntactic parsers, and so on as approximations of the underlying constructions.

1.6 Thesis and approach

This thesis presents methods for representing and automatically tagging linguistically variable semantic relations in English, based on the SCL approach to applying the principles of construction grammar. My hypothesis is that, using this approach, we can:

1. Construct **richer, more flexible linguistic representations** than are available under traditional NLP schemes. These representations can capture **linguistically variable semantic relations** in a uniform manner, and can represent **multiple simultaneous meanings** of a linguistic expression.
2. **Manually annotate corpora** using these representations to a high degree of reliability.
3. Build **automated machine learning taggers** for constructional realizations of semantic relations.

I will use causal language and the phenomena that intersect it as a testbed for this hypothesis. As we have seen, causation is a common and semantically important phenomenon, and exhibits the sort of linguistic variation that demands a constructional approach (as do the semantic domains that overlap with it). I will demonstrate the hypothesis in three steps, corresponding to the three parts of the hypothesis:

1. **Define an annotation scheme** in which all constructions that can indicate causality are tagged (Chapter 4). I operationalize constructions as lexico-syntactic patterns (that is, patterns including both words and syntactic relations), where some elements are fixed and some can vary. The annotation scheme specifies what usages should be included as instances of the relation, and what spans and metadata should be annotated.
2. **Annotate a corpus** following this annotation scheme, marking instances of the relevant constructions (Chapters 4–5). As part of this process, we compile a CONSTRUCTICON—a list of all constructions we have found that can encode causation. Annotators consult this master list of constructions when deciding what instances to annotate and what to include in the annotation. By setting up the annotation process this way, we keep annotators from having to make the same decisions repeatedly about which constructions should be included. The result is a substantial corpus of consistently annotated text.
3. **Build automated systems** that can be trained on the annotated data to recognize new instances of these construction patterns. I offer two distinct approaches:
    a) A multi-stage pipeline that first detects possible construction instances using data-derived lexico-syntactic patterns, fleshes out their arguments, then filters out false positives with

[8]Of course, as stated in insight 1 above, sub-lexical morphemes or even grammatical structures with no lexical component are equally valid indicators of constructions. For this thesis, annotating and automatically tagging such constructions remains an aspiration for the future, but see §9.2.1 for some discussion of how they might be handled.

statistical classifiers (Chapter 7). This approach is hand-tuned to the particulars of tagging causal language.

b) A neural network that builds up instances of causal language as it scans each sentence word by word (Chapter 8). This method is more general-purpose, making fewer assumptions about the data.

The systems achieve up to 53% and 60% F_1, respectively, on the task of finding instances of causal language, with high accuracy in identifying the cause and effect spans. We expected that parse information would prove essential and that the hand-tuned, pattern-based methods would perform better than the neural network on a few coherent clusters of cases. To our surprise, both of these hypotheses turned out to be ill-supported by our experiments and error analyses (§8.6–§8.7).

In the remainder of the thesis, I will delve into each of these steps in detail. Part I will present the annotation schemes and annotated corpora (steps 1 and 2 above), while Part II will describe the techniques for automatically tagging novel text with these annotation schemes (step 3). I will conclude with a discussion of the implications of this work for construction-based semantic annotation and automated tagging, both in English causal language and more broadly (Part III).

1.7 Summary of contributions

The key contribution of this thesis is the SCL approach, which plays out both in annotation and in automation:

- On the annotation side, we demonstrate that the approach is a practical way to upgrade the representational power of semantic annotation so that it can handle arbitrarily complex triggers and multiple overlapping relations. We achieve this specifically in the semantic domain of causal relations, which have proven remarkably difficult to annotate. This is possible thanks to our emphasis on what the text asserts to be causal, rather than what is causal in the real world.
- On the automatic tagging side, our semantic parsing methods offer great flexibility with regard to the spans of and the overlaps between triggers and arguments. This level of flexibility is unique among systems that follow the semantic parsing paradigm of finding each instance of a relation of interest and assigning an interpretation.[9]

These contributions are embodied in publicly downloadable annotated corpora and tagging systems.

Through all of the above, the thesis lays the groundwork for an SCL-based research agenda—a program of incorporating CxG directly into semantic parsing and other NLP applications.

[9]See Table 6.2 for a detailed comparison of our methods' capabilities against others'.

CHAPTER 2

A Construction Grammar Primer

Before we proceed further on annotating and automatically tagging causal language, let us first review the key elements of construction grammar, which will provide important context for this work.

The most widely known theories of the mechanics of language are those in the tradition of TRANSFORMATIONAL GRAMMAR, particularly the Chomskyan tradition. These theories assume that any given sentence is arrived at by combining more basic units into successively more complex ones through a series of composition and transformation operations. Morphemes (word roots, prefixes, etc.) and words are merged into phrases and clauses, and the rules of syntax combine independent phrases into larger phrases. As larger units are built out of smaller ones, their meanings are determined by COMPOSING the meanings of their constituents. In the words of Michaelis (2013), “Grammar is organized in the mind of the speaker as a number of hermetically sealed modules, which in the course of a derivation hand off data one to the other.”

The permissible larger phrase structures are drawn from a limited list canonical forms. Variants of these forms are generated through syntactic TRANSFORMATIONS, which produce, e.g., passive clauses from active-voice ones. Transformations can add, move, or duplicate the elements within a syntactic tree. The formal rules for these transformations necessitate a large inventory of invisible placeholders and markers to ensure consistent syntactic structures for the transformations to operate on.

CxG developed in contradistinction to such theories. Rather than relying on invisible elements and transformations, it ties meanings tightly to the surface forms used to express them. And rather than assuming that any larger phrase must be neatly composed from independent morphological and syntactic modules, CxG admits arbitrarily complex patterns as first-class citizens. In CxG, every conventionalized linguistic pattern, whether lexical, syntactic, morphological, or some combination thereof, is just another construction—a pairing of form and meaning—somewhere on the lexicon/syntax continuum. CxG thus posits a single set of mechanisms to explain linguistic phenomena at all levels of complexity: constructions, along with the machinery for merging their meaning and form sides to form a final sentence and its meaning.

In this preliminary chapter, I review existing theoretical and computational work on CxG, with an emphasis on aspects that provide inspiration for this thesis. Specifically, I:

- summarize the **motivations and core principles** of CxG (§2.1);
- sketch the **major theoretical frameworks** (§2.2);
- outline **existing computational implementations of CxG** (§2.3); and

- establish **what this thesis draws from CxG** (§2.4).

2.1 Motivations and core principles of CxG

CxG was born out of the following realization: to the extent that attempts to formalize grammar have succeeded, they have done so largely by ignoring huge swaths of real-world language use (Hoffmann and Trousdale, 2013). In the Chomskyan tradition, the object of study is an abstracted, idealized version of language—"the core system, putting aside phenomena that result from historical accident, dialect mixture, personal idiosyncracies," and other "peripheral" phenomena (Chomsky and Lasnik, 1993). Yet idiosyncrasies are not occasional exceptions to the rule; they often **are** the rule. Corpora are riddled with fixed or semi-fixed idioms (e.g., *X takes Y for granted*) and peculiar, context-specific grammatical constructions (e.g., *nth cousin Y times removed*, *that's not to mention*, *for all its flaws*; see Fillmore et al. [2012]). While the generative tradition has proposed supposedly universal mechanisms to handle some of these cases, they often seem awkward or ad-hoc (e.g., the "small clause" analysis of resultatives, as in Carrier and Randall [1992]).

CxG was created to provide a uniform analysis framework for the full spectrum of language use, treating all of it as equally part of language users' mental representation of the language. CxG aims to be both empirically adequate, explaining observed linguistic behaviors and data, and cognitively plausible, matching what is known about how humans actually process language.

Goldberg (2013) lays out four key assumptions that characterize constructionist approaches to grammar:

1. **Grammatical constructions and word roots are all "learned pairings of form and function."** They are thus equal citizens of the mental lexicon, lying on the same continuum of constructional complexity. Some constructions, such as nouns, are standalone, possessing no internal structure and a self-contained meaning. Others, like the passive verb construction, must be merged with other CONSTRUCTS—instantiations of constructions—to form a complete form/meaning pairing. There is no privileged distinction between syntax and lexicon or even morphology; indeed, some constructions contain elements of all three.

   "Grammatical constructions" here includes those that exhibit limited generalization. For example, the construction ⟨*preposition*⟩ ⟨*bare count noun*⟩, as in *she's* ***in prison*** or *go* ***to bed***, denotes participating in an activity stereotypically associated with the noun. While the construction can be applied to many nouns, its productivity is limited: one cannot go *to bath* or *to airport*, and Americans (unlike Brits) cannot go *on holiday* or *to hospital*. A construction is taken to include a set of constraints, some systematic and some idiosyncratic, that describe what constructs it can be applied to.

2. **What you see is what you get,** i.e., surface forms are tied directly to the semantics, rather than reflecting hidden layers of intermediary transformations. If a pattern of form/meaning correspondences is observed in surface forms, a construction grammarian does not follow the transformational grammarian in positing syntactic transformations of deeper, more universal structures that would produce the pattern as a by-product. Rather, he or she takes that pattern to be a distinct construction that carries its own meaning and properties. For example, *Mina sent a book through the metal detector* shares structural similarities with *I coughed the moth out of my mouth* and many other such constructs, suggesting that speakers' mental lexicon includes a verb-general "caused motion" construction, rather than a transitive sense of *cough*.

3. **Constructions are organized in a structured network,** with more specific constructions (e.g., the verb *hit*) inheriting default properties from more general ones (e.g., transitive-verb). A second important relationship between constructions is meronymy: constructions can select for particular kinds of constructs to fill their open slots. Thus, ⟨*preposition*⟩ ⟨*bare count noun*⟩ inherits most of the usual properties of prepositional modifiers (e.g., that the preposition precedes the noun), but with some modifications to the constraints on slot-fillers (e.g., a determiner is forbidden rather than required).
4. **Languages vary widely with respect to what constructions they incorporate.** Of course, there are some crosslinguistic similarities—e.g., there is a practical need for language users to describe actions whose agent is unspecified or de-emphasized, so many languages have converged on some version of agentless passive verbs. But the specifics of each language's version of a construction, or even whether the construction exists at all, may differ greatly.

In addition to these core principles, CxG emphasizes the role of repeated usage in establishing and learning constructions. Speakers observe recurring patterns, even fully compositional ones like *sooner or later*, and come to store them as their own units. These constructions can then take on idiosyncratic meanings or linguistic properties. They can also be generalized into more abstract constructions like subject-auxiliary inversion (***Did you*** *eat?*). While it is debatable what qualifies a pattern to be an independent construction (see §4.5.3), the key factors are conventionality and frequency of usage.

Under this framework, it is no more difficult to analyze idiosyncratic constructs like *For all its flaws, I still love it* than it is to analyze *John waved to Mary.* The grammar can simply posit an idiomatic construction with two fixed words *for all* and open slots for a noun phrase and a clause, and with a concessive meaning. The *for all* construction merges with the other constructions that produce the slot-fillers, with the associated meaning representations being merged in parallel. Of course, the merging is subject to general constraints about how constructions can combine and construction-specific constraints about what contexts and slot-fillers the *for all* construction expects.

2.2 Prominent CxG theoretical frameworks

CxG is really a family of theories, and a number of versions of it have been fleshed out. Here, I provide a sample of some of the most prominent strands of thought.

Argument realization constructions and Cognitive CxG Much of the work on CxG, particularly in its earliest forms, focused on describing the argument structures of verbs. Initiated by Fillmore et al. (1988) and developed further by Goldberg (1995), this line of research aims to explain when and why verbs can appear in constructions like CAUSED MOTION (*sneezed the napkin off the table*) and the CONATIVE construction (*kicked* ***at*** *the table*).[1] Goldberg proposes that a full construct is produced by fusing the construction for an individual verb with a meaning-bearing argument structure construction. This fusion is permissible as long as the constraints and meaning representations of the two constructions do not conflict. For example, the ditransitive construction (*sent me a letter, baked me a cake*) is interpreted as causing the direct object to receive the indirect object, and can unify with any verb with a causing agent and an object that can be received. Goldberg also proposes a suite of inheritance relations between different constructions, such as part/whole and metaphorical extension.

[1]Although this approach has been most thoroughly explored in verb argument structure, it has also been applied to other constructions that rely on pragmatics, such as the deprofiled object construction (e.g., *tigers only kill at night*).

$$\begin{bmatrix} \textsc{phon} & \text{/kɪm/} \\ \textsc{syn} & \text{PN} \\ \textsc{sem} & \text{‘the intended person named Kim’} \end{bmatrix}$$

[a] An informal version of the SBCG feature structure for the name *Kim*. PHON indicates phonological features, SYN syntactic features (in this case, “proper noun”), and SEM semantic features.

$$\begin{bmatrix} \textit{subject-predicate-construct} & \\ \textsc{mother} & \begin{bmatrix} \textsc{phon} & \text{Kim, laughed} \\ \textsc{syn} & \text{S}[\textit{finite}] \\ \textsc{sem} & \ldots \end{bmatrix} \\ \textsc{daughters} & \begin{bmatrix} \textsc{phon} & \text{Kim} \\ \textsc{syn} & \text{NP} \\ \textsc{sem} & \ldots \end{bmatrix}, \begin{bmatrix} \textsc{phon} & \text{laughed} \\ \textsc{syn} & \text{V}[\textit{finite}] \\ \textsc{sem} & \ldots \end{bmatrix} \end{bmatrix}$$

[b] An SBCG feature structure representing a subject/predicate construct—the subject/predicate construction instatiated with the sign *Kim laughed.*

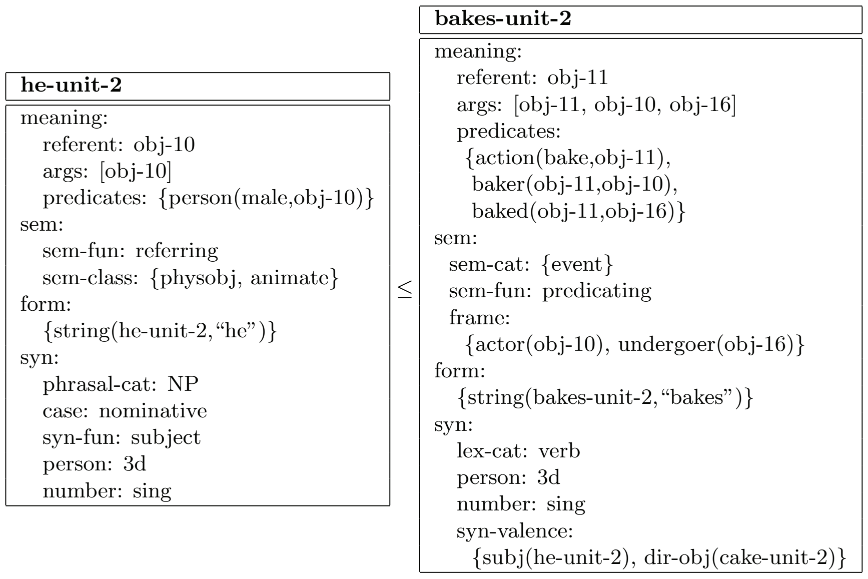
Figure 2.1: Sample SBCG feature structures (from Sag and Boas, 2012).

This approach to CxG prioritizes cognitive plausibility over formal rigor, seeking evidence from psychological experiments. It also highlights usage statistics as an important element of a speaker’s mental “constructicon” entries. Goldberg’s approach has therefore become known as Cognitive CxG (Boas, 2013).

Sign-Based Construction Grammar Sign-Based Construction Grammar (SBCG; Sag and Boas, 2012), an outgrowth of Berkeley Construction Grammar (BCG) and Head-Driven Phrase Structure Grammar (HPSG), emphasizes formal representations and rigor, providing a formal substrate with which CxG theoreticians can express and test hypotheses. As in HPSG, the core device of SBCG is the TYPED FEATURE STRUCTURE, represented visually as an attribute/value matrix (see Figure 2.1). A construction is represented as feature structure template. Constructions specify constraints (e.g., requiring a noun phrase slot-filler) by specifying the type of feature structure that is required for a given attribute. Feature structure types are arranged in a hierarchy to allow default inheritance of constraints and properties.

Feature structures in SBCG are used to represent two major classes of objects: SIGNS, or words and phrases (Figure 2.1[a]); and CONSTRUCTS, which in SBCG means specifically a local combination of signs in some grammatically licensed arrangement (Figure 2.1[b]). Parsing a sentence involves identifying the relevant signs and constructions and unifying their feature structures into a single overarching feature structure for the sign consisting of the entire sentence.

Radical construction grammar Radical construction grammar (RCG; Croft, 2001) is a semi-formal framework for describing constructions and their interactions that emphasizes crosslinguistic diversity. It is “radical” in the sense that it denies the existence of global linguistic categories such as “noun” even within a language. Instead, it posits that all categories are specific to the language and construction: linguistic categories exist only as clusters of permissible components of other constructions. The categories RCG recognizes are ones like “argument 1 of *hit*,” which inherits from “argument 1 of transitive verb,” which itself may inherit from other more general categories that roughly correspond to “noun.”



```
he-unit-2
meaning:
   referent: obj-10
   args: [obj-10]
   predicates: {person(male,obj-10)}
sem:
   sem-fun: referring
   sem-class: {physobj, animate}
form:
   {string(he-unit-2,“he”)}
syn:
   phrasal-cat: NP
   case: nominative
   syn-fun: subject
   person: 3d
   number: sing
```

≤

```
bakes-unit-2
meaning:
   referent: obj-11
   args: [obj-11, obj-10, obj-16]
   predicates:
    {action(bake,obj-11),
     baker(obj-11,obj-10),
     baked(obj-11,obj-16)}
sem:
  sem-cat: {event}
  sem-fun: predicating
  frame:
    {actor(obj-10), undergoer(obj-16)}
form:
   {string(bakes-unit-2,“bakes”)}
syn:
   lex-cat: verb
   person: 3d
   number: sing
   syn-valence:
     {subj(he-unit-2), dir-obj(cake-unit-2)}
```

Figure 2.2: A sample FCG transient structure for the phrase *he bakes* (from Steels, 2017). The ≤ symbol denotes the meets relation.

2.3 Computational instantiations of CxG

We are not the first to try to operationalize CxG in a computational framework. To appropriate Goldberg's (2006) term, most existing projects approach language as “constructions all the way down”—i.e., they aim to build tools that analyze a sentence (or even a whole document) in terms of every construction that licenses it, down to atomic constructions.

Embodied Construction Grammar Embodied Construction Grammar (ECG; Bergen and Chang, 2005) is a computational formalism designed to connect constructions with perception and action in the real world. It draws from the “embodied cognition” hypothesis—the assertion that cognitive processes such as understanding language are not just a matter of shunting symbols, but are inseparably tied to interacting with a physical environment. ECG therefore represents the meaning side of constructions as EMBODIED SCHEMAS—“cognitive structures generalized over recurrent perceptual and motor experience”—which can be activated by linguistic constructions. A simulation engine takes in these activated schemas, along with any additional parameters specified by the sentence, and runs a simulation of a corresponding physical process. That simulation is the meaning representation.

The ECG project has built demonstration grammars that can perform impressive feats within a blocks world. In its present form, though, it has not been scaled up to anything resembling open text, and doing so would require an enormous amount of grammar engineering.

Fluid Construction Grammar The most substantial implementation of CxG to date is Fluid Construction Grammar (FCG; Steels, 2011, 2012, 2017). FCG aims to be as theory-neutral a formalism as possible while still enabling large-scale construction-based grammars. The FCG formalism centers on TRANSIENT STRUCTURES, an extension of feature structures, which can include any features a grammar engineer chooses to throw in. (Usually, of course, transient structures contain information about the form of a sentence and its meaning.) Each construction is represented as a transient structure schema containing some mix of fixed feature values and free variables to be bound during processing. An FCG grammar consists of an inventory of construction schemas that cover both lexical and grammatical constructions.

Importantly, the value of a feature can be not just a conventional data type, such as a boolean value or a sequence of feature sets, but a formula with predicates and operators. Order and adjacency information is represented with precedes and meets predicates between transient structure units, as in Figure 2.2.

The FCG engine supports comprehension and production, both of which are accomplished via a unification-based algorithm. Although no FCG grammar has yet been demonstrated to scale to large corpora, van Trijp (2017) recently demonstrated a precision FCG grammar for English. This effort marks the first practical, reasonably broad-coverage CxG parser.

Precursors to "surface construction labeling": NLP projects that use constructions without a full CxG framework We are aware of remarkably few projects that have adopted a similar approach to ours for incorporating CxG into computational tools.

One previous attempt to automate constructional analysis using traditional NLP representations and tools is that of Hwang et al. (2010) and Hwang and Palmer (2015), who focused on caused motion constructions (CMCs). They first searched a corpus of gold-standard syntactic parses for the syntactic pattern `(NP-SBJ (V NP PP))`, which is the non-exclusive signature of a CMC. They then annotated each match to indicate whether it was in fact a CMC, and built a statistical classifier to automatically predict CMC labels. Given a variety of gold-standard information, their classifier achieves performance on par with or even exceeding human annotators.

The only other similar effort we are aware of is Bakhshandeh et al.'s (2016) work on jointly predicting comparison and ellipsis constructions. They develop an annotation scheme for simultaneously marking the triggers and arguments of such constructions, where the first word or morpheme of a comparative construction is taken as the trigger. They are then able to achieve over 76% agreement on comparative predicate types (greater than, negative superlative, etc.) and ellipsis detection, and 43% agreement on argument spans (F_1 measure on exact match).

It is also worth noting that the NLP subcommunity dedicated to INFORMATION EXTRACTION (IE)—automatically extracting structured information from text—has been freer with what kinds of textual patterns they are willing to interpret as conveying relations. IE work has not explicitly linked these patterns to construction grammar or constructions. Indeed, often they are not even attempting to unpack the intended meaning of the text; they are simply looking for patterns that correlate with facts of interest. Nonetheless, such work represents an important precursor to the methods described in this thesis. It is discussed more thoroughly in §6.1.1.3 and §7.1.

2.4 "Surface construction labeling": what this thesis draws from CxG

As I argued in §1.5, it does not take a full "constructions all the way down" system to make constructions useful for NLP. Our approach, which we term "surface construction labeling" (SCL) , is to shortcut full constructional analysis by approximating it with patterns of words and conventional linguistic categories. Specific constructions of interest, such as those that express causal relations, can be codified as conventionalized combinations of surface forms, with the understanding that they should only be marked when accompanied by, say, particular syntactic configurations and part-of-speech tags. (We need not worry too much about when to count a counstruction, as we are not trying to be theoretically sound anyway; see §4.5.3.) Of course, the constructions can then be automatically tagged in the same SCL paradigm, using traditional NLP tools to approximate the underlying grammatical constructions.

On a theoretical and linguistic level, this approach draws extensively from CxG concepts. Most obviously, the very notion of the construction as the unit of analysis is a CxG innovation. We also adopt most of Goldberg's principles that define constructional approaches: of the three that apply to an individual linguistic analysis (1–3), we rely heavily on both principle 1 (that any construction, whether lexical or grammatical, can carry meaning) and principle 2 (that constructions link meaning directly to surface forms; principle 3, the assumption of a hierchical network of constructions, we drop for operational convenience, though we agree with it in theory). Further following CxG, our criteria for deciding when to call a pattern its own construction are guided by conventionality and frequent usage (see §4.3.5). Finally, we indirectly rely on CxG's assumed machinery for merging constructions into a final construct: some of our tagging guidelines assume that the meaning of the construction being annotated or parsed would need to be modified to account for interacting constructions like negation (see §4.1.1).

By design, however, our computational methods are **not** directly linked to existing CxG research. In keeping with the SCL approach, we do not implement a full construction parser, so the formalisms that have been developed are not directly applicable. Of previous computational approaches, ours has the most in common with Hwang and Palmer's work on CMCs.

Despite sidestepping some of the difficult questions about how to realize a full CxG parser, we believe that SCL work will move other kinds of CxG research forward, as well. Tools built on this approach will enable researchers to more easily explore large corpora to understand what phenomena a full construction grammar will need to explain (see §4.5.3 for examples). Eventually, systems will be able to relax the assumption of shallow semantic parsing that a specific lexical or morphological span carries each relation, which will bring non-lexicalized grammatical constructions within reach. And once constructions are firmly embedded as a lynchpin of semantic analysis, CxG techniques can more easily propagate back through the rest of the NLP pipeline. In the long term, then, SCL will serve as a stepping stone between current NLP systems and fully constructional tools.

Of course, we aim to use this approach to improve NLP in the short term, as well. In several crucial semantic domains, including causality, the constructions that carry meaning are difficult to represent and automatically tag using conventional notions of words and grammar. In such cases, simple lexical analysis fails to capture the bulk of the semantics of even unremarkable sentences (Fillmore et al., 2012). These constructions represent low-hanging fruit for incorporating CxG into NLP tools, while also laying the groundwork for further uptake. Accordingly, for the remainder of the thesis, I will focus on how SCL can be applied specifically to shallow semantic parsing for causal language.

Part I

Annotation and Data Representation

CHAPTER 3

Related Linguistic Analysis and Annotation Work

The interaction between causality and language has attracted attention from linguists both theoretical and applied/computational. On the theoretical side, the focus has been on explaining why certain causal assertions are seen as felicitous or infelicitous. On the applied side, researchers have taken two broad approaches to annotating texts for causality: either they have annotated spans whose referents have a cause-and-effect relationship in the real world, or they have annotated CAUSAL LANGUAGE—language used to intentionally appeal to psychological notions of cause and effect.

To provide background to our causal language annotation work, I will review each body of work in turn. I will also review previous efforts to annotate language for complex constructions of the sort we have seen are needed to capture causal language.

This section and the coming chapters assume familiarity with the notion and various measures of INTER-ANNOTATOR AGREEMENT. For more on these concepts and metrics, see §A.1.

3.1 Theoretical work on causation in language

The theoretical analysis at the foundation of most linguistic accounts of causality is Talmy's (1988) FORCE DYNAMICS model. Talmy, coming from the world of cognitive linguistics, notes a variety of linguistic constructions and distinctions that seem to rely on an intuitive psychological model of two entities applying forces to each other. The entity affected by the interaction, the PATIENT,[1] has an innate tendency toward either action (movement or change) or inaction (rest or stability). Some opposing entity, the AFFECTOR, counteracts the patient's intrinsic inclination with a strength less than or greater than that of the patient's force, and ultimately one of them triumphs.

Thus, to borrow some of Talmy's examples, *The ball's impact caused the lamp to topple* indicates the ball (the affector) overcoming the tendency of the lamp (the patient) toward inaction; in *The dome prevents flyballs from sailing out of the stadium*, the dome overcomes the balls' tendency toward action; *The plug was pulled, letting water flow out* indicates that the plug (the affector) ceased to overcome the water's tendency toward action; and *The stirring rod broke, letting the particles settle* indicates that the

[1]The terms "patient" and "affector" are actually drawn from Wolff et al. (2005). Talmy's original terms were AGONIST for the patient and ANTAGONIST for the patient.

	Patient tendency toward result	Affector-patient concordance	Occurrence of result
Cause	N	N	Y
Enable	Y	Y	Y
Prevent	Y	N	N

Table 3.1: Wolff et al.'s force dynamics characterization of Cause, Enable, and Prevent.

rod ceased to overcome the particles' tendency toward inaction. *Despite*, meanwhile, would denote a natural tendency of the patient that the affector fails to overcome. All psychological or non-physical causation is conceptualized by analogy to such physical force interactions. This model helps to explain why, e.g., *lightning causes fire* and *oxygen enables fire* sound reasonable, but *oxygen causes fire* does not.

Wolff et al. (2005) further develop the force dynamics model as a means of systematizing linguistic expressions of causality. They divide causal relationships into three categories—Cause, Enable, and Prevent—each exhibiting a different configuration of Talmy's force dynamic properties (see Table 3.1). They then arrange various kinds of causal linguistic expressions on a spectrum of specificity. On the least specific extreme, verbs like *affect* specify only the occurrence of some change, and can be used to describe any of the three force dynamic categories. Sentences with simple causal verbs (e.g., *lead to*, *depend on*) or causal conjunctives and prepositions (*because*, *consequently*, *since*) further specify that the end state was achieved, thus ruling out the Prevent interpretation. Periphrastic causatives like *cause to* or *prevent from*, which are typically matrix verbs that take an object and a clausal complement, add a specification of affector-patient concordance (e.g., *allow* specifies that the patient's tendency was toward the result encouraged by the affector). Each such verb thus fully selects for one of Cause, Enable, and Prevent. Lexical causatives like *kill* additionally imply direct causation, i.e., the absence of a mediating force on the causal relationship. Finally, resultative constructions such as *hammered the metal flat* incorporate the means of causation, as well.

In our work, we concern ourselves chiefly with the middle portion of this specificity spectrum; see §4.1.2. We also collapse the categories of Cause and Enable; see §4.1.4.2.

In a somewhat parallel track, Neeleman and Van de Koot (2012) and Hobbs (2005) both wonder about the huge constellation of causal factors underlying any given event (the CAUSAL COMPLEX, in Hobbs' terminology). For example, for a light fixture to turn on, not only must the switch be flipped, but the wiring must also be intact, the bulb must have been manufactured correctly, the power plant supplying the region must be operational, and so on. Few of these factors are typically labeled causes, suggesting a disparity between what philosophers or scientists might deem metaphysically or empirically causal and what speakers intuitively use causal language to describe. Hobbs concludes that parts of the causal complex merit the term "cause" when, given the current discourse context, they cannot be presumed a priori to be true. Similarly, Neeleman and Van de Koot suggest that a speaker chooses to spotlight one CRUCIAL CONTRIBUTING FACTOR (CCF) among all the contributing causal factors.

Since this line of work concerns what causal statements are made in the first place, it does not directly impact our annotations. However, the notion of the speaker profiling a single CCF does inform our choice to retain the agent as the Cause in examples like *I caused a commotion by shattering a glass*; see §4.1.5.2 below. The disparity between causality in the real world and causality in language also drives our decision to sidestep the morass of defining real-world causality for the purposes of annotation (see §4.1.1).

Much of the theoretical work on causation is concerned with lexical causatives like *kill* (i.e., cause to die). We exclude such relations from our annotations, as explained below (§4.1.2).

3.2 Annotation schemes and corpora covering real-world causation

Many previous projects have used the text as an anchor for recording causal relationships that exist in the real world (or at least in the imagined world of the text).

At least three groups have built small corpora of cause and effect pairs as training and/or evaluation data for NLP tasks. SemEval 2007 included a task (Girju et al., 2007) concerning classifying semantic relations between nominals, including causal relations. For example, in the sentence *A person infected with a flu virus strain develops antibodies against it*, *virus* and *flu* are labeled as a cause/effect pair. While it is true that the flu is caused by a virus, it was clearly not the speaker's intention to highlight that relationship in this sentence. This is presumably why the SemEval annotations do not indicate the linguistic triggers for causal relationships: real-world causal relationships, the focus of these annotations, may not be explicit in the text at all. Instead, the task relied on a common-sense notion of real-world causation.

Similarly, Do et al. (2011) had annotators mark causal event pairs in 25 news documents as evaluation data for an event causality identification system. They used the simple definition "the Cause event should temporally precede the Effect event, and the Effect event occurs because the Cause event occurs" (arguably circular, given that it defines causality as "causality plus temporal precedence"). They did not require any language indicating the speaker's intent to express the causal relationship. Marco and Surdeanu (2016) likewise annotated a small number of event pairs specifically in biomedical texts, marking "causal precedence" relations between mentions of biochemical events.

More recently, both the CaTeRS annotation scheme (Mostafazadeh et al., 2016) and the Richer Event Description schema (RED; Ikuta et al., 2014; Croft et al., 2016) have designed more thoroughly fleshed out annotations for real-world causal relations, which they integrated into unified annotation frameworks for temporal and causal relations. CaTeRS annotated a moderately sized corpus of short everyday stories with their scheme, while RED is still piloting integration of causality annotations (O'Gorman et al., 2016).

Each of these projects comes with two important limitations. First, each is focused on one particular class of causes and effects. In the SemEval case, the spans of interest are nouns; in the others, they are event annotations. In our experience, however, causes and effects vary freely between events, states, other kinds of clauses, and objects; none of these schemes captures the full range. (By design, Marco and Surdeanu also limit their annotations to the biomedical domain.)

Second, and more critically, all have found it remarkably difficult to achieve high inter-annotator agreement on real-world causal relations. This is consistent with Grivaz's (2010) finding that human annotators struggle to apply standard philosophical tests to make binary decisions about the presence of causation in a segment of text. She suggests alternative criteria, some of which we take into account in our coding manual (see especially §4.2). Ultimately, however, we concluded that focusing on real-world causation has unnecessarily confounded previous attempts at annotating causation in text. After all, what characterizes true causation has kept philosophers sparring for several millennia (Schaffer, 2014; Dowe, 2008); it seems unlikely that linguists will manage to resolve the disagreements simply by adding the element of textual annotation.

3.3 Annotation schemes and corpora covering causal language

Other projects have, to a greater or lesser extent, focused on annotating **stated** causal relationships, much as we have.

3.3.1 Schemes where causal relations are part of a larger repertoire

Three prominent, broad-coverage annotation schemes for computational semantics have included aspects of causal relations.

PropBank PropBank (Palmer et al., 2005) is one of the canonical schemes for shallow semantic parsing. It describes the core semantic arguments of many verbs, including some of causation (e.g., *cause* and *lead to*), allowing the arguments to be annotated consistently across different syntactic realizations. See Figure 3.1 for a sample verb annotation. Core verb arguments such as agent and patient are assigned verb-dependent, numbered role labels like Arg0 and Arg1; other adjuncts and modifiers are given ArgM labels indicating their function. Notably, the repertoire of ArgM categories include ArgM-PRP for purpose and ArgM-CAU for cause. The entire Penn Treebank corpus of syntactically parsed newswire text (Marcus et al., 1994) has been augmented with PropBank annotations.

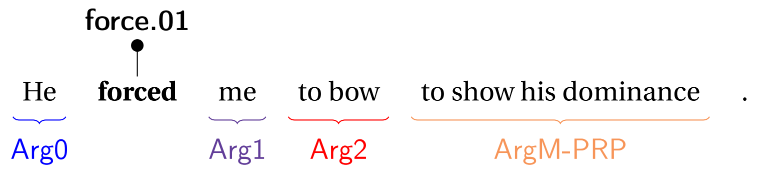


Figure 3.1: A sample PropBank annotation for the causal target verb *force*. ArgM-PRP indicates a purpose modifier.

FrameNet FrameNet (Ruppenhofer et al., 2016) has long been the gold standard for rich, domain-general shallow semantic parsing. It specifies a lexicon of semantic FRAMES, each characterizing some type of event, relation, or entity. Each frame entry includes a list of semantic roles corresponding to the participants in the frame, which are assigned frame-specific textual labels like Experiencer or Effect. The FrameNet project has annotated a large, diverse corpus with frame-evoking elements (i.e., lexical triggers of frames), as well as with the sentence spans corresponding to the participants in the evoked frames. An example appears in Figure 3.2. The lexicon includes a number of causation-related frames (e.g., Causation, Make_possible_to_do, and Preventing), as well as some Purpose and Explanation roles within frames.

While each incorporates elements of causality, neither FrameNet nor PropBank encodes the fact that the various manifestations of causality are semantically related—that all the relevant frames or roles describe a cause facilitating or inhibiting an effect. Different linguistic realizations of the same relation will be annotated very differently. This makes it difficult to identify cause-and-effect relationships in a unified manner. Additionally, as noted in §1.3, FrameNet does not have a natural way to represent as triggers constructions that are not MWEs or constituents.

The ASFALDA French FrameNet project recently proposed a reorganized frame hierarchy for causality, along with more complete coverage of French causal lexical units (Vieu et al., 2016). They merged some frames (e.g., Cause_to_start and Launch_process), added new ones (e.g., Attributing_cause), and adjusted some frame roles. They also added several frame-frame relations to better connect causality-related frames with each other and other frames. Some constructions would still be too complex to represent, since ASFALDA did not adjust the LU criteria that prevent FrameNet from incorporating non-contiguous constructions as triggers. Still, under their framework, many of our insights could likely be merged into mainline English FrameNet for a more coherent picture of causality.

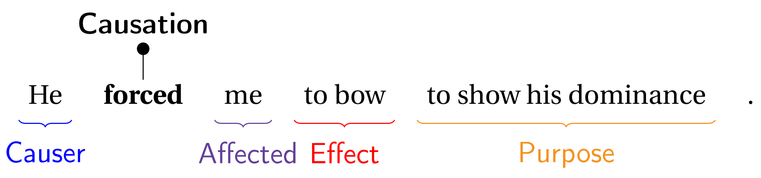


Figure 3.2: A sample FrameNet annotation for the causal frame-evoking element *force*.

Penn Discourse TreeBank The Penn Discourse TreeBank (PDTB; Prasad et al., 2008) is effectively a shallow semantic parsing scheme dedicated to discourse relations—relations between clauses and sentences. The PDTB offers a taxonomy of high-level relation classes—Temporal, Contingency, Comparison, and Expansion—each of which is subdivided into more specific subclasses. Explicit discourse connectives are tagged with these subclasses at whatever level of granularity annotators feel confident in. When there is no explicit connective between adjacent sentences but they still seem to be linked by one of the discourse relations, annotators mark an Implicit connective with the most relevant sense. Arguments are labeled simply as Arg1 and Arg2, depending on their syntactic positions; the proper interpretation of these labels depends on the relation. Annotators can mark relations from multiple classes, which inspired our annotation practices for relations that overlap with causality; see §4.2. Causality is included among the available subtypes under Contingency.

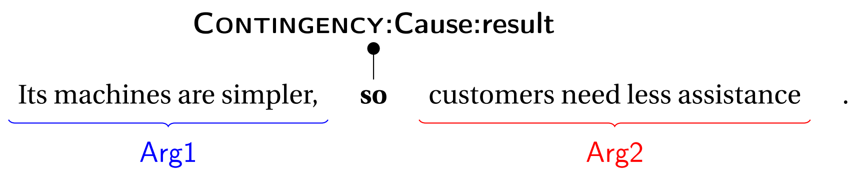


Figure 3.3: A sample Penn Discourse TreeBank annotation for the causal discourse connective *so*.

The PDTB thus treats causal relations as first-class objects of interest. Still, it does not represent any additional metadata distinguishing different kinds of causal relationships. More importantly, it restricts the linguistic triggers to conjunctions and adverbials, and thus excludes other realizations of the same relationships (e.g., causal verbs with arguments instead of discourse relations between clauses).

Indeed, as noted in §1.2, limitations on trigger types affect all three resources to some extent. PropBank covers exclusively verbs.[2] FrameNet's frame-evoking elements are much broader, covering verbs, prepositions, adverbs, conjunctions, and even some MWEs, but it still requires that the trigger be either a single word or a MWE that acts like a word. As we have seen, causal language demands a more flexible approach.

The PDTB does include a special category of annotation called AltLex, which is used for discourse relations that are not expressed by a conventional discourse connective. An AltLex can carry any of the usual PDTB relations. However, AltLex is not designed primarily to give a linguistic account of how the causal (or other discourse) relationship is expressed. Rather, it exists to remove the redundancy that an Implicit relation would introduce when there is already some language that implies the relation. For example, AltLex spans include *the most likely reason for this disparity* and *after that*, both of which would make an implicit causal annotation redundant. Such phrases are very unpredictable, including many words beyond the linguistic triggers. They are also restricted to relations between sentences.

[2]PropBank has recently begun to include some nominal and adjectival predicates that are derived from verbs, so that cases like *the **destruction** of the city* can be annotated; see Bonial et al. (2014).

Preposition schemes Just as verb-oriented annotation schemes have included some causal verbs, preposition-oriented annotation schemes have included some causal prepositions. Such schemes include The Preposition Project (TPP; Litkowski and Hargraves, 2005), Tratz's (2011) preposition inventory, and STREUSLE (Schneider et al., 2016). These schemes include Cause and sometimes Purpose as possible senses for prepositions like *from*, *for*, and *with*.

Rhetorical Structure Theory A fifth annotation paradigm that deserves some mention is rhetorical structure theory (RST; Mann and Thompson, 1988). Although in the discourse parsing community RST's highly structured analyses have faded in popularity compared to PDTB-style annotations, RST does include a rich vocabulary of causal relationships. Our division of causal relations into three types, Consequence, Motivation, and Purpose, closely parallels the Volitional, Non-volitional, and Purpose distinctions of the RST Discourse Treebank (Carlson et al., 2001).

Abstract Meaning Representation Abstract Meaning Representation (AMR; Banarescu et al., 2013) has been gaining traction as a semantic annotation framework, with several NLP shared tasks (e.g., May, 2016; May and Priyadarshi, 2017) in the past few years. Unlike shallow semantic annotation schemes, AMR does not tie its meaning representations to explicit spans in the sentence. However, it is not quite a purely conceptual representation, either, as are interlinguas or some "deep" semantic representations. AMR describes each sentence's meaning as a graph of labeled relations between nodes, where each node represents an entity, event, or state described or implied by some word(s) in the sentence. The scheme includes about 100 relations, among them :cause, :condition, and :purpose.

In general, AMR consistently maps causal closed-class words and grammatical constructions to either :cause or :condition. Annotators may even choose to mark as causal a relation that was expressed with, say, temporal language (e.g., *when*). Because the annotations are not linked directly to the text, AMR can also handle odd or idiomatic causal connectives such as *opens the way for*, at least in principle (though annotators may choose to record their meaning compositionally). Indeed, conventions have been developed for constructions concerning degrees, quantities, and comparisons, even when those constructions are not easily described in lexical terms (Bonial et al., in press).

However, lexical items that encode causation are represented in a more fragmented fashion. For example, verbs are each assigned a verb sense, as in PropBank, but there is no link between, say, cause-01 and induce-01. Nor are these verb senses, which are used to label graph nodes, associated in any way with the labels like :cause that decorate the graph edges. Thus, while AMR allows somewhat greater variety in causal connectives than existing shallow semantic annotation schemes, it does not (yet) offer a uniform treatment of causal phenomena.

3.3.2 Schemes focusing specifically on causal language

Closer to our work is the causality-specific scheme proposed by Mirza et al. (2014), who extend the TimeML scheme for temporal relations to incorporate causal relations. Their model is rich enough to capture a variety of linguistic triggers of causation, as well as cause and effect spans. It particularly attempts to follow Talmy's force dynamics model and Wolff et al.'s derived taxonomy of causal expressions (see §3.1). However, like the PDTB, it does not distinguish the different types of causal relationships. It also does not rigorously define what it counts as causal, and like RED and CaTeRS, it is limited to event-event relations. Finally, it assumes that triggers will be contiguous, leading to composite triggers like *the combined effect of* and *thanks in part to*, rather than treating these as instances of simpler connectives like *effect* and *thanks to*.

A small corpus of event pairs conjoined with *and* has been tagged as causal or not causal (Bethard et al., 2008), along with tagging temporal relations between the same events. They had annotators paraphrase sentences in causal or non-causal forms. Given how difficult it is to tell whether *and* is intended to convey causality, we do not annotate it, but Bethard et al.'s work gives a sense of what might be necessary to do so. Like Mirza et al.'s, this corpus focuses only on relations between events.

The project most similar in spirit to ours is BioCause (Mihăilă et al., 2013), which provides an annotation framework for causal relations in biomedical texts. The BioCause framework, like ours, marks the trigger and argument spans and the direction of causality. There are two primary differences between BioCause and our work. First, we focus exclusively on stated cause-and-effect relations, which make up few of the BioCause annotations; the bulk of that corpus describes claim-and-evidence relationships. Second, our work aims to be more general in scope. As such, our scheme also does not examine some kinds of domain-specific language that BioCause includes (e.g., *upregulation*). In a sense, our annotation approach may be thought of as a generalization of BioCause to broader domains, and also an attempt to pin down more precisely what kinds of relationships to annotate as causal.

3.4 Annotating based on construction grammar

Few projects have annotated data for constructions in a way that would be conducive to computational analysis and NLP. The closest attempt to ours is that of FrameNet, which has piloted an additional layer of constructional annotation. The FrameNet group has long recognized the many aspects of meaning that are not fully captured by an analysis of lexical triggers (Fillmore, 2008, 2006); indeed, FrameNet's initial emphasis on lexical units was purely a matter of operational design decisions. The group has begun an extensive project to document and annotate grammatical constructions in English, the FrameNet Constructicon (Fillmore et al., 2012), which serves as a companion repository of grammatical information for constructions that are not merely lexical. The project is still in development, and coverage remains sparse. Our work could be considered a demonstration of applying the approach to scalable full-text annotation. Our lexicography could also provide a basis for causality-related entries in the FrameNet Construction.

Similarly preliminary efforts are underway for VerbNet (Schuler, 2005), another verb lexicon that clusters verbs by their argument structure patterns. Bonial et al. (2011) looked to extend VerbNet to caused motion constructions (CMCs), and linked the changes in the verb resource to a small corpus annotated for CMCs. Bonial et al. (2014) likewise sought to expand PropBank to cover light verb constructions, "degree-consequence" constructions (as in *too political for my liking*), and others, though often without decomposing the semantics of the constructions. These efforts, too, are still in the pilot phase, not yet backed by sizable annotated corpora.

Finally, some variants of HPSG (see, e.g., Sag, 1997) explicitly represent constructions as feature structures, generalizing HPSG's representation of syntax to allow interleaved signs and construction daughters (similarly to SBCG; see §2.2). The Redwoods Treebank (Oepen et al., 2002) offers a corpus of HPSG trees produced by the English Resource Grammar (ERG; Flickinger, 2011) and hand-disambiguated by annotators, making it essentially an annotated corpus of construction-oriented HPSG parses. The variants of HPSG that ERG and Redwoods are based on generally focus on core grammatical constructions more than on covering idiosyncratic constructions of particular semantic relations such as causality.

3.5 Summary

While there have been several efforts to analyze the linguistics of causality, to annotate for causality, and to annotate arbitrary constructions, each leaves a gap that needs to be filled for semantic parsing of causal language. The theoretical analyses, while insightful, are not operationalized in a computation-friendly way. Existing annotations for causality either focus on real-world causality, which makes for low inter-coder agreement, or impose significant limitations on the form of the linguistic triggers or the semantic types of the arguments. Additionally, none cover the overlaps between causality and other relations other than temporal ones, and many—particularly FrameNet and PropBank—scatter semantic information related to causality throughout their lexica and semantic role repertoires. Finally, CxG-based annotations for more specific constructions than core grammatical ones have not yet been fully developed, nor have such annotations been applied to substantial full-text corpora. With the annotation scheme and corpus described in the next two chapters, we aim to cover all of these bases.

CHAPTER 4

The BECAUSE Annotation Scheme for Causal Language

The first step in our journey toward more flexible semantic parsing is to define the representation schemes we will use. These schemes will formalize how we wish to analyze causal language, and consequently what output we will expect a parser for these relations to produce.

As discussed in §1.3 and elaborated further in §3.3, existing representation and annotation schemes struggle to capture the full range of causal language, making it difficult to analyze and extract causal relationships in a coherent, comprehensive manner. This chapter presents the BECAUSE annotation scheme, which attempts to fill that gap.

Specifying such an annotation scheme requires grappling with two overarching issues:

- The **scope** of the annotations—the scheme must define the boundaries of the phenomenon to be annotated. This is particularly difficult for causation, a complex concept that has been heavily discussed in philosophical and psychological circles. Its boundaries are fuzzy: as noted in §3.1, causation is a psychological construct that we use to explain the world around us, and it does not perfectly match either empirical reality or the language we use to describe it (see §3.1 and Neeleman and Van de Koot, 2012). Furthermore, causation is intertwined with other relations, such as temporal order and conditionals, and other linguistic attributes, such as modality and negation. This raises important questions about how to carve out a meaningful piece of the semantic space for an annotation scheme to represent.
- The **annotatability** of the scheme—it must be possible for human annotators to apply the scheme consistently, and (ideally) without too much difficulty. This raises practical questions of how to reliably guide annotators to consistent decisions while still keeping the annotations as useful as possible.

There are also countless smaller edge cases to handle. For example, sometimes the relation of interest is negated, hedged (e.g., *I believe that...*), or conjoined with another assertion. Should *That might neither ease nor impede our efforts* be annotated at all? If so, how many separate constructs of causation does it contain, and do they need metadata indicating the negation and lack of certainty? For these cases and more, the scheme must specify whether and how to annotate.

In this chapter, I present four contributions toward coping with the complexity of annotating causal language:

1. Drawing on principles from Construction Grammar, I present BECAUSE (**B**ank of **E**ffects and **Cau**ses **S**tated **E**xplicitly), a **novel annotation scheme** for causal language (§4.1). The scheme provides a uniform representation for a wide spectrum of linguistic expressions, while still allowing for semantically relevant dimensions of variation. We attempt to limit the complexity of annotation by focusing not on the hairy metaphysics of causation, but on the claims about causation that are explicit in the text.
2. I describe additional layers of the annotation scheme that allow **multiple simultaneous meanings** to be assigned to an instance of a construction (§4.2). In addition to demonstrating that the approach generalizes beyond causality, this allows the scheme to encompass aspects of the semantic phenomena that intertwine with causality, and makes it easier to annotate boundary cases where causality is not the only relation present. Including this additional information adds only marginally to the difficulty of the annotation task.
3. I **compare two approaches to guiding annotators' decisions** about causal language, one using an annotation manual only and the other using an expert-compiled CONSTRUCTICON along with an annotation manual (§4.3). The constructicon-based methodology is similar to the two-stage methodology used in PropBank and FrameNet annotations: an initial phase of corpus lexicography produces a lexicon, and is then followed by a second phase in which annotators identify instances of the lexical frames in a corpus. In our case, the "lexicon" is a list of English constructions that conventionally express causality.
4. I discuss some **lessons learned** from our annotation efforts—the broader implications of our experience for difficult annotation tasks such as this one (§4.5).

This chapter incorporates material from Dunietz et al. (2015), which presented the first version of the annotation scheme and corpus (BECAUSE 1.0), and Dunietz et al. (2017b), which described the expanded corpus with extensions to overlapping relations (BECAUSE 2.0). It also draws from Dunietz et al. (2017c), which first introduced the "surface construction labeling" concept (under the name "constructions on top") and elaborated on its relationship with FrameNet.

4.1 Causality in the BECAUSE 2.0 annotation scheme

To improve upon the existing resources, we have designed an annotation scheme for causal language that borrows concepts from construction grammar. Unlike most previous schemes, ours allows arbitrary linguistic patterns—i.e., arbitrary constructions—to be marked as signaling causation. These constructions can be individual words of any part of speech, or they can be multi-word expressions or more complex patterns. The scheme also attempts to carefully define the scope of the annotations.

4.1.1 Annotation scheme design philosophy

As discussed above (§3.2), for the purposes of this project, we are **not** concerned with identifying relationships that are causal in some "true" metaphysical sense. Instead, we are concerned only with **what the text asserts**—causal **language** and what is meant by it. If and only if the text explicitly appeals to some psychological notion of promoting or hindering, then the relationship it asserts is one we want to represent, whether or not it is metaphysically accurate.

For example, the assertion *cancer causes smoking* states a false causation, but it would nonetheless be annotated. In contrast, *bacon pizza is delicious* would not be annotated, even though bacon may in fact cause pizza to be delicious,[1] because the causal relationship is not stated as such.

Although the boundaries of causality are not well-defined, we wished to study causal language in isolation to the extent we could. We therefore designed the annotation scheme to exclude language that expresses causality only as part of a larger relation (e.g., *destroy* implies causing a change of state, but that causality is embedded within the larger event of destruction). We likewise exclude language whose causal interpretation is vague or merely suggestive. However, we also designed the scheme to be composable with other components of semantic analysis: negation, aspect, hedging, and so on. We assume that other annotation schemes will represent these elements, and that this additional information may alter the semantics of the causal relationship as a whole.

In accordance with the SCL approach, we do not attempt to annotate all the underlying constructions that give rise to a particular sentence (morphological constructions, syntactic constructions, and so on). Instead, following the example of the FrameNet Constructicon, we annotate only the elements that contribute to the high-level construction that carries causal meaning. These elements may be drawn from multiple underlying constructions.

Our current focus is English only. We believe that the basic components of the annotation scheme should apply in other languages, but many adjustments would be needed; see the discussion of future work in §9.2.1.

4.1.2 Our operational definition of "causal language"

We use the term CAUSAL LANGUAGE to refer to clauses or phrases in which one event, state, action, or entity (the Cause) is **explicitly presented as** promoting or hindering another (the Effect). The Cause and Effect must be deliberately related by an explicit trigger, which we term the CAUSAL CONNECTIVE. As emphasized above, the words "presented as" are essential to this definition.

Causal relations can be expressed in English in many different ways. In this work, we exclude:

- **Causal relationships with no lexical trigger.** We do not annotate implicit causal relationships ("zero" discourse connectives). We expect our work to be compatible with other work on such relationships, such as the implicit relations in the PDTB and systems for recovering those relationships (e.g., Conrath et al., 2014). Similarly, we exclude resultative constructions not containing an explicitly causal verb.[2]
- **Connectives that lexicalize the means or the result of the causation.** For example, *kill* can be interpreted as *cause to die*, but it encodes the result, so we exclude it. This decision was made to allow the scheme to focus specifically on language that expresses causation. If lexical causatives were included, the vast majority of transitive verbs in the English language would have to be considered causal; it would be impossible to disentangle causation as a semantic phenomenon with its own linguistic realizations. It would also be impossible to annotate the Cause and Effect separately from the connective.[3]

[1]As a Jew who has never had bacon pizza, I maintain a strict agnosticism on this count.

[2]This has the unfortunate consequence that *John made his house red* would be annotated, but *John painted his house red* would not. Ultimately, of course, we would like to find ways to annotate even constructions where the form is a more abstract syntactic configuration, as is the case for resultatives, but for this work we consider such constructions out of scope.

[3]If lexical connectives are ever desired, the PropBank or FrameNet lexicon could be augmented to indicate which verb senses are causal, and the associated corpus could then act as a supplemental causal language corpus.

Omitting lexical causatives is consistent with previous causal language annotation schemes (e.g., Mirza et al., 2014), though we are not aware of previous attempts to define what must be lexicalized for a verb to be excluded.

- **Connectives that assert an unspecified causal relationship.** *Smoking is linked to cancer*, for example, specifies neither the direction of the causal link nor what sort of link is present, so we do not annotate it.

4.1.3 Anatomy of a causal language instance

For each instance of causal language that meets these criteria, we annotate up to four spans, any of which may be non-contiguous and which may overlap with each other:

- **The causal connective**[4]—the centerpiece of each instance of causal language, consisting of all words whose lemmas consistently appear as part of the causality-signaling construction. For example, the bolded words in ***enough*** *money* ***for*** *us* ***to*** *get by* would be marked as the connective. The connective is not synonymous with the causal construction; rather, it is a lexical proxy indicating the latter's presence.

  Following the principles of CxG, the connective may be any surface linguistic pattern conventionally used to indicate causation.

- **The Cause.** Causes are events, actors, or states of affairs, usually expressed as complete clauses or phrases (as opposed to a set of words that does not form a coherent grammatical unit).

- **The Effect.** Also generally an event or state of affairs, expressed as a complete clause or phrase.

- **The Means.** An action, also generally a complete clause or phrase. This argument exists to distinguish between the causing agent and the action by which that agent causes the Effect, both of which can be simultaneously present. For example, in *I caused a commotion by shattering a glass*, *shattering a glass* would be marked as the Means. See §4.1.5.2.

Either the Cause or the Effect may be absent, as in a passive or infinitive, though we do not mark an instance unless at least one is present. Means arguments are usually omitted; they are marked only when expressed as a *by* clause, a *via* clause, an absolute clause (e.g., *Singing loudly, she caused winces all down the street*), or a handful of other conventional devices.

If any of an instance's arguments consists of a bare pronoun, including a relative pronoun such as *which*, a coreference link is added back to its antecedent, assuming there is one in the same sentence. This allows systems that use the corpus to decide between using the pronouns or their antecedents as arguments, depending on whether they are more interested in the semantic or linguistic slot-fillers, and perhaps on whether they have access to a coreference resolver.

4.1.4 Additional metadata associated with each causal language instance

In principle, many semantic attributes are relevant to interpreting a causal assertion, including the time(s) at which it holds, its factivity, its epistemic status (including any attribution given in the text), and so on. For the most part, we assume that such attributes, if needed, will have to be covered by other

[4]The term "connective" is borrowed from the discourse literature, where it refers to a word or phrase expressing a discourse relation such as causality. It would perhaps have been clearer to retain the more generic semantic parsing term "trigger," but I use "connective" throughout the thesis to maintain consistency with our previous publications.

	Facilitate	Inhibit
Consequence	Let's give settlement systems THE **NECESSARY** CONTROLS **to** *manage the risks that they face*.	THE NEW REGULATIONS should **prevent** *future crises*.
Motivation	WE DON'T HAVE MUCH TIME, **so** *let's move quickly*.	THE COLD **kept** *me* **from** *going outside*.
Purpose	*He tied a complex knot* **so that** IT WOULDN'T COME UNDONE TOO EASILY.	(Not possible; see §4.1.4.2)

Table 4.1: Examples of every allowed combination of the three types of causal language and the two degrees of causation (with **connectives** in bold, CAUSES in blue small caps, and *Effects* in red italics). The bottom row is an example of epistemic causality, which we do not annotate in BECAUSE 2.0.

annotation schemes. We annotate just two pieces of metadat that are specific two causality: the type of causation and its polarity.

4.1.4.1 Types of causation

We distinguish three different types of causal relationships, each of which can have subtly different semantics. Examples of each are given in Table 4.1.

- Consequence instances assert that the Cause naturally leads to the Effect via some chain of events, without highlighting the conscious intervention of any agent. The majority of instances are Consequences (see Table 5.1).
- Motivation instances assert that some agent perceives the Cause, and therefore consciously thinks, feels, or chooses something. Again, what is important for this scheme is how the relationship is presented, so an instance is Motivation only if it frames the relationship in a way that highlights an agent's decision or thought.
- Purpose instances assert that an agent chooses the Effect out of a desire to make the contents of the Cause span true. What distinguishes Purposes from Motivations is whether the motivating argument is a fact about the world or an outcome the agent hopes to achieve.

The first version of the BECAUSE annotation scheme included a fourth category, Inference, for evidentiary uses of causal language like *She met him previously, because she recognized him yesterday*. Rather than asserting an actual chain of events from Cause to Effect, such instances borrow the language of Consequence to present the Cause as evidence or justification for the Effect (EPISTEMIC CAUSATION—"I believe this/you should believe this because. . . "). We eliminated this category because unlike other categories of causation, these instances do not attempt to describe a real-world causal relation (Hobbs [2005] calls them "washed-out causality...causality applied to the informational domain"), and unlike other overlapping relations, they never also express true causation; they constitute a different sense of *because*.

The decision tree used by annotators to determine the causation type is shown in Figure 4.1.

The structure of Purposes Note that there is a confusing duality in Purposes. For example, the desire for a particular outcome (e.g., *it wouldn't come undone too easily*) motivates, or causes, the Effect (*he tied a complex knot*). But from another perspective, tying a complex knot is a cause whose effect may be the knot resisting coming undone.

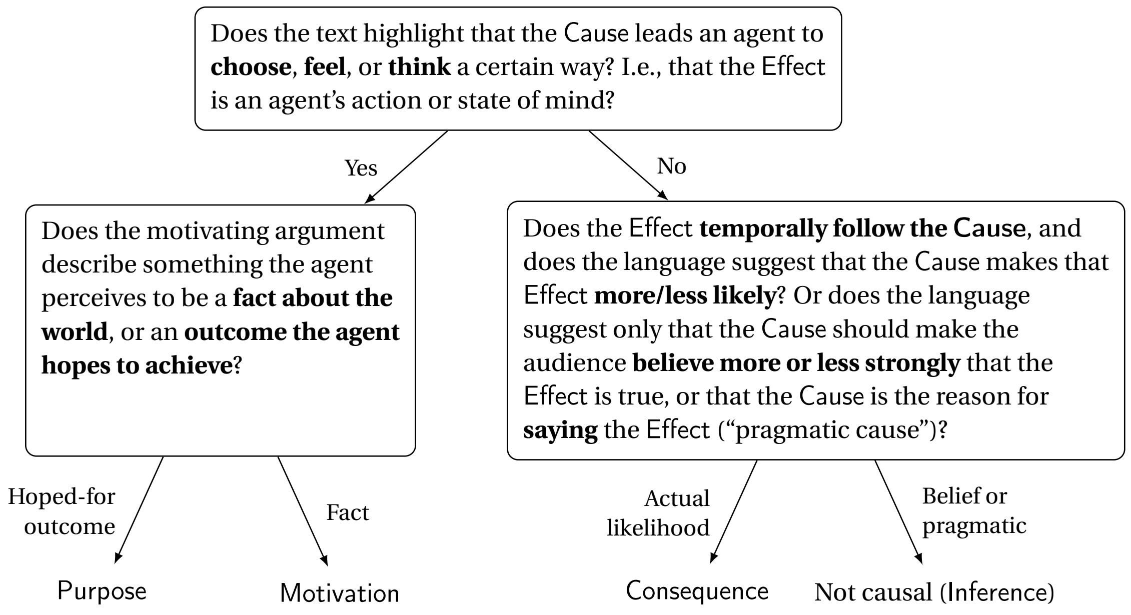

Does the text highlight that the Cause leads an agent to **choose**, **feel**, or **think** a certain way? I.e., that the Effect is an agent's action or state of mind?
Yes
No
Does the motivating argument describe something the agent perceives to be a **fact about the world**, or an **outcome the agent hopes to achieve**?
Does the Effect **temporally follow the Cause**, and does the language suggest that the Cause makes that Effect **more/less likely**? Or does the language suggest only that the Cause should make the audience **believe more or less strongly** that the Effect is true, or that the Cause is the reason for **saying** the Effect ("pragmatic cause")?
Hoped-for outcome
Fact
Actual likelihood
Belief or pragmatic
Purpose
Motivation
Consequence
Not causal (Inference)

Figure 4.1: The decision tree for causation types.

.

To understand the causal relationships here, we need a clearer picture of the structure of purposeful action. In a typical such case, an agent A desires some state S. Because of this desire, A takes some action X. Subsequently, if action X achieves its intended purpose, then S in fact obtains. Schematically, the structure is:

$$A \text{ desires } S \text{ and } A \text{ believes } X \text{ will cause } S \rightarrow A \text{ performs action } X \rightarrow S \text{ may obtain}$$

A statement may focus on either causal link in this chain. We treat Purpose language such as *in order to* as focusing on the first of these relationships; the second would be profiled by a statement like *we set clearer policies, which may make our company stronger.*

4.1.4.2 Degrees of causation

In principle, causal relationships lie on a spectrum from total prevention to total entailment. As noted above (§3.1), Wolff et al. (2005) discretize this spectrum into three categories: Cause, Enable, and Prevent. There is also arguably a mirror image of Enable, which could be termed Disentail, wherein the cause makes it possible for the effect **not** to hold (e.g., *Unless Congress acts, spending cuts will take effect automatically*, in which Congress' action makes it possible for spending cuts **not** to take effect).

In practice, however, we found in pilot annotations that annotators were able to reliably distinguish only positive and negative causation. We therefore label the degree of each instance simply as either Facilitate or Inhibit.[5]

The degree is part of the relation expressed by the connective; it does not take into account the contents of the arguments. For example, *her illness **caused** her not **to** go to the party* would be annotated as Facilitate, even though the Effect span is negated. As mentioned in §4.1.1, we expect that the

[5]In its current form, this attribute would more accurately be called "polarity." We retain the term "degree," however, with the expectation that we may wish to reintroduce finer-grained categories.

(14) The **cause** of the fire was a lit cigarette butt.
(15) Their support seems **essential to** the organization's continuity.
(16) She researches E. coli, the **cause** of many an infection.
(17) He pointed to his predecessor's mistakes as the **cause** of the current crisis.

Table 4.2: Examples of varied contexts for nominal and adjectival connectives.

.

interpretation of a causal relation may change once the contents of the argument span are unpacked and elements such as negation are made explicit.

This is why no Purpose instance can be labeled Inhibit, as noted in Table 4.1: all Purpose connectives express the motivation for the agent's decision, even if that decision is to not do something. We have not encountered any connectives that mean "a desire for outcome S kept agent A from performing action X"; expressing this meaning always seems to depend on negating one of the arguments.

4.1.5 Noteworthy difficult edge cases

Below are several edge cases that proved unusually difficult and for which our decisions significantly impacted the corpus. The full annotation manual, which includes annotation guidelines for many more edge cases, can be found in Appendix B (see in particular §B.4).

4.1.5.1 Connective spans for adjectival and nominal connectives

Connectives that are verbs, adverbs, conjunctions, prepositions, and complex constructions typically exhibit standalone, easily defined linguistic patterns linking Cause and Effect. Many connectives, however, are adjectives or nouns. These can be embedded within many different constructions, making it hard to determine the boundaries of the connective. As the examples in Table 4.2 demonstrate, such a connective can appear within the argument of a copula (14), a verb complement (15), an appositive (16), a prepositional argument (17), and more. While BECAUSE 1.0 would annotate copulas and prepositions as part of the connective (*cause of* __ *was* __ in 14 and __ *as cause of* __ in 17), this unnecessarily separates connectives by the context in which they appear.

For better parsimony and consistency, BECAUSE 2.0 marks only the noun or adjective and the function words used to introduce their arguments as part of the connective (*cause* and *essential to* in 14–17). We do not include words from matrix constructions that introduce the connective itself (e.g., the copula in 14). We also omit prepositions that are mandated not by the connective but by its context (e.g., the possessive *of* in *cause of*, which is absent in contexts such as *its cause was...*).

4.1.5.2 Agentive vs. non-agentive causes

When an Effect is caused by an agent taking an action—e.g., *I caused a commotion by shattering a glass*—either the agent (*I*) or the agent's action (*shattering a glass*) could plausibly be considered the Cause. Earlier versions of BECAUSE struggled with this distinction: the agent was taken to be metonymic for their action, so the agent was only annotated if the action was not explicitly present. However, given the scheme's focus on constructions, it seems odd to say that the Cause argument of the construction switches when a *by* clause is added.

Instead, for consistency, we now always label the agent as the Cause, but we add a Means argument for cases where but the action taken by that agent is also explicitly described. This includes cases where the agent has been syntactically removed, as in a passive or infinitive.

Another possibility would have been to divide causes into Cause and Agent arguments to distinguish between causing events and causing agents. Although FrameNet follows this route in some of its frames, we found this distinction difficult to make in practice. For example, a non-agentive cause might still be presented with a separate means clause, as in *inflammation triggers depression by altering immune responses*. In contrast, Means are relatively easy to identify when present, and tend to exhibit more consistent behavior with respect to which constructions introduce them.

4.1.5.3 Cases where part of the connective is conjoined

In multi-word connectives, as in ***enough** food **to** survive*, one or more connective words can sometimes be repeated if the connective is interrupted by a conjunction: ***enough** food **to** survive and even **to** thrive*. We annotate these cases as two separate causal instances sharing an argument. In the above example, *enough food* would be the Cause argument for two different *enough to* instances.

4.1.5.4 The connectives *to* and *for*

To and *for* might be the two most frustratingly overloaded prepositions in the English language. Not coincidentally, they are also extremely common. The combination necessitated some special guidelines for these two connectives.

To is marked as a Purpose connective whenever it can be rephrased as "in the hopes of" or "with the goal of." We initially tried "in order to" as a test, but that test also allows some non-Purpose usages, as in *I needed 10 points (in order) **to** win the competition*.

Occasionally, *to* is used in a way that might be synonymous with "in the hopes of," but is also arguably setting off a verb argument—e.g., *I used caulk **to** fix the leak*. It is difficult to reliably distinguish when a Purpose reading was intended—i.e., when the statement is intended to answer the question of why the agent took the action they did, rather than merely describing the action. However, the distinction has few implications for downstream processing, so we decided to annotate all such cases as Purpose.

There is also a Consequence usage of *to* that means "required for," as in *We don't have the votes **to** pass it*. The test for this usage is whether the *to* can be rephrased as "which is/are needed to."

For has even subtler variations in meaning. We decided on the following guidelines for the confusable cases:

- *For* is not annotated as causal at all when used to mean an exchange of goods (*buy **for** $5*, *swap mine **for** yours*), a topic (*my ideas **for** a better world*), or an item's *raison d'être* (*a vase **for** the flowers*, *a forward-facing camera **for** video chat*). This last category is somewhat close to Purpose, but it denotes the purpose of an item's existence rather than the purpose of an agent's action.
- When *for* indicates a precipitating action, as in *He thanked the crowd **for** listening* or *I'm reporting you **for** harassing me*, it is annotated as Motivation.
- When it describes a precipitating need or situation, the annotation depends on whether the need/situation is a pre-existing reality or a hoped-for outcome. In a case like *I go to the mall **for** the crowds*, where going does not cause the presence of crowds, the *for* is annotated as Motivation. But in *I went to the store **for** a bag of carrots*, where procuring carrots is the desired outcome, *for* is annotated as Purpose.

4.2 Annotating overlapping semantic relations

The constructions used to express causation overlap with many other semantic domains. For example, the *if/then* language of hypotheticals and the *so* ⟨*adjective*⟩ construction of extremity have become conventionalized ways of expressing causation, usually in addition to their other meanings. In BECAUSE 2.0, we annotate the presence of these overlapping relations, as well.

A connective is annotated as an instance of either causal language or a non-causal overlapping relation whenever it is used in a sense and construction that **can** carry causal meaning. The operational test for this is whether the word sense and linguistic structure allow it to be coerced into a causal interpretation, and the meaning is either causal or one of the relation types below.

Consider, for example, the connective *without*. It is annotated in cases like ***without** your support, the campaign will fail*. However, annotators ignored uses like *we left **without** saying goodbye*, because in this linguistic context, *without* cannot be coerced into a causal meaning. Likewise, we include *if* as a Hypothetical connective, but not *suppose that*, because the latter cannot indicate causality.

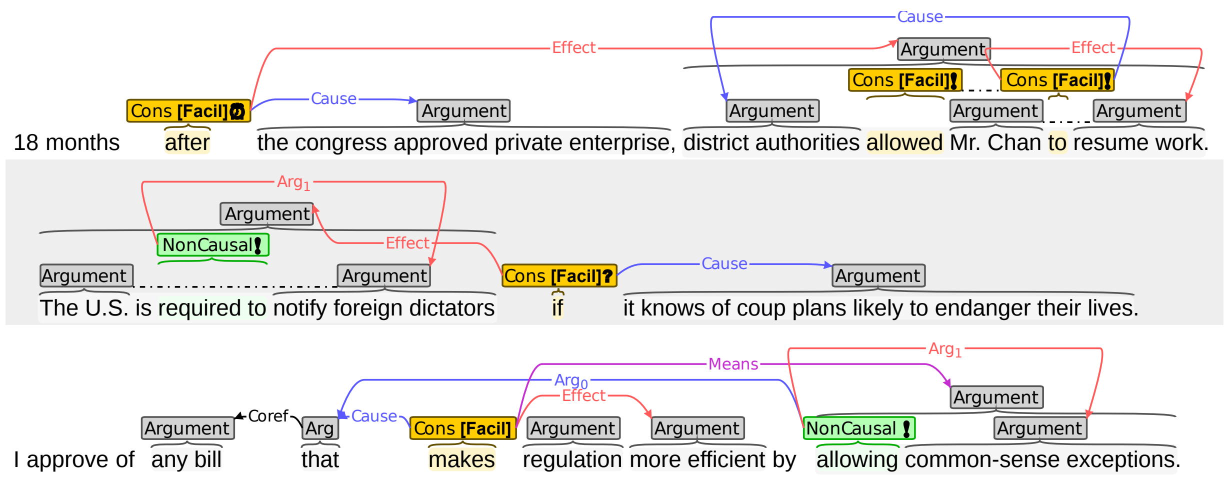
All overlapping relations are understood to hold between an ArgC and an ArgE. When annotating a causal instance, ArgC and ArgE refer to the Cause and Effect, respectively. When annotating a non-causal instance, ArgC and ArgE refer to the arguments that would be Cause and Effect if the instance were causal. For example, in a Temporal relation, ArgC would be the earlier argument and ArgE would be the later one.

The following overlapping relation types are annotated:

- Temporal: when the causal construction explicitly foregrounds a temporal order between two arguments (e.g., *once*, *after*) or simultaneity (e.g., *as*, *during*).
- Correlation: when the core meaning of the causal construction is that ArgC and ArgE vary together (e.g., *as*, *the more... the more...*).
- Hypothetical: when the causal construction explicitly imagines that a questionable premise is true, then establishes what would hold in the world where it is (e.g., *if... then...*).
- Obligation/permission: when ArgE is an agent's action, and ArgC is presented as some norm, rule, or entity with power that is requiring, permitting, or forbidding ArgE to be performed (e.g., *require* in the legal sense, *permit*).
- Creation/termination: when the construction frames the relationship as an entity or circumstance being brought into existence or terminated (e.g., *generate*, *eliminate*).
- Extremity/sufficiency: when the causal construction also expresses an extreme or (in)sufficient position of some value on a scale (e.g., *so... that...*, *sufficient... to...*).
- Context: when the construction clarifies the conditions under which the Effect occurs (e.g,. *with*, *without*, *when* in non-temporal uses). For instance, *With supplies running low, we didn't even make a fire that night.*

All relation types present in the instance are marked. For example, *so offensive that I left* would be annotated as both causal (Motivation) and Extremity/Sufficiency. When causality is not present in a use of a sometimes-causal construction, the instance is annotated as Non-causal, and the overlapping relations present are marked.

It can be difficult to determine when language that expresses one of these relationships was also intended to convey a causal relationship. Annotators use a variety of questions to assess an ambiguous instance, largely based on Grivaz (2010):



Effect
Cause
Argument
Effect
Cons [Facil]
Cons [Facil]
Cause
Argument
Argument
Argument
Argument
18 months after the congress approved private enterprise, district authorities allowed Mr. Chan to resume work.
Arg1
Argument
NonCausal
Effect
Argument
Argument
Cons [Facil]
Cause
Argument
The U.S. is required to notify foreign dictators if it knows of coup plans likely to endanger their lives.
Means
Arg1
Arg0
Effect
Argument
Coref
Cause
Argument
Arg
Cons [Facil]
Argument
Argument
NonCausal
Argument
I approve of any bill that makes regulation more efficient by allowing common-sense exceptions.

Figure 4.2: Several example sentences annotated for BECAUSE using the BRAT annotation tool (Stenetorp et al., 2012). The question mark indicates a hypothetical, the clock symbol indicates a temporal relation, and the thick exclamation point indicates obligation/permission.

- **The "why" test:** After reading the sentence, could a reader reasonably be expected to answer a "why" question about the potential Effect argument? If not, it is not causal.
- **The temporal order test:** Is the Cause asserted to precede the Effect? If not, it is not causal.
- **The counterfactuality test:** Would the Effect have been just as probable to occur or not occur had the Cause not happened? If so, it is not causal.
- **The ontological asymmetry test:** Could you just as easily claim the Cause and Effect are reversed? If so, it is not causal.
- **The linguistic test:** Can the sentence be rephrased as "It is because (of) X that Y" or "X causes Y?" If so, it is likely to be causal.

Figure 4.2 showcases several fully-annotated sentences that highlight the key features of the BECAUSE scheme, including examples of overlapping relations.

4.2.1 Lingering difficulties

Our approach to overlapping relations leaves open several questions about how to annotate semantic relations like causality that have blurred boundaries.

First, it does not eliminate the need for binary choices about whether a given relation is present; our annotators must still mark each instance as either indicating causation or not. Likewise for each of the overlapping relations. Yet some cases suggest overtones of causality or correlation, but are not prototypically causal or correlative. These cases still necessitate making a semi-abitrary call.

The ideal solution would somehow acknowledge the continuous nature of meaning—that an expression can indicate a relationship that is not causal, entirely causal, weakly or ambiguously causal, or anywhere in between. But it is hard to imagine how such a continuous representation could be annotated in practice.

Second, some edge cases remain a challenge for our new scheme. Most notably, we did not examine every semantic domain sharing some overlap with causality. Relations we did not address include:

- Origin/destination (as mentioned in §5.2; e.g., *the sparks **from** the fire*, ***toward** that goal*)
- Topic (see §4.5.4)
- Componential relationships (e.g., ***As part of** the building's liquidation, other major tenants will also vacate the premises*)
- Evidentiary basis (e.g., *We went to war **based on** bad intelligence*)
- Having a role (e.g., ***As** an American citizen, I do not want to see the President fail*)
- Placing in a position (e.g., *This move **puts** the American people **at** risk*)

These relations were omitted due to the time and effort it would have taken to determine whether and when to classify them as causal. We leave untangling their complexities for future work.

Other cases proved difficult because they seem to imply a causal relationship in each direction. The class of constructions indicating necessary preconditions was particularly troublesome. These constructions are typified by the sentence *(**For** us) **to** succeed, we all **have to** cooperate.* (Other variants use different language to express the modality of obligation, such as *require* or *necessary*.) On the one hand, the sentence indicates that cooperation enables success. On the other hand, it also suggests that the desire for success necessitates the cooperation.[6] We generally take the enablement relationship to be the primary meaning, but this is not an entirely satisfying account of the semantics.

4.3 Annotation process: the "constructicon" makes annotating for causality practical

4.3.1 Initial annotation process: coding manual only

In designing the first version of BECAUSE, before overlapping relations were added, we worked with three annotators to identify and decide on difficult cases. Once we felt the coding manual was ready for large-scale annotation, we spent several weeks training a previously uninvolved annotation expert to apply the scheme. My annotations on 201 sentences (containing 88 instances of causal language) were then compared against the new annotator's to determine inter-annotator agreement. The counts of different causation types are shown in Table 4.3.

Under this process, annotators were expected to consider all principles and special cases laid out in the manual for each decision: whether something counted as causal language at all, what words should be included in the connective, and what the argument spans should be. Decision trees (see Figure 4.1) were provided to determine the degree and type of the instance.

4.3.2 Initial annotations disappointed in both consistency and annotator effort/satisfaction

Our initial results (Table 4.4) did not seem to reflect our many iterations of feedback with the new annotator. For connectives that matched, the argument annotations agreed fairly well, as did the degrees.

[6]Necessary precondition constructions are thus similar to constructions of Purpose, such as *in order to*. As spelled out in §4.1.4.1, a Purpose connective contains a similar duality of causations in opposing directions: it indicates that a desire for an outcome causes an agent to act, and hints that the action may in fact produce the desired outcome. However, in Purpose instances, it is clearer which relationship is primary: the desired outcome may not obtain, whereas the agent is centainly acting on their motivation. In precondition constructions, both the precondition and the result are imagined, making it harder to tell which of the two causal relationships is primary.

	BECAUSE 1.0 IAA subcorpus annotated with:	
	Manual only	**Constructicon**
Consequence	66	33
Motivation	18	11
Purpose	4	21
Inference	0	4
Total	88	69

Table 4.3: Number of instances of each causation type in the two BECAUSE 1.0 subcorpora used for IAA. "Manual only" refers to the first round of IAA described in §4.3.2; "Constructicon" gives the counts for the IAA corpus used in §4.3.4. Counts are from my annotations. Documents were drawn randomly from the 2007 Washington section of the New York Times corpus.

	Partial overlap:	
	Allowed	**Excluded**
Connectives (F_1)	0.70	0.66
Degrees (κ)	0.87	0.87
Causation types (κ)	0.25	0.29
Argument spans (F_1)	0.94	0.83
Argument labels (κ)	0.92	0.94

Table 4.4: Inter-annotator agreement for the coding-manual-only approach, showing the middling degree of reliability achieved for connectives and causation types. κ indicates Cohen's kappa. Each κ score was calculated only for spans that agreed (e.g., degrees were only compared for matching connective spans). The difference between the two columns is that for the left column, we counted two annotation spans as a match if at least a quarter of the larger one overlapped with the smaller; for the right column, we required an exact match.

But the agreement rate for the connectives themselves was only moderately good, and agreement on causation types was abysmal.

Furthermore, the annotator, who had more than 30 years of annotation experience in other tasks, reported that she had found the process torturous and time-consuming, and that she did not feel confident in her choices. Even to achieve the results in Table 4.4, the annotator had to ask several clarification questions about specific constructions. This matched the experience of the earlier annotators who had helped us develop the scheme: they felt the guidelines made sense, and for any given annotation they could reach consensus via discussion, but even after working with the scheme for months, annotating still felt difficult and uncertain.

These results raised two important questions. The first was a matter of procedure: what could we do to improve the annotation process and reliability? The second question was more fundamental: even assuming we could improve the agreement scores, how should we interpret the fact that annotators were struggling so? If the scheme was still unintuitive after so much training, was it even meaningful at all?

The next few subsections address the first question. We return to the second in §4.5.2.

4.3.3 Modularizing the annotation process with corpus lexicography

The biggest factor dragging down annotators' comfort seemed to be the sheer number of decisions they had to make. In particular, we were expecting them to mentally redraw for every possible connective the

Connective pattern	⟨cause⟩ prevents ⟨effect⟩ from ⟨effect⟩
WordNet senses	`prevent.verb.01` `prevent.verb.02`
Annotatable words	prevent, from
Degree	Inhibit
Type restrictions	Not Purpose

Table 4.5: A sample entry in the constructicon.

fuzzy line between causal and non-causal, keeping in mind the entire gestalt of guidelines and special cases. It is no surprise that this task felt overwhelming, especially given that even once they had decided an instance was causal, they still faced decisions about annotation spans, causation type, and degree.

Much of this effort is in fact redundant. Most connectives in a text will be familiar, and the uses of any given connective are fairly consistent. Once a decision about a linguistic pattern has been made once, that decision can often be applied to future instances of the pattern.

Accordingly, we split the annotation process into two phases. In the first phase, we compiled a CONSTRUCTICON—a simple, human-readable list of known causal constructions—by manually cataloguing all connectives seen so far (including in the original annotation set). This catalog could then be quickly consulted whenever annotators encountered a potential connective. As exemplified in Table 4.5, the catalog gives the word senses that each connective pattern applies to, as well as possible variants, which words to include in the connective span, the degree the connective indicates, and in some cases restrictions on its causation type. In the latest version of BECAUSE, the constructicon also includes a list of possible overlapping semantic relations. Building the constructicon thus requires the same difficult decisions, but these decisions can be made once in consultation with others, and then applied repeatedly to new instances of each pattern.

In the second phase, annotators used the constructicon to label novel text. The task primarily now consisted of recognizing known patterns and making sure that the word senses used in the text matched the senses for which the patterns were defined.

Of course, there is a cycle in this process: if annotators spot a plausible connective that is not in the constructicon, they can propose it to be added. But given the relative rarity of novel connectives, this is not the annotators' primary task.

Some statistics on the BECAUSE 2.1 constructicon are given in §5.2, and the full constructicon can be found in Chapter C.

4.3.4 Constructicon-based annotation results: a significant improvement

Using this method, we trained another annotator for about a day. After just two rounds of annotation with feedback, the new annotator and I both used the constructicon to annotate a new dataset of 260 sentences, drawn from the same corpus, containing 69 instances of causal language.[7] The counts of causation types are again shown in Table 4.3.

We expected inter-annotator agreement to decrease compared to our previous attempt. The new annotator had far less annotation experience, and he had received a fraction of the training on this task.

[7]We did not reuse the same dataset because I had become too familiar with it and it had informed the constructicon, so it would not have been a meaningful test.

	Partial overlap:	
	Allowed	**Excluded**
Connectives (F_1)	0.78	0.70
Degrees (κ)	1.0	1.0
Causation types (κ)	0.82	0.80
Argument spans (F_1)	0.96	0.86
Argument labels (κ)	0.98	0.97

Table 4.6: Inter-annotator agreement results with annotators using the constructicon. See Table 4.4 for a fuller description of how these statistics were computed.

Additionally, we had fewer coded instances, which tends to cause κ scores to drop, and it seemed likely that the lower density of causal language would make it harder to spot the occasional instance.

In fact, our results (shown in Table 4.6) improved on our initial results in several important respects. First, there was a modest increase in F_1 for connectives. Second, agreement on causation types was now excellent. Third, all other metrics, even those that had already been high, improved slightly. And perhaps most significantly, these results were achieved with a fraction of the training time—a day instead of weeks—and the annotator found annotating quite painless.

Given that these results were computed on a different dataset, it is possible that the improvements are not as great as they seem. Nonetheless, the difference in annotator comfort was striking, and we believe that both datasets are representative.

Of course, the lexicography work itself still takes significant effort—effort that we were able to shortcut somewhat by mining our existing annotations to build the constructicon. But in general, the lexicography could be done in parallel with refining the scheme itself as trial datasets are annotated.

Thus, it is largely thanks to the constructicon that we were able to achieve high consistency in annotating causal language. All annotation efforts subsequent to this IAA test were performed using a constructicon.

4.3.5 Lexicography practices and decision-making

Our process for adding an entry to the constructicon typically started with annotators flagging a previously undiscussed construction, though occasionally constructions were added on the basis of language encountered in day-to-day life. The annotator would bring the example to my attention, at which point we would search through our corpus for more instances like it. We would also search the Corpus of Contemporary American English (COCA; Davies, 2008), a very large corpus of recent documents from a variety of genres. If we found more than a few clearly causal uses of the construction, and the causal reading felt smooth and natural to us as native speakers, we would include it in the constructicon.

When we ultimately recorded a new construction in the constructicon, we would also record other variants of the same construction we had seen, even if they had not yet shown up in our corpus. We would also compare any cases that did not appear to be causal to determine what tended to distinguish them, and record these distinctions to guide annotators' future decisions. In questionable cases, the annotators and I would discuss until we were satisfied that we had come up with a criterion for that construction that separated the causal examples we had found from the non-causal ones.

Lexicography frequently involves decisions about when a word or phrase is being used in a normative way and when it is being exploited to convey a meaning beyond its usual payload (see Hanks, 2013).

When annotating purely causal language, this question was rarely the locus of our difficulties in drawing lines. The tricky aspect was determining when a causal meaning was intended at all; when it was, it was usually fairly clear that causality was a conventional sense of that construction. To the extent that *brings to* is causal in sentences like *The initial confession* ***brought*** *others* ***to*** *follow suit*, any similar metaphorical use of *bring to*—and there are many—is equally so.

Where we did have some trouble with exploitation was in the context of overlapping relations. For example, *comes after* (in its non-physical sense) seems to be primarily temporal, but it also seems causal in uses like *The investigation* ***comes after*** *government watchdogs raised red flags.* But is it **conventionally** causal, or is the temporal construction merely being exploited to convey causality? Of course, there is no definitive line here, especially since constructions like this that were conventionalized began as unusual exploitations. But if we saw a pattern in the corpus of a construction being routinely used to convey causality, and their doing so did not strike us as a clever or unusual twist, we deemed it sufficiently conventionalized to include.

4.4 BECAUSE 2.0 inter-annotator agreement

For the second version of the annotation scheme, inter-annotator agreement was calculated between the two primary annotators on a sample of 8 documents from the year 2007 of the Washington section of the New York Times corpus (Sandhaus, 2008). The documents totaled 260 sentences, containing 98 causal instances and 82 instances of overlapping relations (per my annotations).

Statistics appear in Table 4.7. κ indicates Cohen's kappa, % indicates percent agreement of exact matches, and J indicates the average Jaccard index (see §A.1). Each κ and argument score was calculated only for instances with matching connectives.

The results show very good agreement on connectives: the F_1 score between annotators is 0.77. One factor in raising this score may be the inclusion of overlapping relations. Allowing annotators to mark multiple relations means fewer cases where they must arbitrarily decide whether a connective is primarily causal or primarily expressing a different relation. Under the first version of the annotation scheme, which excluded overlapping relations, the latest corpus contains 210 instances where such arbitrary choices would have been necessary.[8] Indeed, when we tested inter-annotator agreement using that scheme (Table 4.6), F_1 between similarly skilled annotators was 7 points lower. However, some of that difference may be due to chance differences in the IAA datasets, or to other clarifications in the annotation scheme that reduced ambiguity.

At 0.70, agreement on causal relation types is not perfect, but it is still high. Unsurprisingly, most of the disagreements are between Consequence and Motivation. Degrees are close to full agreement; the only disagreement appears to have been a careless error. Agreement on argument spans is likewise quite good.

For overlapping relations, only agreement on ArgEs is lower than for causal relations; all other metrics are significantly higher. The connective F_1 score of 0.89 is especially promising, given the apparent difficulty of deciding which uses of connectives like *with* or *when* could plausibly be coerced to a causal meaning.

[8]This is the number of instances that are annotated with both causal and overlapping relations and which would have been ambiguous under the earlier guidelines—i.e., the guidelines neither explictly excluded them nor deemed them always causal.

	Causal	**Overlapping**
Connective spans (F_1)	0.77	0.89
Relation types (κ)	0.70	0.91
Degrees (κ)	0.92	(n/a)
Cause/ARGC spans (%)	0.89	0.96
Cause/ARGC spans (J)	0.92	0.97
Cause/ARGC heads (%)	0.92	0.96
Effect/ARGE spans (%)	0.86	0.84
Effect/ARGE spans (J)	0.93	0.92
Effect/ARGE heads (%)	0.95	0.89

Table 4.7: Inter-annotator agreement for BECAUSE 2.0. % indicates percent accuracy. Argument statistics were computed only for connectives found by both annotators, as in §4.4 (so all metrics are independent of which annotator is considered the gold standard). The statistics in the second column count any overlapping relation annotation, whether or not it is also causal.
An argument's head was determined automatically by parsing the sentence with version 3.5.2 of the Stanford Parser (Klein and Manning, 2003) and taking the highest dependency node in the argument span.
Means arguments were not included in this evaluation, as they are quite rare—the IAA dataset included only two, one of which was missed by one annotator and the other of which was missed by both. Both annotators agreed with these two Means arguments once they were pointed out.

4.5 Lessons learned

4.5.1 When is lexicography appropriate?

The lexicography-based approach to semantic annotation, embodied by the constructicon, is of course not new. Several high-profile annotation projects have used it successfully, most notably PropBank and FrameNet. But it is a relatively uncommon approach for projects to take. Our experience suggests that although lexicography may not work well for every annotation effort, it may be more widely useful than current practice would indicate.

The essential question, then, is what characteristics make a project a good fit for corpus lexicography. Our experience here is limited, but one feature of our project seems to have made it particularly amenable to this approach: without a constructicon, annotators had to make the same decisions repeatedly. This was the core reason why the constructicon was useful; a constructicon would not save any work if it did not codify frequently made decisions. Crucially, repeated decisions not only were frequent, but constituted the vast majority of our annotators' decisions. If connectives had been highly unpredictable and a large fraction of annotators' decisions concerned whether to add a new constructicon entry, having codified previous decisions would have been much less useful. In fact, the burden of searching corpora to establish guidelines for when and whether to annotate a construction might have just slowed the whole process down, making annotating even more painful.

Of course, a lexicography-based approach intensifies concerns about meaningfulness. Adopting a lexicon may increase inter-annotator agreement, but what annotators are agreeing on is more constrained. A generous reading is that that the experts who compiled the lexicon have helped less-expert annotators make more accurate choices. But there is a less charitable reading, as well: if such constraints are needed for agreement, perhaps the annotation scheme fails to capture meaningful categories—perhaps it is merely a fiction of the minds of its designers. It is to this concern that I turn next.

4.5.2 What does low non-expert agreement say about validity?

What imparts validity to an annotation scheme is a fundamental question that haunts every annotation project. Even a well-thought-out scheme can include arbitrary, empirically meaningless decisions, which would seem to undermine the scheme's value as a description of a real linguistic phenomenon.[9]

This risk of arbitrariness is precisely what appears to bother Riezler (2014) in his discussion of circularity in computational linguistics: it is entirely possible that an annotation scheme has high inter-annotator agreement and can even be reproduced by software, and yet the scheme is empirically empty. The agreement can be achieved simply by developing a shared body of implicit, arbitrary theoretical assumptions among expert or intensively trained coders. Meanwhile, the fact that the annotations can be reproduced automatically shows only that the theory can be expressed both as an annotation scheme and as an annotation machine, not that it encapsulates something meaningful.

Thus, the problem of arbitrary assumptions raises especially serious questions about any scheme for which expertise seems to be required. If a scheme requires expert input or intensive training to reach agreement, that seems to suggest that the scheme is really a "stone soup" of theoretical, possibly arbitrary assumptions among the experts.

One tempting solution is Riezler's first suggestion for breaking circularity: using naïve coders, such as crowdsourced annotators. The instructions that convey the scheme to the coders, who do not share the same theoretical assumptions, constitute a second theory that the original theory can be grounded in. This, Riezler implies, would demonstrate the empirical reality of the theory behind the scheme, which he presumably would argue confirms its value.

For many schemes, high agreement among naïve coders may indeed break the circularity of the scheme. But as our lexicography-based approach highlights, this solution may not address the deeper problem of arbitrariness. Consider an annotation guide that relies on a lexicon to save the coders decisions. It is debatable whether this would qualify as a sufficiently different description of the theory to break circularity. Either way, though, if the original scheme was arbitrary, the arbitrariness still remains, even if naïve coders achieve high agreement. The arbitrary rules are no longer hidden in the heads of the annotators, but instead they are baked directly into the annotation guidelines as pre-made decisions. It seems, then, that the possibility of crowdsourcing (or, more generally, non-expert annotation) is not **sufficient** to make a scheme worthwhile.

Some (though notably not Riezler) have argued that disagreement among naïve coders demonstrates the empirical emptiness of a scheme—i.e., that the possibility of crowdsourcing is still a **necessary** condition for a scheme's validity. (The concerns we raised above suggest this argument, as well.) This argument is also problematic, because it assumes that naïve coders' explicit knowledge accurately reflects how their language works. That may seem reasonable—after all, naïve coders are competent users of the language. But in practice, there is no reason to expect the average person to have meta-linguistic awareness, any more than one would expect a baseball player—a competent user of physics—to correctly identify the physics phenomena at work when he swings. The fact that expertise is required to precisely describe a phenomenon does not mean that the phenomenon is not empirically real.

If, consequently, agreement among naïve coders is neither necessary nor sufficient to ascribe value to an annotation scheme, how do we proceed?

One way out is Riezler's second proposal: extrinsic task-based evaluation. If an annotation scheme is useful for a particular downstream NLP task—e.g., information extraction—then in some sense it is irrelevent whether the scheme is arbitrary; it at least correlates with the truth enough to be practically useful. We hope our scheme for causal language will fall into this category by proving useful, both directly to humans seeking causal information and for downstream information extraction.

[9] Similar questions arise in designing and assessing tests for social science research (Trochim, 2006).

(18)	***too*** *sweet* ***to*** *eat*
(19)	***too*** *sweet* ***for*** *me* ***to*** *eat*
(20)	*sweet* ***enough to*** *eat*
(21)	*sweet* ***enough for*** *me* ***to*** *eat*
(22)	*sweet* ***enough that*** *I can eat it*
(23)	***so*** *sweet* ***that*** *I can't eat it*
(24)	***so*** *sweet I can't eat it*

Table 4.8: Examples of causal extremity constructions.

Another way out is a type of usefulness that Riezler does not discuss. Often, simply attempting to formalize a phenomenon yields insights into some aspect of language, even if the formalization is empirically questionable.

It could be, for example, that our causal language scheme invents empirically meaningless semantic categories. However, it may still suggest hypotheses about how people use certain causal constructions. For instance, how often people talk about inhibiting vs. facilitating may vary dramatically depending on the genre. If validated, such an observation would yield valuable insights about language use and perhaps psychology—insights we would not have even thought to look for without the annotation scheme.

In short, then, we do not believe that low agreement among naïve coders (or a need for expert guidance in decision-making, such as a lexicon) necessarily impugns the value of an annotation scheme as a whole. Accordingly, we hope that our suggestion of construction-based lexicography will help others build annotation schemes and corpora that are valuable by the criteria I have outlined. In our own future work, we hope to demonstrate that our causal language scheme meets these criteria, as well.

4.5.3 How should constructions be individuated in CxG-based annotation?

Computational formalizations of CxG typically try to spell out a list of linguistic forms and inheritance relationships. Conceptually, this is similar to building a lexicon of words, except that the units are more structured, with arguments, semantic mappings, and constraints.

This approach does eliminate the problematic abstraction barriers that conventional NLP systems must contend with. However, building a constructicon for NLP use raises an equally serious problem of what to include as a construction.

As an example, consider the examples in Table 4.8 of constructions where the extremity of a graded attribute leads to some result. (Variants of all of these were found in the corpus.) How many different constructional patterns are in play here? One option is simply to say that each example employs a separate construction. However, this approach is extremely unparsimonious: (20), (21), and (22) share an obvious *enough ⟨complement⟩* structure; (20) is almost identical to (21), but with an optional argument removed, and likewise for (18) and (19); (24) is identical to (23) but without the optional complementizer; and the *to* tokens in (18), (20), and (21) all seem to perform identical functions.

The usual CxG answer is to construct a hierarchy of inheritance relationships that capture just the right generalizations. Such hierarchies have indeed been carefully constructed in other domains (e.g., Hasegawa et al., 2010). At some point, though, a product of multiple interacting constructions starts to become conventionalized, enabling the combined form to take on unique, non-compositional properties. How conventionalized must such a combination be before it is considered its own construction?

Does *as a result* appear as a collocation often enough that it deserves its own constructicon entry, even though more compositional variants such as *as one result* occasionally show up?

In theory, the difficulty of individuating constructions should be particularly problematic for SCL. After all, its entire *raison d'être* is to sidestep a full-fledged analysis of all the underlying constructions—but without the complete analysis, it is impossible to determine where one construction ends and another begins, or where multiple constructions are interacting. This approach avoids the work of the analysis, but it also cannot reap the benefits.

In practice, this issue is not too damning for SCL; the takeaway here for NLP constructicon developers is simply that the decisions must be made. The lines between constructions may be drawn semi-arbitrarily, but as long as they mostly match the right set of surface patterns, it should not matter whether a pattern is considered one construction or two, the implications for the theory notwithstanding. In fact, the lack of precise definition may even be a benefit: decisions can be made more on the grounds of convenience for machine learning algorithms than rigorous consistency and elegance.

This "quick-and-dirty" style of analysis also allows researchers to discover observations like the patterns in (18)–(24). These are the very observations that a fuller construction grammar will ultimately need to explain. The discovery of such organized observations is precisely the hope that Fillmore expressed in his introduction to the FrameNet Constructicon (Fillmore et al., 2012).

4.5.4 How should overlapping semantic relations be annotated?

Our experience suggests several lessons about annotating multiple overlapping relations. First, it indicates that a secondary meaning can be evoked without losing any of the original meaning. In terms of the model of prototypes and radial categories (Lewandowska-Tomaszczyk, 2007), the conventional model for blurriness between categories, an instance can simultaneously be prototypical for one type of relation and radial for another. For instance, *the ruling* ***allows*** *the police* ***to*** *enter your home* is a prototypical example of a permission relationship. However, it is also a radial example of enablement (a form of causation): prototypical enablement involves a physical barrier being removed, whereas *allow* indicates the removal of a normative barrier.

A second lesson: even when including overlapping semantic domains in an annotation project, it may still be necessary to declare some overlapping domains out of scope. In particular, some adjacent domains will have their own overlaps with meanings that are far afield from the target domain. It would be impractical to simply pull all of these second-order domains into the annotation scheme; the project would quickly grow to encompass the entire language. If possible, the best solution is to dissect the overlapping domain into a more detailed typology, and only include the parts that directly overlap with the target domain. If this is not doable, the domain may need to be excluded altogether.

For example, we attempted to introduce a Topic relation type to cover cases like *The President is fuming* ***over*** *recent media reports* or *They're angry* ***about*** *the equipment we broke* (both of which have the flavor of Motivation), where one argument is about and possibly caused by another. Unfortunately, opening up the entire domain of topic relations turned out to be too broad and confusing. For example, it is hard to tell which of the following are even describing the same kind of topic relationship, never mind which ones can also be causal: *fought* ***over*** *his bad behavior* (behavior caused fighting); *fought* ***over*** *a teddy bear* (fought for physical control); *worried* ***about*** *being late*; *worried* ***that*** *I might be late*; *skeptical* ***regarding*** *the code's robustness.* We ultimately determined that teasing apart this domain would have to be out of scope for this work.

For a discussion of the patterns in overlapping relation annotations in our corpus, see §5.2.1 (particularly Table 5.1).

(25) We headed out **in spite of** the bad weather.
(26) We value any contribution, **no matter** its size.
(27) Strange **as** it seems, there's been a run of crazy dreams!

(28) You're **as** bad **as** my mom!
(29) **More** boys wanted to play **than** girls.
(30) Andrew is **as** annoying **as** he **is** useless.
(31) I'm poor**er than** I'd like.

Table 4.9: Examples of concessive and comparative language, with the tokens participating in the relevant construction bolded.

4.6 Contributions and future work

This chapter has described the BECAUSE annotation scheme and corpus, the most comprehensive, flexible, and cohesive attempt to date at representing the causal relations expressed by a text. Using the "surface construction labeling" approach to applying CxG, the scheme handles a wide variety of realizations of causal relations. Along the way, I have empirically demonstrated the importance of constructicon-building for annotating in this style.

Despite the need for further investigation of the issues detailed in §4.2.1, our attempt at extending causal language annotations to adjacent semantic domains was also largely a success. We have demonstrated that it is practical and sometimes helpful to annotate all linguistic expressions denoting a semantic relationship, even when they overlap with other semantic relations. We were able to achieve high inter-annotator agreement even—perhaps especially—with these overlaps.

In the future, it would be interesting to see if annotators could reach better agreement on detailed degrees of causation using a hierarchy. Facilitate could be split into Cause and Enable, retaining the higher-level classification as an annotation option. Then, much as the PDTB allows annotators to choose the most specific level of the relation hierarchy that they can confidently distinguish, annotators could choose between Cause and Enable, or they could simply select Facilitate if the construction is ambiguous.

Beyond the domain of causality, we would like to apply a similar annotation methodology to other semantic relations. Other fundamental relation types exhibit similarly constructional behavior; concepts and components of meaning that are central to human thinking are squeezed into language ubiquitously and at all levels of linguistic structure. Concessives and comparatives are two examples of other areas where complex constructions abound, as shown in Table 4.9, and doubtless there are many more. (For more on comparative constructions, see Hasegawa et al. [2010]; for more on concessives, see Barth [2000] and Vergaro [2008].) Annotating such relations will likely reveal some weaknesses of the annotation approach, necessitating additional research on refinements and extensions. We also hope to extend our approach to other kinds of constructions (particularly morphological and order-based ones), which would allow the approach to be applied to other languages; see §9.2.1.

For now, however, I proceed to describing the corpus annotated with the BECAUSE scheme.

CHAPTER 5

The BECAUSE Corpus of Causal Language

On its own, an annotation scheme like BECAUSE is not terribly useful to either linguists or tool-builders. Only once a large amount of data has been annotated with it can linguists can begin to look for empirical patterns in language usage, and can engineers train and evaluate NLP systems that automatically produce annotations for downstream use.

To that end,[1] in this chapter I present two contributions:

1. I describe the **annotated corpus** we have built for causal language (§5.1).
2. I analyze the corpus to derive **insights about causal language** and its interactions with its semantic neighbors (§5.2). This analysis is an initial pass at identifying some of the empirical patterns that the corpus has to offer.

Like the previous one, this chapter draws from Dunietz et al. (2015), Dunietz et al. (2017b), and Dunietz et al. (2017c).

5.1 The fully-annotated BECAUSE corpus

5.1.1 Rationale for document selection

We had several goals in mind when selecting documents for the BECAUSE corpus:

1. We sought texts with a reasonably high concentration of causal language. This led us to choose specifically the Washington section of the New York Times (NYT) corpus (Sandhaus, 2008): in initial pilots, these documents seemed denser in causal language than sections like world news, science, or opinion. As a rule, more literary or lyrical writing tended to be less dense in causal language, preferring to express causality implicitly or with lexical causatives (which is in itself an intriguing observation).
2. We wished to annotate some documents that had already been annotated with other schemes. Hand-annotated syntactic parses were of particular interest, since they allow testing causal language taggers for the impacts of parse errors by comparing results with and without gold-standard parses. This led us to choose a selection of Wall Street Journal (WSJ) documents from

[1]Arguably this should read "to those ends," but *to that end* appears to be an insufficiently compositional Purpose connective to allow such a phrasing.

the Penn Treebank (Marcus et al., 1994). We also selected some documents from the Manually Annotated Sub-Corpus (MASC; Ide et al., 2010) with the hope that that corpus' many layers of annotations, semantic and otherwise, will allow studying the interactions between causal language and other kinds of linguistic phenomena.

3. We aimed for at least some diversity in the topics covered. Our documents are skewed toward politics (particularly given our NYT selection), but they also contain many articles on economics, health, business, and more.

4. Likewise, we aimed for some diversity in the types of language. Our sample is again somewhat skewed toward newswire (NYT, Wall Street Journal, etc.), but we included several Slate pieces from the MASC corpus, which provide a different style of writing. We also included several transcripts of oral Congressional hearings for a sample of spoken language, which seems to be richer in causal expressions.

5.1.2 Documents in the corpus

The BECAUSE 2.1 corpus[2] consists of the following exhaustively annotated documents, which total 4867 sentences and 123,674 tokens:

- 59 randomly selected articles, totaling 1924 sentences, from the year 2007 in the Washington section of the NYT corpus
- 47 documents, totaling 1542 sentences, randomly selected[3] from sections 2–23 of the Penn Treebank
- 772 sentences[4] transcribed from Congress' Dodd-Frank hearings, taken from the NLP Unshared Task in PoliInformatics 2014 (Smith et al., 2014)
- From MASC, 10 newspaper documents (Wall Street Journal [WSJ] and NYT articles, totalling 547 sentences) and 2 more casually written Slate articles (82 sentences, drawn from the "Journal" genre)

All of these documents were annotated either by me or by an experienced, trained annotator who worked with us on the development of the second version of the annotation scheme. At the annotator's request, he was not provided with the annotations from BECAUSE 1.0 (the smaller dataset that excluded overlapping relations). I checked over the vast majority of the new annotations, frequently comparing them to BECAUSE 1.0 to help spot errors.

All annotations were performed using BRAT (Stenetorp et al., 2012),[5] a web-based annotation tool (see Figure 4.2).

[2]Publicly available at `https://github.com/duncanka/BECAUSE`. Version 2.1 includes some minor corrections to errors in the 2.0 release, which were spotted in the process of computing the statistics for this chapter.

[3]We excluded WSJ documents that were either earnings reports or corporate leadership/structure announcements, as both tended to be merely short lists of names/numbers.

[4]The remainder of the document was not annotated due to constraints on available annotation effort.

[5]`http://brat.nlplab.org`. Minor modifications were made to BRAT to facilitate annotating and searching. The added search features included searching by attribute and recursive search within a directory. The changes affecting annotation were bug fixes for tab characters within annotations and checks to ensure that all and only those attributes applicable to a given span type can be annotated for that span. See `https://github.com/duncanka/brat/tree/all-merged` for the modified code.

5.1.3 Examples of interesting types of constructions

Many examples so far have been relatively straightforward. Below are a variety of more interesting samples from the corpus that demonstrate the range of annotations (lightly modified to allow abbreviating the sentential contexts). Connectives are shown bolded, CAUSES/ARGCS in blue small caps, *Effects/ArgEs* in red italics, and `Means` in purple typewriter text. For arguments with coreference links, the antecedent is marked in brackets matching the argument's formatting.

(32) Multi-word expressions:

a) *Almost all of that* is **attributable to** THE INCREASE IN EDUCATION AID. [Consequence/Facilitate]
b) *An airline might sacrifice costly safety measures* **in order to** REPAY DEBT. [Motivation/Facilitate]
c) THE 23-5 VOTE **clears the way for** *consideration on the House floor next week*. [Consequence/Facilitate]

(33) Gappy connectives:

a) SONOGRAMS **allow** *doctors* **to** *estimate the fetus's weight*. [Consequence/Facilitate; **not** Obligation/permission]
b) *The participants appeared to have been chosen* **on** the **grounds that** THEY PROVIDED A BULWARK AGAINST EXTREMIST VIEWS. [Motivation/Facilitate]
c) No government's cooperation will be as **necessary** as MEXICO'S **if** *immigration reform* **is to** *succeed*. [Consequence/Facilitate. Accounting for ellipsis would require post-processing.]

(34) Overlaps between connectives and their arguments:

a) There are not **ENOUGH** POLICE **to** *satisfy small businesses*. [Consequence/Facilitate, +Extremity/sufficiency]
b) The report said it was **TOO** EARLY **to** *say whether the security effort in Baghdad would achieve lasting security gains*. [Consequence/Inhibit, +Extremity/sufficiency]
c) *They are buying small numbers of machines* **for** EVALUATION **PURPOSES**. [Motivation/Facilitate]

(35) Single-argument instances:

a) We recommended more closely linking the regulatory structure to the **reasons why** *we regulate*. [Motivation/Facilitate; Cause unspecified]
b) The Bakersfield Supermarket went out of business last May. The **reason** was not HIGH INTEREST RATES OR LABOR COSTS. [Consequence/Facilitate; Effect in previous sentence. Negation would require post-processing.]
c) Are there people who want to stay in their homes but are being **forced into** *foreclosure*? [Motivation/Facilitate; Cause passivized out]

(36) Instances with Means:

a) THE FED `has injected reserves into the banking system` in an effort to calm the markets and **avert** *a repeat of 1987's stock market debacle*. [Consequence/Inhibit; *in effort to* is its own connective, as well.]
b) THIS `“monetized” the coin`, **making** *it legal tender*. [Consequence/Facilitate]
c) WE must `establish a modern regulatory structure` to **discourage** *unsound practices and conduct*. [Motivation/Inhibit; *to* is its own connective, as well.]

(37) Various uses of *to*:

a) *Torricelli took the Senate floor* **to** APOLOGIZE TO THE PEOPLE OF NEW JERSEY. [Purpose/Facilitate]
b) Democrats would have THE STRENGTH **to** *override a veto*. [Consequence/Facilitate, where *to* means "needed to"]

(38) Various uses of *for*:

a) *Further insight about speculation would be of tremendous help*, **for** IT IS TRUE THAT SPECULATORS BEAR BLAME. [conjunctive *for* as Consequence/Facilitate]
b) *Human rights groups criticize Mahathir* **for** ARRESTING AND JAILING SCORES OF SUSPECTED MILITANTS. [prepositional *for* as Motivation/Facilitate]
c) *The Journal editorial page gets one point* **for** FAILING TO NOTE THE PARDON IN ITS INITIAL OP-ED. [prepositional *for* as Consequence/Facilitate]
d) *Powell touched down in Brunei* **for** TWO DAYS OF TALKS WITH MEMBERS OF THE ASSOCIATION OF SOUTH EAST ASIAN NATIONS. [prepositional *for* as Purpose/Facilitate]

(39) Causal instances with overlapping relations:

a) **Within** minutes **after** THE COMMITTEE RELEASED ITS LETTER, *Torricelli took the Senate floor to apologize to the people of New Jersey*. [Motivation/Facilitate, +Temporal]
b) *Auburn football players are reminded of last year's losses* **every time** THEY GO INTO THE WEIGHT ROOM. [Consequence/Facilitate, +Correlation]
c) Previously, *he allowed increases in emissions* **as long as** THEY DID NOT EXCEED THE RATE OF ECONOMIC GROWTH. [Motivation/Facilitate, +Hypothetical]
d) [HE WILL ROLL BACK A PROVISION KNOWN AS NEW SOURCE REVIEW] THAT **compels** *utilities* **to** *install modern pollution controls whenever they significantly upgrade older plants*. [Consequence/Facilitate, +Obligation/permission. *Whenever* is its own connective, as well.]
e) MANY EXPECTED SYNERGIES OF FINANCIAL SERVICE ACTIVITIES **gave rise to** *conflicts and excessive risk taking*. [Consequence/Facilitate, +Creation/termination]
f) **With** HAMAS CONTROLLING GAZA, *it was not clear that Mr. Abbas had the power to carry out his decrees*. [Consequence/Facilitate, +Context]

(40) Non-causal overlapping relations:

a) *I am going to have to restrain myself and others from asking any questions* **after** THE 5 MINUTES. [Non-causal, +Temporal]
b) **Amid** ALL THE COURTROOM CHARACTERIZATIONS OF HIM, *Colonel Chessani sat stone-faced at the defense table in his desert fatigues, jotting notes*. [Non-causal, +Context]
c) These steps include enhancing the [EXECUTIVE ORDER WHICH] **created** *the President's Working Group on Financial Markets*. [Non-causal, +Creation/termination]

5.2 Corpus statistics and analysis

The corpus contains a total of 4867 sentences, among which are 1803 labeled instances of causal language, meaning about three sentences out of every eight contain causal expressions, on average. 1634 of these instances, or 90.7%, include both Cause and Effect arguments. 587—about a third—involve overlapping relations. The corpus also includes 584 non-causal overlapping relation annotations. The frequency of both causal and overlapping relation types is shown in Table 5.1.

Based on this corpus and the lexicography practices described in §4.3.5, the constructicon currently

	Consequence	Motivation	Purpose	**All causal (subtotal)**	Non-causal	**Total**
None	625	319	272	1216	–	1216
Temporal	120	135	–	255	464	719
Correlation	9	3	–	12	5	17
Hypothetical	73	48	–	121	24	145
Obligation/permission	67	5	–	72	27	99
Creation/termination	37	4	–	41	43	84
Extremity/sufficiency	53	9	–	62	–	62
Context	17	15	–	32	25	57
Total	994	537	272	1803	584	2387

Table 5.1: Statistics of various combinations of relation types. Note that there are 9 instances of Temporal+Correlation and 3 instances of Temporal+Context. This makes the bottom totals less than the sum of the rows.

Connective POS type	**Connective pattern count**	**Connective token count**
Adjectival	9	15
Adverbial	7	152
Coordinating conj.	2	72
Subordinating conj.	13	318
Nominal	8	39
Prepositional	19	495
Verbal	88	513
Complex	68	199
Total	214	1803

Table 5.2: Connective pattern and token counts by POS type.

includes 290 construction variants, covering 192 lexically distinct connectives (e.g., *prevent* __ and *prevent* __ *from* __ rely on the same primary connective word but are counted as distinct constructions). The full constructicon for the latest version of BECAUSE can be found in Appendix C.

Table 5.2 shows the breakdown of different linguistic categories of causal connectives in the corpus. These statistics were computed automatically by assigning the most common category to each of the 214 observed lexical connective patterns, so the numbers are not 100% precise; e.g., different uses of *since* were conflated. Any idiomatic construction that was difficult to place in the other buckets was labeled as "complex." Typical complex connectives include *for* __ *to* __, __ *must* __; *in light of*; *so as to*; *with ⟨determiner⟩ goal of*; and *it takes* __ *to* __.

The table reveals that the most common parts of speech used to express causation are verbs, followed by prepositions and subordinating conjunctions. Complex constructions, too, are far from rare, constituting over 11% of annotated instances. Unsurprisingly, complex constructions also seem to be exceptionally idiosyncratic, accounting for a disproportionate 32% of lexical connective patterns. The spread of causal language across these different POS categories reinforces the need for a unifying representation, and the prevalence and variety of complex connectives reinforces the need for that representation to allow for complex constructions.

5.2.1 What the statistics say about the interactions between causal languange and overlapping relations

A few caveats about interpreting the statistics in Table 5.1: first, Purpose annotations do not overlap with any of the categories we analyzed. However, this should not be interpreted as evidence that they have no overlaps. Rather, they seem to inhabit a different part of the semantic space. Purpose does share some language with origin/destination relationships (e.g., ***toward** the goal of*, ***in order to** achieve my goals*), both diachronically and synchronically; see §4.5.4.

Second, the numbers do not reflect all constructions that express, e.g., temporal or correlative relationships—only those that can be used to express causality. Thus, it would be improper to conclude that over a third of temporals are causal; many kinds of temporal language simply were not included. Similarly, the fact that all annotated Extremity/sufficiency instances are causal is an artifact of only annotating uses with a complement clause, such as *so loud I felt it*; *so loud* on its own could never be coerced to a causal interpretation.

Several conclusions and hypotheses do emerge from the relation statistics. Most notably, causality has thoroughly seeped into the temporal and hypothetical domains. Over 14% of causal expressions are piggybacked on temporal relations, and nearly 7% are expressed as hypotheticals. This is consistent with the close semantic ties between these domains: temporal order is a precondition for a causal relationship, and often hypotheticals are interesting specifically because of the consequences of the hypothesized condition. The extent of these overlaps speaks to the importance of capturing overlapping relations for causality and other domains with blurry boundaries.

Another takeaway is that most hypotheticals that are expressed as conditionals are causal. Not all hypotheticals are included in BECAUSE (e.g., *suppose that* is not), but all conditional hypotheticals are[6]: any conditional could express a causal relationship in addition to a hypothetical one. In principle, non-causal hypotheticals could be more common, such as *if he comes, he'll bring his wife* or *if we must cry, let them be tears of laughter*. It appears, however, that the majority of conditional hypotheticals (84%) in fact carry causal meaning.

Finally, the data exhibit a surprisingly strong preference for framing causal relations in terms of agents' motivations: nearly 45% of causal instances are expressed as Motivation or Purpose. Of course, the data could be biased towards events involving human agents; many of the documents are about politics and economics. Still, it is intriguing that many of the explicit causal relationships are not just about, say, politicians' economic decisions having consequences, but about why the agents made the choices they did. It is worth investigating further to determine whether there really is a preference for appeals to motivation even when they are not strictly necessary.

5.2.2 Comparison to other schemes reveals frequent constructional phenomena

Many of the causal constructions in BECAUSE did not make it into other schemes, as shown in Table 5.3. These statistics were computed by looking up each connective in the other schemes' lexica. FrameNet captures many more connectives than the others, but it often represents them in frames that are not linked to causality, making comparison difficult. Similarly, many of the instances counted as misses for FrameNet in this table are cases where the causal relationship is represented isomorphically: it could be derived from FrameNet, but a causal or (sometimes causal) frame would not be identified without extra

[6]We did not annotate *even if* as a hypothetical, since it seems to be a specialized concessive form of the construction. However, this choice does not substantially change the conclusion: even including instances of *even if*, 77% of conditional hypotheticals would still be causal.

Corpus	**Connective types**	**Connective tokens**
PDTB	10.8%	32.1%
Mirza and Tonelli (2014)	23.0%	44.8%
FrameNet	66.7%	72.0%

Table 5.3: Percentages of the causal connectives in BECAUSE that would be partially or fully annotated under other annotation schemes. Connectives were grouped into types by the sequence of connective lemmas, with minor adjustments to combine variants of the same construction differing only in word order.

	Construction complexity			
FrameNet status	Single word	Contiguous words	Non-contiguous	**Total by FN status**
Represented	1118	55	50	1223
Partially represented	14	39	23	76
Represented isomorphically	212	0	0	212
Missing LU	238	43	0	281
Not representable	0	0	11	11
Total by complexity	1582	137	84	1803

Table 5.4: Counts of causal instances in the BECAUSE corpus by complexity and FrameNet status. See §5.2.2 for details of how these were computed.

information linking specific roles to frames. For example, a purpose phrase like *I left* ***to get food*** would be labeled with a Purpose role, but this role is not explicitly connected to the Purpose frame.

Of course, there are gaps in our constructicon, as well—connectives we never encountered. In particular, Mirza and Tonelli annotated several verbs that we never stumbled across, including *reignite*, *touch off*, and *send*. And FrameNet, with a larger full-text annotation base, has a selection of additional LUs under frames like Causation and Reason.

To determine the prevalence and practical impact of constructional phenomena, it is especially instructive to tease apart FrameNet's coverage in more detail. FrameNet is the most flexible and broad-coverage of the existing schemes for annotating causal meanings, so coverage gaps are less likely to be caused by including only discourse connectives or having a small lexicon.

Statistics on FrameNet's coverage of various types of constructions in BECAUSE are shown in Table 5.4. A few notes on how these statistics were computed:

- The number/contiguity of words in a construction are determined by the standards of FrameNet and the FrameNet Constructicon: for prepositional arguments of the construction, the initial prepositions (e.g., *prevent* ***from***) do not count, even though BECAUSE annotates them as part of the connective.
- In some cases, FrameNet includes a single-word lexical target that participates in the construction, but the FrameNet Constructicon would nonetheless record a construction with multiple "construction-evoking elements." One such example is *enough X for Z to Y*: FrameNet already has an LU for this use of *enough*. These are counted as multi-word constructions.
- The statistics were computed automatically, using sequence of connective tokens as an indicator of the causal construction. However, this method occasionally conflates gappy and non-gappy

versions of a construction, such as X *is* ***sufficient to*** Y and ***sufficient*** X ***to*** Y. Thus, the numbers should not be taken as extremely precise.

- Constructions included in frames that are often but not always causal were counted as fully represented.
- A construction was counted as partially represented if significant portions of the construction were missing (other than argument-initial prepositions).

Not surprisingly, FrameNet's coverage is good for the single-word connectives, fully representing about 70% of them. Many of the remainder are simply LUs that have not yet been documented in FrameNet. (The bulk of the 212 single-word "represented isomorphically" instances are the word *to*, used in its Purpose sense.) No single-word connectives are unrepresented or unrepresentable in FrameNet.

With multi-word connectives, however, constructional effects become apparent. At 40%, coverage of contiguous multi-word constructions is substantially lower than for single words, and fully 28% of contiguous multi-word connectives cannot be represented as LUs without missing significant pieces of the construction. Non-contiguous constructions are relatively uncommon, but 40% of them cannot be represented as FrameNet LUs. Many of these are extremity/sufficiency constructions like *enough to*, which are in FrameNet but without recording the multiple construction-evoking elements; the number would be even higher if these were counted as not represented. In total, the instances that are partially or fully unrepresentable without a richer construction representation constitute nearly 5% of all annotations—an underestimate, considering that a significant number of the connectives not (yet) in FrameNet would be represented only partially.

The prevalence of constructional phenomena speaks to the need for enriching shallow semantic representations with constructions, as the FrameNet Constructicon has begun to do. The BECAUSE lexicographic work could contribute to FrameNet's efforts, serving as a basis for causality-related entries in the FrameNet Construction. It might first be necessary to reorganize the causality-related frames as suggested by Vieu et al. (2016). Next, lexicographers would need to examine each construction and define its "construct elements." Automating this task would be difficult, since construct elements are defined by prose descriptions of constraints that would not be obvious to a machine, such as "an AP headed by *long*." However, with some human intuition, mapping the BECAUSE representation into FrameNet Constructicon entries ought to be fairly straightforward, since both represent an instance of a construction as a sequence of fixed or semi-fixed elements and slot-fillers.

5.2.3 Novel constructions still appear often

To assess what fraction of causal connectives are represented in the corpus, we examined how the number of constructional patterns increases as more of the corpus is seen. We first split the corpus sentences into 10 random segments and counted the unique sequences of connective lemmas within each segment. We then computed the cumulative sums of these counts for the first i segments, with i ranging from 0 to 10, obtaining estimates of the number of connective patterns in the first $i \times 10\%$ of the corpus. Finally, we repeated the process for a total of 20 randomizations and averaged the results. The plot of number of patterns vs. fraction of the corpus is shown in the solid line in Figure 5.1.

Although the growth rate of new connectives is slowing down by the 100% mark, it seems clear that there are many more causal constructions to be discovered. This is consistent with our anecdotal experience from annotating: as we were finishing the corpus, we would still encounter a new construction every 2–3 documents.

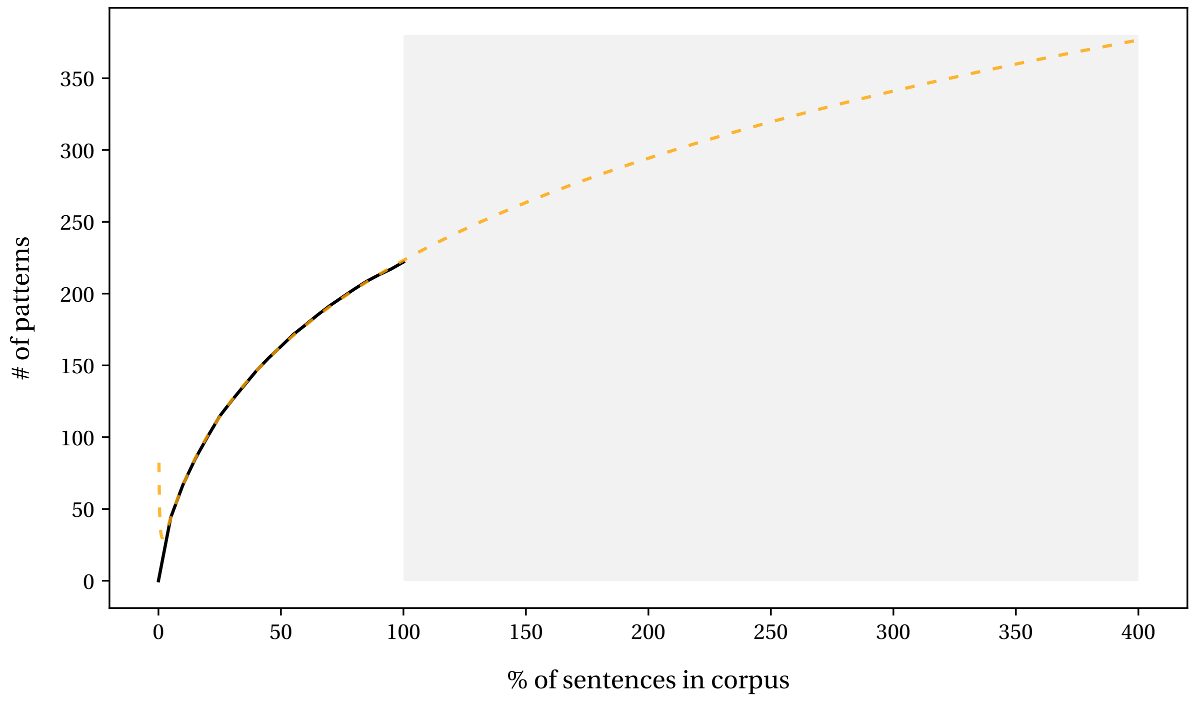


Figure 5.1: Saturation of constructions as corpus size increases. The dashed line is a quadratic fit in log space; see the text for details. The dashed line in the shaded region extrapolates the fit beyond 100% of the current corpus.

The growth rate is not well-modeled as logarithmic: a linear fit in log space—i.e., a fit of the form $f(x) = \lambda_1 \log(x) + \lambda_2$—fits very poorly, with the logarithmic function flattening out sooner. A much better fit is obtained from a quadratic fit in log space, shown with the dashed line in Figure 5.1. (The exact fit is $f(x) = 11.6 \cdot \log(x)^2 + 94.6 \cdot \log(x) + 223.2$.) On reflection, this is a reasonable growth rate to see for connectives: the rate of new words is approximately logarithmic, and while most novel connectives can be characterized as new words (or word senses), we see some contribution from arbitrary combinations —most commonly pairings—of new words, leading to an approximately quadratic term. Thus, new connectives show up at a rate governed by both the log of the corpus size and the square of that log.

New connectives are often very close to old ones, but with synonyms substituted in—e.g., *sufficient* instead of *enough*. Counting such connectives separately contributes to the quadratic growth rate, since new words and synonyms of known words can combine to form new connectives.

Assuming this is a good approximation of the growth rate, the rate should become flatter by about 3–4 times the current corpus size, where it reaches the vicinity of 1.7 times the current number of connective patterns.

5.3 Conclusion

The BECAUSE corpus offers exhaustive annotations for causality and its semantic neighbors on a substantial number of documents, including several different document types and genres. Many of these documents have also been annotated with other schemes, including syntactic parses. The corpus thus offers a rich resource for analyzing real-world causal language, yielding tantalizing insights about the interactions between causality and other domains and about the importance of constructional phenomena to cognitively basic semantic relations. Of course, it also presents a ripe target for automated tagging, as we shall see in the coming chapters.

In the future, it would be interesting to do a more formal cross-genre analysis of how and how often causality manifests, comparing, e.g., textbooks, scientific papers, fiction, and news, as well as different subject areas such as physical sciences, history, current events, and human interest stories. Such studies might also reveal interesting new modes of expression for causality.

We hope that the new corpus, our annotation methodology and the lessons it provides, and our observations about linguistic competition will all prove useful to the research community.

Part II

Automatic Tagging

CHAPTER 6

Introducing the Tagging Task

In Part I, I presented a shallow semantic annotation scheme for causal language. The next step is to create semantic parsers based on that scheme—automated systems that can apply the scheme to previously unseen text. Following our project's basis in construction grammar, we cast this task as one of recognizing constructions and their slot-fillers: a parser must tag the connective and its arguments. Notably, we do not explicitly operationalize the mechanisms that construction grammar theory posits for parsing constructions. In keeping with the SCL approach, with respect to the tagging methods CxG provides an operationalized data representation and conceptual inspiration; see §2.4.

The parsing task is made considerably more difficult by the fact that our linguistic targets can be complex constructions. In traditional semantic parsing, the set of possible triggers in a sentence is simply the set of words in the sentence (possibly augmented with a predefined list of MWE phrases). But once we embrace arbitrarily complex constructions as connectives, as BECAUSE does, the space of possible triggers grows exponentially. A connective can be any set of words in the sentence—i.e., any member of the superset of words. Thus, the task demands methods that can trim this space down to a manageable size.

In this chapter, I first **contextualize our tagging task** among existing NLP tasks and systems (§6.1). In addition, I compare the expressive capacity of the systems in the chapters to come to the most similar of those existing systems (§6.1.2). I then **define the parameters** of the task (§6.2), describe a **simple memorization-based system** as a benchmark (§6.3), and specify the **evaluation metrics** (§6.4). In the following chapters (7 & 8), I present several different systems that perform the task.

6.1 Background and related work

This section presents related work that specifically addresses tasks similar to ours. For a review of the basic NLP concepts and terminology that our work relies on, see §A.2. (Although neither this section nor the appendix defines the various kinds of classifiers described below, for the purposes of this discussion any unfamiliar ones can be safely glossed over.)

	Our task	**PropBank SRL**	**FrameNet SRL**	**PDTB tagging**	**Traditional IE**	**Open IE**
Finds relational tuples in text	✓	✓	✓	✓	✓	✓
Agnostic to how relation is expressed	✓	✗	✓	✗	✓	✓
Cares whether text intended to express relation	✓	✓	✓	✓	✗	✗
Canonicalizes relations	✓	✓	✓	✓	✓	✗
Canonicalizes arguments	✗	✗	✗	✗	✓	✗
Evaluated on	Mention $P/R/F_1$	Mention $P/R/F_1$	Mention $P/R/F_1$	Mention $P/R/F_1$	Extracted relation $P/R/F_1$	Extracted relation precision

Table 6.1: A comparison of the goals of our task against well-established similar tasks.

.

6.1.1 Existing semantic parsers and parsing tasks

6.1.1.1 Parsing general-purpose shallow semantic annotation schemes

In addition to previous attempts to incorporate CxG into NLP (see §2.3), our work inherits from traditional shallow semantic parsing, which generally involves similar tasks to ours. Taggers have been developed for all three of the major annotation schemes discussed in §3.3.1. Our task also shares some elements with information extraction (IE).

Table 6.1 compares the goals of each task, which are described in more detail below. Broadly speaking, PropBank, FrameNet, and PDTB tagging share our task's focus on identifying what the text intended to express, but with varying degrees of flexibility regarding linguistic realizations. IE tasks, meanwhile, share our task's breadth of linguistic expressions, but they focus more on extracting a set of correct facts than on interpreting the intended meaning of each individual sentence.

PropBank tagging Automatically tagging PropBank annotations is arguably the easiest shallow semantic parsing task, since nearly every verb is a valid trigger and the repertoire of argument labels is small. First pioneered by Gildea and Palmer (2002), efforts in this space gained serious attention with the CoNLL conference's 2004 and 2005 "shared tasks" (Carreras and Màrquez, 2004, 2005) on SEMANTIC ROLE LABELING (SRL), a term often used for tagging in the PropBank or FrameNet paradigms. The task was extended in 2008 and 2009 (Surdeanu et al., 2008; Hajič et al., 2009) to cover PropBank-style nominal predicates from NomBank (Meyers et al., 2004), and to include simultaneously predicting syntactic dependencies alongside PropBank/NomBank argument structures.

Over the course of these shared tasks, dozens of taggers have been created for PropBank. Before the 2008 task, determining each verb's sense was rarely addressed at all; since then, most systems have used some form of classification to determine each verb's sense (see Surdeanu et al., 2008; Hajič et al., 2009). Techniques for tagging the arguments are more varied. (For examples and fuller discussions of the methods below, as well as furthur citations, see Carreras and Màrquez, 2004, 2005; Surdeanu et al., 2008; Hajič et al., 2009.) One popular strategy is to build a multi-stage pipeline in which likely argument spans are first identified, then tagged with the best-fitting argument labels. These two stages can also be followed up with a post-processing or reranking step (e.g., Ciaramita et al., 2008). Other systems have gravitated toward the single-stage sequence tagging option, which treats PropBank SRL as a word-by-word classification problem: for each verb, the tagger assigns each word in the sentence a tag indicating its role (None, Arg0, Arg1, etc.). Such taggers frequently convert the argument labels to some variant of the BEGINNING-INSIDE-OUTSIDE (BIO) format, where each word is marked as the **b**eginning of a span, **i**nside a span, or **o**utside any span. For example, a two-word Arg0 followed by a non-argument word would be marked with the sequence B-A0 I-A0 O. Additional approaches include inference over graph structures (e.g., Sun et al., 2008) and greedy transition-based tagging (see §8.2.2 for details and a literature review).

All manner of statistical machine learning tools have been employed for the various stages and styles of tagging. Many taggers have used standard classifier models like support vector machines (SVMs; e.g., Samuelsson et al., 2008), maximum-entropy classifiers (e.g., Zhao et al., 2009), and perceptrons (e.g., Ciaramita et al., 2008). Others (e.g., Moreau and Tellier, 2009) have used graph-based techniques such as conditional random fields (CRFs, described in §7.2.2.2; Lafferty et al., 2001). In most cases, the machine learning models rely on a variety of hand-engineered features, including surface-level features (word form, word position, span length, etc.), automatically extracted linguistic features (POS tags, phrase boundaries, verb voice, and so on), and automatically extracted semantic features (e.g., named entities and word embeddings).

The most recent systems are built with neural networks; see §8.2.1 for further discussion of these approaches. State-of-the-art SRL systems (e.g., Zhou and Xu, 2015; FitzGerald et al., 2015) typically achieve between 70 and 85% F_1 on predicate senses and argument labels, depending on the method, the dataset, the exact evaluation metric used, and whether the test data was from the same domain as the training data.

The PropBank SRL task is generally much easier than tagging causal constructions. SRL triggers are almost always given as input, and even when they are not they can be easily recognized as verbs or nominalizations. Also, PropBank almost always annotates complete phrases as arguments, whereas BECAUSE arguments are frequently broken up. Thus, we cannot directly apply typical PropBank tagging methods. One of our pattern-based methods does adapt the CRF-based sequence tagging technique for argument tagging; see §7.2.2.2. We also use many similar machine learning features in our pattern-based systems; see Tables 7.2 and 7.3.

Note that, while the full PropBank tagging task does include assigning sense labels to each verb, this is not a full canonicalization of the relation, since similar senses of different verbs are not linked. Also, many taggers do not perform this portion of the task. (Hence the partial check in the pertinent row of Table 6.1.)

FrameNet tagging A more difficult task that directly influenced our work is FRAME-SEMANTIC PARSING (Baker et al., 2007), the task of automatically producing FrameNet annotations. In contrast to PropBank, words and MWEs from many different parts of speech can act as FrameNet triggers, as described in §3.3.1 (though as noted there, FrameNet still imposes some restrictions on how relations can be expressed). Taggers therefore cannot rely on POS tags to discern the triggers; they must automatically identify diverse frame triggers (or TARGETS, in the FrameNet parlance), as well as assigning them senses (picking the relevant frame) and tagging their argument spans (or FRAME ELEMENTS).

Work on frame-semantic parsing has been sparser than on PropBank SRL (despite the fact that SRL actually started with FrameNet tagging; see Gildea and Jurafsky, 2002), but FrameNet has spawned its own shared task (Baker et al., 2007). On the full frame-semantic parsing task, most systems have taken a pipeline approach. Targets are first identified with either a whitelist or a set of simple rules. They are then assigned a frame, which determines the available frame elements, and finally the frame elements are identified and labeled. Variants of this approach were taken by Johansson and Nugues (2007b), the UTD-SRL system (Bejan and Hathaway, 2007), and the SEMAFOR system (Das et al., 2014). The first two used SVMs and, in the case of UTD-SRL, maximum-entropy models to classify each target word into an appropriate frame, classify possible argument spans/boundaries as correct or incorrect, and finally label each argument span with a frame element label. Both used large collections of frame- or word-specific classifiers. SEMAFOR, meanwhile, used just one log-linear classifier for frame identification, based on a latent-variable model of "prototypical" frame instances, and a second log-linear classifier for the role labels of candidate argument spans. It then applied a global beam search or, in later versions, an integer linear program to find the best set of non-overlapping argument spans. SEMAFOR achieves F_1 scores of about 68% for frame identification and 80% for argument identification. Other systems have recently achieved even better results with neural networks; see §8.2.1.

Frame-semantic parsing is both easier and harder than causal construction tagging. On the one hand, the scope of semantic relations covered by FrameNet is far broader, and the inventory of frame-specific argument types is accordingly larger as well. On the other hand, as discussed in earlier chapters, FrameNet's lexical triggers tend to be less complex than BECAUSE's, and the approaches that have been used to find these triggers would not work for arbitrary, non-contiguous constructions. We do take some inspiration from these systems, though: the pipeline architecture we adopt in Chapter 7 is loosely inspired by SEMAFOR's, but with a multitude of classifiers like those of Johansson and Nugues.

PDTB tagging End-to-end discourse tagging in the style of the PDTB entails predicting many pieces of information beyond predicate-argument structures: the PDTB includes attribution spans (e.g., *said Mr. Jones*), implicit connectives, entity-based coherence relations, and more. In many cases it also requires tagging arguments from multiple sentences. The first system to take on the full task was Lin et al.'s (2014), which consisted of a long pipeline of components designed to simulate the process followed by corpus' human annotators. Candidate connectives were drawn from a whitelist and classified for sense labels (including the null label). If a classifier determined that arguments were in the same sentence, simple syntactic rules were applied to find the argument spans; otherwise, the previous or following sentence would be selected.

The more widely addressed version of the task is the slightly narrowed one proposed for two CoNLL shared tasks (Xue et al., 2015, 2016), termed SHALLOW DISCOURSE PARSING. This version of the task required systems to tag the primary word of the connective (or the existence of the discourse relation, for non-explicit relations), the two argument spans, and the connective's sense label. All such systems to date have followed the overall pipeline architecture of Lin et al., with varying machine learning methods at each stage. Approaches to disambiguating possible connectives have included SVM, maximum-entropy, naïve Bayes, and decision tree classifiers, as well as neural networks and token-level sequence tagging with models like CRFs. For argument identification, the menu includes CRF sequence taggers and classifiers applied to syntactic subtrees. State-of-the-art shallow discourse parsers achieve whole-relation F_1 scores of just under 30% on unseen data (exact match of arguments and connective spans), and connective and argument F_1 scores of about 92% and 52%, respectively.

The best-performing system[1] for English in the 2016 shared task was the Oslo-Potsdam-Teesside (OPT) system (Oepen et al., 2016), which uses an SVM to filter possible connectives; an SVM-based ranker to rank candidate argument spans; some handcrafted rules to tweak those argument spans; and an ensemble of an SVM, a decision tree forest, and a majority-class classifier to classify relation types. The ECNUCS system (Wang and Lan, 2015, 2016), the 2015 winner, is competitive, as well. It closely

[1]As measured on the "blind" test set that participants did not have access to during development.

follows the pipeline architecture of Lin et al., but with several enhancements. The most significant of these is finer-grained argument extraction when the arguments are not in the same sentence: clauses in the preceding sentence are each classified to determine whether or not they belong to Arg1, and likewise for Arg2 and the clauses in the sentence with the connective.

The similarities and differences between our work and shallow discourse parsing echo those between our work and frame-semantic parsing: the trigger/argument tagging structure of the task is similar, and similar pipeline and and machine learning techniques may apply. However, our semantic scope is narrower, while the complexity of causal connectives and the less-predictable structure of Cause and Effect arguments makes the task more difficult.

6.1.1.2 Causality-specific taggers

A handful of projects that annotated specifically for causality (see §3.3.2) also built associated automatic taggers. Mirza and Tonelli (2014), following Mirza et al. (2014), built an SVM-based system for predicting causal links between events using their fairly broad-coverage causal extensions to TimeML annotations. Their work differs from ours in that it requires arguments to be TimeML-recognized events; it requires causal connectives to be contiguous spans; and it relies on gold-standard TimeML annotations for prediction. Their annotation scheme also employs a somewhat different definition of causality. Similarly, Bethard et al. (2008) built an SVM classifier for causal/non-causal relations between events conjoined by *and*. Both of these taggers essentially perform pairwise classification rather than full shallow parsing.

More recently, Hidey and McKeown (2016) automatically constructed a large dataset with PDTB-style AltLex annotations for causality. Using this corpus, they achieved high accuracy in finding causality indicators. This was a somewhat easier task than ours, given their much larger dataset and that they limited their causal triggers to contiguous phrases. Their dataset and methods for constructing it, however, could likely be adapted to improve our systems.

As in annotation, work on the BioCause corpus is closer to ours. In particular, Mihăilă (2014) developed machine learning approaches for tagging causal relations in biomedical texts. Causal triggers were found via either sequence tagging or candidate classification, and arguments were found similarly to the PDTB parser pipelines, both using a variety of hand-engineered features. With semi-supervised data expansion techniques, the BioCause team was able to reach 82% F_1 for causal triggers. Although their definitions of causality and causal triggers diverge significantly from ours, their data expansion techniques would be an appealing avenue for future work.

6.1.1.3 Information extraction

In addition to SRL, our shallow semantic parsing task resembles tasks of INFORMATION EXTRACTION (IE). The goal in IE is to automatically extract structured information from free text, generally in the form of logical propositions such as PrizeWinner(Kip Thorne, Nobel, Physics, 2017). In traditional IE, the relation itself and all members of the relational tuple must be disambiguated and normalized into a form amenable to inclusion in a database. OPEN INFORMATION EXTRACTION (OIE; Banko et al., 2007) relaxes this constraint somewhat, allowing both the relation and the participants to be left as non-canonical free text. For example, the aforementioned relation could equally well be extracted as ⟨*Thorne*, *received*, *the 2017 Nobel Prize in physics*⟩ or ⟨*Kip Thorne*, *won*, *this year's physics Nobel*⟩, depending on the source text.

It is worth emphasizing the similarities and contrasts between IE and our causal language task (summarized in Table 6.1). Like both SRL and our task, IE attempts to find relations of interest and their arguments. OIE in particular resembles SRL and our task in that it does not demand that argument spans be canonicalized. IE also shares one of our key departures from conventional SRL: it throws open the floodgates to any and all linguistic expressions of the relations of interest.

However, there is an important sense in which our task is still closer to SRL: IE systems are evaluated on whether, given a set of documents, they ultimately extract a correct body of facts. They are not typically concerned with whether those facts were explicit or merely implied, nor does it matter whether an individual sentence was correctly interpreted; what matters is that the resulting body of assertions is correct. In contrast, we follow SRL in seeking **all explicitly stated** relations: we aim to tag each mention of a relation, and evaluate accordingly. This is particularly important when it comes to causality: our principled annotation scheme for causal **language** sidesteps the question of what counts as real-world causation, which IE approaches would have to contend with (see §3.2).

The most relevant IE work has focused on so-called EXTRACTION PATTERNS, the inspiration behind the pattern-based methods for tagging causal language described in the next chapter. I therefore review pattern-oriented IE work as a prelude to that chapter (§7.1).

6.1.2 A preview: comparing our methods' expressive capacity to others'

The raison d'être of our task is the expressive flexibility it inherits from the BECAUSE data representation. Accordingly, our automated taggers aim to allow for flexibility in ways that previous systems often have not. Although these systems have yet to be introduced, Table 6.2 compares them against several systems for related tasks, serving to illustrate some key features we seek to implement. I will periodically refer back to this table in the coming chapters when discussing related work for specific methods.

The systems compared in the table are (with parenthesized references to where in the thesis each is discussed further):

- Causeway-S (abbreviated CW-S) and Causeway-L (CW-L), our pattern-based causal language taggers (Chapter 7)
- DeepCx, our neural network-based causal language tagger (Chapter 8)
- Open-SESAME (Swayamdipta et al., 2017) and Täckström et al. (2015), neural network-based taggers for FrameNet and PropBank (§8.2.1)
- OPT (Oepen et al., 2016), a PDTB tagger described above (§6.1.1.1)
- DARE (Adolphs et al., 2011), Ollie (Mausam et al., 2012), and Odin (Valenzuela-Escárcega et al., 2016), IE systems that match highly flexible patterns to identify relations and their arguments (§7.1)

A few notes about the table:

- The top half of the table is concerned with representational capacity—i.e., what the underlying data representation supports, and what subset of that can actually be produced by the tagger. The half below the line addresses what elements the tagger can attend to or ignore.
- "Need not come between arguments" indicates that the trigger can be elsewhere in the sentence—e.g., *because* ⟨*x*⟩, ⟨*y*⟩, not just ⟨*x*⟩ *because* ⟨*y*⟩. "Need not be on parse path..." indicates the same concept, but for parse paths: some systems only notice trigger words that are found on the parse path between the two arguments.
- When it was discernible from the literature or personal knowledge, the table notes where a feature that is not present could easily be added without re-architecting the system.

- Half checks were awarded for the following reasons:
  - Multiple simultaneous relations with DARE and Odin: these systems theoretically allow a pattern match to be associated with any relation label. Thus, one pattern could assign a conjunction of labels, or multiple nearly-identical rules could be included that would assign different relation labels when they match.
  - Discontinuous arguments in Odin: the data structure that represents the text span matched by a pattern gives only a single contiguous span. However, the pattern that produced the match may consist of a number of smaller sub-patterns interrupted by words that are ignored. These sub-matches can be recovered programmatically with the Odin API, so it is possible to reconstruct a discontinuous argument excluding the irrelevant words.
  - Single-word arguments in Causeway-L: Causeway-L's pattern matching assumes two arguments. However, the sequence tagging algorithm for argument labels can end up labeling only one argument.
  - Overlap between triggers in DeepCx: allowed, but with some restrictions if the first word of the trigger is shared; see §8.3.2 and §8.3.6.
  - Entity tags in Ollie: Ollie does not take NER tags as input, but it does generalize some nouns to Person and Location categories.
  - Entity tags in Causeway & dependency labels in Causeway-L: used only for filtering pattern matches, not for pattern-matching.
  - Non-argument and non-trigger words in Open-SESAME, Täckström et al., and OPT: words outside the argument spans that are ultimately selected can influence the scores of other argument span alternatives, which of course factor into which span labeling receives the top score.
  - Non-argument and non-trigger words in Causeway-L: used for features in sequence labeling for arguments.
  - Linear order in Causeway-S and OPT: used only in the features for filtering out false positives and (in OPT) for argument span scoring.

6.2 Task description

6.2.1 Task components: connective discovery and argument identification

Because it is so hard to identify the relevant components of a construction—tenses, grammatical relations, etc.—we do not task our systems with explicitly tagging all of these elements. Instead, we apply the SCL approach: We tag the words that participate in a causal construction as a proxy for that construction. We leave it to annotators (when humans are annotating) or to machine learning (during automated tagging) to assess when the full constellation of constructional elements is present.

We define the task of tagging causal language to be reproducing a subset of the annotations of the annotation scheme described in Chapter 4. For evaluation purposes, we split this task into two parts:

1. **Connective discovery**, in which the spans of causal connectives are annotated. A connective span may be any set of tokens from the sentence. This can be thought of as recognizing instantiations of causal constructions.

		Systems in this thesis			Shallow semantic parsers			Info. extraction systems		
	System	**CW-S**	**CW-L**	**DeepCx**	**Open-SESAME**	**Täckström**	**OPT**	**DARE**	**Ollie**	**Odin**
	Underlying data representation	BECAUSE			FN	FN/PB	PDTB	*n*-ary entity relations		
Triggers…	…can be discontinuous	✔	✔	✔	✗	✗	–	✔	✔	✔
	…can be of arbitrary parts of speech	✔	✔	✔	✔	✔	✗	✔	✔	✔
	…can be null	✗	✗	✗	✗	✗	✔	✔	–	–
	…need not come between arguments	✔	✔	✔	✔	✔	✔	✔	✔	✔
	…need not be on parse path between arguments	✔	✔	✔	✔	✔	✔	–	✔	✔
	…can express multiple relations at once	–	–	–*	✗	✗	–	✓	–	✓
An argument can be…	…discontinuous	✔	✔	✔	✔	✔	–	–*	✔	✓
	…drawn from multiple disjoint parse subtrees	–	✔	✔	✔	✔	–	–	–	✔
	…in a different sentence from the trigger	–	–	–	–	–	✔	–	–	–
Argument count can be…	…1	–*	✓	✔	✔	✔	✔	–	–	✔
	…> 2	–*	–*	✔	✔	✔	✗	✔	–	✔
Overlaps are allowed between…	…different triggers	✔	✔	✓	✗	✗	✗	✔	✔	✔
	…arguments of one trigger	–	–	✔	✗	✗	✗	–	–	–
	…a trigger and its arguments	–	✔	✔	✔	✔	✗	–	–	–
Tagger…	…learns from data	✔	✔	✔	✔	✔	✔	✔	✔	–
	…evaluates evidence beyond exact pattern match	✔	✔	✔	✔	✔	✔	–	–	–
	…can generalize to unseen triggers	–	–	✔	–	–	–	✔	✔	n/a
Tagging can discriminate directly on…	…entity tags	✓	✓	–*	–*	–*	–*	✔	✓*	✔
	…dependency/constituent labels	✔	✓	✔	–	✔	✔	✔	✔	✔
	…words beyond those in trigger and arguments	–	✓	✓	✔	✓	✓	–	–	✔
	…linear order of words	✓	✔	✔	✔	✔	✓	–	–	✔
…but can selectively ignore…	…the trigger's POS	–	–	✔	✔	✔	✔	✔	✔	✔
	…the arguments' POS	✓	✓	✔	✔	✔	✔	✔	✓	✔
	…dependency/constituent labels	–	✓	✔	–	✔	✔	✔	–	✔
	…words beyond those in trigger and arguments	–	✓	✔	✓	✓	✓	✔	–	✔
	…linear order of words	✓	✔	✔	✔	✔	✓	–	–	✔

Table 6.2: A comparison of trigger/connective and argument patterns detectable by different systems. ✗= not supported by underlying representation. ✓= partially supported (e.g., used only to filter, or must be either always specified or never specified). –= supported by data representation but not by system. * = would be straightforward to add (to the extent the representation allows).

2. **Argument identification** (or argument ID), in which Cause and Effect spans are identified for each causal connective. This can be thought of as identifying the slot-fillers of the causal construction.

All of the systems described in the coming chapters use at least partially distinct machinery for each subtask. Still, they do not perform either subtask entirely independently; they use information from each subtask to inform decisions about the other.

The task is defined in terms of text spans. Nonetheless, to achieve a high score on it, a tagger must respond to the meaning of the construction and arguments in context, just as human annotators do. This can be achieved by analyzing indirect cues that correlate with meaning, such as lexical information, syntactic structure, and tense/aspect/modality information.

Compared to the annotation scheme, our task is limited in two important ways. First, we tag only causal relations, not the overlapping relations that may accompany them. Second, we do not distinguish between types or degrees of causation. We leave both of these extensions to future work.

Nonetheless, the task is more difficult than it may appear. Two of the reasons for this are familiar issues in NLP. First, there is a surprisingly long tail of causal constructions (see §5.2.3). Second, recognizing these constructions inherits all the difficulties of word sense disambiguation. Consider again examples 1–9 (Table 1.1), whose connectives—*promotes*, *impedes*, *so*, *because*, *from*, *contingent on*, *opens the way for*, *for* __ *to* __, and *so* __ *that* __—demonstrated the diversity of causal language. Every one of these connectives except 6 (*contingent on*) and 7 (*opens the way for*) has a non-causal meaning.[2] The third and most important reason is that, as mentioned above, our space of possible lexical triggers is very large compared to that of traditional semantic parsing problems.

6.2.1.1 Assumed inputs for the task

We assume as input text documents, split into sentences tokenized with the Stanford CoreNLP pipeline (Manning et al., 2014), where each sentence has been annotated with CoreNLP POS tags (Toutanova et al., 2003) and a dependency parse in the Universal Dependencies scheme. We also assume that Stanford CoreNLP's token lemmas and NER tags (Finkel et al., 2005) are available, although only the systems described in Chapter 7 make use of them.

Rather than using dependency parses as input, we could have built on top of parsers that produce both syntactic and semantic structures, such as the English Resource Grammar (ERG; Flickinger, 2011) or another HPSG variant. Though these parsers can produce impressively sophisticated analyses, we elected to use dependency parsers because they proved significantly more robust; there were many sentences in our corpus that we could not parse with ERG. However, incorporating semantic information from such a system when it is available would be an interesting extension for future work.

6.2.2 A more formal description of the task's inputs and outputs

The subsection above describes the task in intuitive terms. Here, I describe a data format that more rigorously defines the inputs and outputs of the task. The format is tailored to our particular task definition, but it could easily be adapted to many other kinds of construction tagging tasks (see §6.2.2.1).

The format is based on the CoNLL-U format for dependency parsing (Nivre et al., 2016) and the CoNLL-2008 format for joint parsing of syntactic and semantic dependencies (Surdeanu et al., 2008). As in these formats, an annotated document is represented as a series of sentences separated by blank lines. Each sentence is represented with one line per token. See Figure 6.3 for an example sentence.

Each token line starts with the following tab-separated fields:

[2] *Because* can be used in a discourse sense—roughly, *I'm saying this because. . . .*

ID	FORM	LEMMA	POS	HEAD	DEPREL	DEPS	NE	CXN-1	CXN-2
1	Amy	Amy	NNP	4	nsubj	4:nsubj	B-PER	E	_
2	’s	be	VBZ	4	aux	4:aux	O	E	_
3	not	not	RB	4	neg	4:neg	O	E	_
4	coming	come	VBG	0	root	0:root	O	E	_
5	because	because	IN	9	mark	9:mark	O	T	_
6	she	she	PRP	9	nsubj	9:nsubj	O	C	_
7	’s	be	VBZ	9	cop	9:cop	O	C	_
8	so	so	RB	9	avdmod	9:advmod	O	C	TC
9	tired	tired	JJ	4	advcl	4:advcl	O	C	C
10	she	she	PRP	12	nsubj	12:nsubj	O	C	E
11	’s	be	VBZ	12	aux	12:aux	O	C	E
12	falling	fall	VBG	9	ccomp	9:ccomp	O	C	E
13	over	over	RP	12	compound:prt	12:compound:prt	O	C	E
14	.	.	.	_	_	_	_	_	_

Table 6.3: A sample sentence annotated for causal language in our CoNLL-style format

1. **ID:** Token index, starting at 1 for each new sentence
2. **FORM:** The raw form of the token from the sentence
3. **LEMMA:** Lemma of the token; _ if lemmas are unavailable
4. **POS:** POS tag of the token
5. **HEAD:** The ID of the token’s dependency parent (0 if it is the root word)
6. **DEPREL:** The dependency relation between the token and its parent (root if it is the root word)
7. **DEPS:** Enhanced dependency graph in the form of a list of head-deprel pairs; normally just HEAD:DEPREL unless the parser supports dependency links that break the tree structure
8. **NE:** Named entity tag of the token, in BIO format (e.g., B-PER indicates the start of a person entity, I-PER indicates a continuation of a person entity span, and O indicates a word that is not part of a named entity); _ if entity tags are unavailable

After these columns, there is one additional column for each construction instance of interest (the CXN columns in Figure 6.3). In each such column, each token is marked with a set of labels indicating its role(s) in the corresponding construction instance. The available labels are T for trigger (i.e., connective), C for Cause, E for Effect, and M for Means. A token may have multiple labels if it performs multiple functions with respect to the construction instance. For instance, the word *so* is Figure 6.3 serves both as part of the trigger and as part of the cause for the *so X that Y* construction, so it is marked as TC.

In training, all available data is provided. In testing, all columns after NE are omitted, and it is the job of the tagger to reconstruct them.

Note that nothing about the methods described in the forthcoming chapters relies on this format; in fact, we do not currently use it for data-processing. However, the format does serve as a rigorous description of the inputs and outputs, and it clarifies what the space of possible outputs is. It would also be useful as a format for data interchange or shared evaluation, especially for similar construction-oriented tasks in the future.

6.2.2.1 Possible format extensions

As with other CoNLL-based formats, columns can be added to incorporate additional information about each token. For example, it may be useful in future to incorporate additional columns from the CoNLL-X format. In particular, the POS column could be split into UPOSTAG (universal POS tag) and XPOSTAG (language-specific POS tag), and a FEATS column could be added for morphological features. Coreference columns could also be included.

In the current formulation, there is no place to record metadata about each construction instance. For instance, we might want to extend the task to include predicting the degree and causation type for causal language instances, in which case the data format needs a way to represent these attributes. Similarly, a construction-based FrameNet tagger would need to indicate frame labels. The simplest solution is to add a row at the start of each sentence with an underscore (_) in the eight fixed columns, and a label or set of labels/metadata for each CXN column. Thus, the example in Figure 6.3 would include an initial row whose only non-blank entries would be Motivation in the CXN-1 column and Consequence in CXN-2.

Of course, if the format is extended to other construction types, labels other than C, E, and M will be needed for the argument roles, but that is no impediment; any letters or even Unicode characters would do to uniquely identify an argument type. Alternatively, roles could be specified as a comma-separated list of full role names.

6.3 A simple path memorization benchmark for comparison

Our task differs significantly from existing tasks such as frame-semantic parsing, both in the forms of allowable triggers and in the semantic relationships targeted. Our results are therefore not directly comparable to a frame-semantic parsing baseline. Instead, we compare our results against a simple system of our own construction.

The system is based on memorizing causal/non-causal labels for dependency paths between possible argument heads and possible connective words. At training time, it first extracts the set $\mathscr{L}$ of known sequences of connective lemmas—i.e., $\mathscr{L} = \{\langle \textit{prevent}, \textit{from} \rangle, \langle \textit{because}, \textit{of} \rangle, \ldots\}$. It then examines the parse paths between these connectives and every pair of words in the sentence. It builds a table of how many times each combination of connective sequence and parse path has or has not been marked as causally related: for every appearance of a connective $L \in \mathscr{L}$, it considers every possible combination of L with two content words c (possible Cause) and e (possible Effect) from the sentence. The system finds $d_{c,L}$ and $d_{e,L}$, the shortest dependency paths to L from c and e, respectively. (The shortest path to L is defined as the shortest path to any connective word $\ell \in L$.) If L is annotated as signalling a causal relationship between Cause head c and Effect head e, the system increments the count $t\{L, d_{c,L}, d_{e,L}\}$; otherwise, it decrements it. Both c and e must be less than two dependency links from some $\ell \in L$ for the algorithm to count the tuple.

The test algorithm finds all tuples (L, c, e) for each test sentence, subject to the same constraint on the parse distance. If $t\{L, d_{c,L}, d_{e,L}\} > 0$, the system annotates a causal language instance with connective L and argument heads c and e. Argument heads are expanded into spans using the same algorithm as for Causeway-S, described below in §7.2.1.3.

I refer to this algorithm as the benchmark algorithm in the chapters to come.

6.4 Evaluation metrics

In keeping with the subtasks defined above (§6.2.1), we measure performance on connective discovery and argument ID separately. Our metrics for comparing automated taggers with human gold-standard data on these subtasks are similar to those used above (§4.4) for human inter-annotator agreement. For connective discovery we use precision, recall, and F_1 measure, requiring connectives to match exactly; no partial credit is awarded for, e.g., detecting *because* when the correct connective is *because of.* For argument ID, we split out metrics by Causes, Effects, and Means. For each argument type, we report:

- F_1 of connective/argument pairs, where matches must match exactly to earn credit;
- F_1 of connective/argument pairs, where 50% of the tokens in the larger of the two spans must match to earn credit; and
- the average Jaccard index for gold-standard vs. predicted spans, **given a correct connective**.

Punctuation tokens are not included in exact span comparisons or Jaccard index calculations.

Jaccard indices are reported to give a sense of how far off argument tagging is when it does not provide an exact match. This metric is reported only for correctly predicted connectives, as there is no way to automatically evaluate argument span overlap for false positives. As a result, Jaccard indices are **not directly comparable** between pipelines—the scores represent how well argument ID works given the previous stage, rather than in an absolute sense. Argument precision, recall, and F_1 scores, on the other hand, represent how well argument ID and connective discovery work together, since connective false positives and false negatives count against these argument scores.

The subset of instances on which we evaluate in Chapter 7 is slightly more restricted than the one we use in Chapter 8. The systems in Chapter 7 are designed to find instances with both a Cause and an Effect span, so we evaluate them only on such instances, and we ignore Means spans. In Chapter 8, we expand the evaluation to include all instances, even those with just one argument, and we include Means in the evaluation.

6.5 Conclusion

In this chapter, I have reviewed some of the techniques and systems that our causal language taggers are either partially comparable to or rely on as input. I have also spelled out the task they will be expected to perform. The stage is now fully set to discuss the actual taggers in Chapters 7 and 8.

CHAPTER 7

Pattern-Based Tagging Methods

One approach to automating a tagger is to build a pipeline that imitates the tagging procedure of the human annotators. (This is precisely what most PDTB taggers do; see §6.1.1.1). In our case, the first step under this approach should be finding patterns of words that look plausibly similar to constructicon entries. Of course, the surface patterns are just basic cues that **may** indicate a causal relationship; the semantics of causation draw on much more than simple keywords. Thus, each pattern match will need to be retained or thrown away based not just on its linguistic properties, but also on the semantics of the context and the putative arguments.

In this chapter, I suggest two related implementations of this approach for the causal construction tagging task defined in Chapter 6. After reviewing some of the information extraction work that inspired the approach (§7.1), I present **Causeway-S** and **Causeway-L**, two versions of a tagging pipeline based on **tentative pattern matching**, and compare the two approaches. Both approaches use automatically induced patterns, either **syntactic** (§7.2.1) or **lexical** (§7.2.2), to find possible lexical triggers of causal constructions and their likely arguments. The taggers then apply a mix of construction-specific classifiers and construction-independent classifiers that use linguistic, contextual, and semantic features to determine when causal constructions are truly present.

I report on three sets of experiments (§7.3.1) assessing the two systems' performance, the impacts of various design features, and the effects of parsing errors. The results indicate the viability of the approach, and point to further work needed to improve pattern-based construction recognition (§7.3.2).

The material in this chapter draws from Dunietz et al. (2017a), with all systems rerun and re-analyzed on the BECAUSE 2.1 corpus.

7.1 Related work

Despite the differences in orientation between our task and information extraction (see §6.1.1.3), the shared emphasis on broad coverage makes techniques from IE highly relevant. Most IE systems learn some form of LEXICO-SYNTACTIC PATTERNS that have been found to correlate with the relation(s) of interest, a paradigm we borrow in our pattern-based methods (Chapter 7). Relatively simple patterns were first used for tasks like hypernym discovery and event participant extraction (Hearst, 1992; Snow et al., 2004), and have since been extended and applied to many different IE tasks, including detecting causal verbs (Girju, 2003) and causation relations more generally (Ittoo and Bouma, 2011). Our pattern-based tagging approach is rooted in these lexico-syntactic patterns from IE.

IE systems have employed patterns with varying degrees of expressive power. The simplest way to represent patterns is as match sequences of fixed words, POS tags, entity type tags, and the like (see, e.g., Riloff and Jones, 1999; Muslea, 1999; Ravichandran and Hovy, 2002; Kozareva and Hovy, 2010). Such patterns can be naturally expressed as regular expressions over token sequences.

Patterns also often include aspects of sentences' dependency or constituency parse structures in their patterns. The AutoSlog (Riloff, 1996) system was one of the earliest. The more recent ReVerb system for OIE (Fader et al., 2011) looks for verbs with particular POS tag patterns, then finds pairs of noun phrases that the verb may be relating and assigns a confidence score to the extracted relation. Many IE systems (e.g., Sudo et al., 2001, 2003; Shinyama and Sekine, 2006; Alfonseca et al., 2012; Li et al., 2015), including ReVerb's predecessor TextRunner (Banko et al., 2007), allow syntax-based patterns to be more general, representing them as fragments of dependency trees with slots.

The most flexible pattern-based extractors can freely mix and match these elements. DARE (Adolphs et al., 2011), for instance, operationalizes patterns as modular, composable dependency subgraph templates, where each node and edge can have (optionally underspecified) lexical, syntactic, or semantic features. Ollie (Mausam et al., 2012) is a similar dependency-based rule learning framework that generalizes many rules from individual lemmas into broader classes of words. Probably the most flexible system to date in terms of pattern expressiveness is Odin (Valenzuela-Escárcega et al., 2016), a formalism and associated codebase for describing and matching (but not learning[1]) extraction patterns. It can specify or underspecify a pattern as an arbitrary combination of words, lemmas, POS tags, and entity tags, related by arbitrary sequential or dependency graph constraints.

On a system design level, one feature that distinguishes our work from most pattern-based IE systems is that they tend to assess patterns' reliability on the pattern level, trusting a pattern fully once it is deemed worthy. In contrast, we classify each pattern match as correct or incorrect based on additional context cues, including the contents of the putative argument spans. (See the "evaluates evidence beyond exact pattern match" row in Table 6.2.) However, this is not entirely unique to our approach; several IE systems (e.g., TextRunner; ReVerb; Alfonseca et al., 2012) likewise assign probabilistic scores to individual matches.

Many of the pattern formalisms impose much stricter constraints than we do on the types of phenomena they can pick up, as detailed in Table 6.2 and §6.1.2. Much work has focused on verb arguments in particular (e.g., Sudo et al., 2001; Fader et al., 2011), which of course represent just one of many trigger types we are interested in. Other systems also impose various restrictions on word order, word types, and so on. Those on the more complex end of the spectrum (DARE, Ollie, and Odin) are as flexible as our patterns, or even slightly more so, in terms of what types of triggers they can learn and recognize. However, as shown in the second and fourth groups in Table 6.2, they still do not allow for as full a range of argument realizations as we see in the BECAUSE corpus: they typically do not allow fragmentary arguments or arguments from multiple parse subtrees, nor do they allow overlaps between different arguments or between arguments and triggers. Some of these issues, particularly the ability to handle discontinuous spans, are addressed by the pattern-based methods discussed in this chapter. The remainder will have to wait for the still more flexible techniques of Chapter 8.

[1]Odin was designed for manual pattern specification rather than learning patterns from corpora. However, there has been at least one attempt to automatically induce patterns that are represented and matched using Odin (Valenzuela-Escárcega et al., 2016), with the goal of producing human-interpretable and even human-correctable rules.

7.2 Causeway: Causal construction tagging methods

Causeway is our umbrella name for two related systems that perform the causal language tagging task. Causeway-S is the variant based on patterns of syntactic (dependency) parses, while Causeway-L is based on patterns of word lemmas. Each technique is implemented as a pipeline with four stages:

1. **Pattern-based tentative connective discovery**. Both techniques extract lexical or lexico-syntactic patterns for connectives from the training data. These patterns are then matched against each test sentence to find tokens that **may** be participating in a causal construction.
2. **Argument identification**, which marks the Cause and Effect spans.
3. **A statistical filter** to remove false matches.
4. **A constraint-based filter** to remove redundant connectives. Smaller connectives like *to* (which is causal in sentences like *I left to get lunch*) are usually spurious when a larger connective includes the same word, like *cause* X *to* Y. When a larger and a smaller connective both make it through Stage 3 together, we remove the smaller one.

Because argument ID is done before filtering, the arguments output by Stage 2 do not quite represent the Cause and Effect arguments of a causal language instance. Rather, they represent what the Cause and Effect spans would be **if** the connective is indeed causal. For the same reason, even connective discovery is not complete until the end of the pipeline.

At training time, this tentativeness does create a slight inconvenience: some connectives tagged during connective discovery are not in fact causal, so we lack gold-standard labels for these connectives. We therefore train the argument ID stages only on instances whose connectives were correctly tagged by the first stage. This allows the training to depend on the matched patterns and the gold-standard labels.

For patterns and/or features, both versions of Causeway rely on syntactic parses from version 3.5.2 of the Stanford Parser (Klein and Manning, 2003) in the enhanced, non-collapsed dependency format. Another possible input would have been semantic role labeling (SRL) tags from an automated tagger, but SRL tags could not form the basis of our system the way syntactic relations can, because they apply only to limited classes of words (primarily verbs). Also, by examining syntactic relations and undoing passives (see §7.2.1.1), we get most of the information SRL would provide. Still, we may include SRL tags as inputs in the future.

I now describe each of the two versions of the pipeline in turn, focusing on their differing first and second stages. I also elaborate on the design of the classifier.

Throughout, the head of a span is taken to be the token that is highest in the dependency tree. For tokens that are equally high (e.g., if the parser erroneously splits a phrase into two sister subtrees), verbs or post-copular nouns are preferred over other nouns, and nouns over other parts of speech.

7.2.1 Causeway-S: Syntax-based tagging

The syntax-based approach relies on a simple intuition: each causal construction corresponds, at least in part, to a fragment of a dependency tree, where several nodes' lemmas and POS tags are fixed (see Figure 7.1). Accordingly, the first stage of Causeway-S induces lexico-syntactic patterns from the training data. At test time, it matches the patterns against new dependency trees to identify possible connectives and the putative heads of their Cause and Effect arguments. The second stage then expands these heads into complete argument spans.

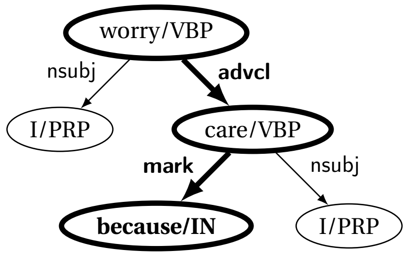


Figure 7.1: A UD parse for the sentence *I worry because I care*, with the tree fragment corresponding to the *because* construction bolded. Bolded nodes with unbolded text indicate the slots in the construction. (*Care* is a dependent of *worry*, rather than *because*, in keeping with the UD design philosophy, which maximizes dependencies between content words.)

7.2.1.1 TRegex pattern matching

For pattern matching, we use TRegex (Levy and Andrew, 2006), a grep-inspired utility for matching patterns against syntax trees. During training, the system examines each causal language instance, generating a TRegex pattern that will match tree fragments with the same connective and argument structure. In the example from Figure 7.1, the generated pattern would match any tree meeting three conditions:[2]

- some token t_1 has a child t_2 via a dependency labeled advcl
- t_2 has a child t_3 via a dependency labeled mark
- t_3 has the lemma *because* and a POS tag of IN

At test time, TRegex matches these extracted patterns against the test sentences. Continuing with the same example, it would recover t_1 as the Effect head, t_2 as the Cause head, and $\{t_3\}$ as the connective.

TRegex is designed for phrase-structure trees, so we transform each dependency tree into a PTB-like parenthetical notation (see Levin et al., 2014). Each non-leaf node corresponds to a token, represented by its lemma and index, and its first two children are its incoming dependency label and its POS tag. Its remaining children are its dependency daughters.

Syntactic patterns involving verbs can vary systematically: the verbs can become passives or verbal modifiers, which changes the UD dependency relationships. To generalize across these, we crafted a series of scripts for TSurgeon (Levy and Andrew, 2006), a tree-transformation utility built on TRegex. The scripts duplicate arguments of passive verbs and past participial modifiers into active-voice positions, ensuring that patterns will be generated and match as active voice. (Because the pattern-matching stage favors recall, not precision, we duplicate any node that **may** be a verb argument.) Each sentence is transformed before both pattern extraction and matching.

7.2.1.2 TRegex pattern extraction

The algorithm for extracting TRegex patterns first preprocesses all training sentences with TSurgeon. Next, for each causal language instance with both a Cause and an Effect, it uses the Dreyfus-Wagner algorithm (Dreyfus and Wagner, 1971) to find a minimum-weight subtree of the dependency graph that includes the connective, the Cause head, and the Effect head. The algorithm uses this subtree to build the pattern: for each subtree edge *rel* from head A to dependent B, the pattern requires that the test

[2]The actual TRegex pattern encoding these conditions is:
`/^because_[0-9]+$/=connective_0 <2 /^IN.*/ <1 mark >(/.*_[0-9]+/=cause <1 advcl >(/.*_[0-9]+/=effect)).`

$$\begin{array}{rl}
\textbf{Terminals} & \{\langle\text{Cause}\rangle, \langle\text{Effect}\rangle, \langle\text{unspecified}\rangle\} \cup \mathscr{P} \cup \mathscr{W} \cup \bigcup_{d\in\mathscr{D}} \left\{ \xleftarrow{d}, \xrightarrow{d} \right\} \\
\textbf{Start} & \langle\text{Cause}\rangle \text{ — } \langle\text{Effect}\rangle \\
\textbf{Production rules} & x \text{ — } y \Rightarrow x \xrightarrow{\text{nsubj}} y \mid x \xleftarrow{\text{nsubj}} y \mid x \xrightarrow{\text{csubj}} y \mid x \xleftarrow{\text{csubj}} y \mid x \xrightarrow{\text{prep}} y \mid \ldots \\
& x \text{ — } y \Rightarrow x \text{ — } T \text{ — } y \\
& x \Rightarrow x \text{ — } T \\
& T \Rightarrow \langle\text{lemma}\rangle \mid \langle\text{lemma}\rangle : \text{POS} \mid \langle\text{unspecified}\rangle \\
& \text{POS} \Rightarrow \text{NN} \mid \text{NNS} \mid \text{VBG} \mid \ldots \\
& \langle\text{lemma}\rangle \Rightarrow \text{a} \mid \text{apple} \mid \text{because} \mid \text{biology} \mid \text{create} \mid \ldots
\end{array}$$

Figure 7.2: A semi-formal graph grammar for patterns Causeway-S can induce. See text for details and explanation of notation.

sentence include nodes related by a dependency labeled as *rel*. If A or B is a connective word, then the pattern also checks for its lemma and POS in the test sentence nodes. Patterns with more than six non-argument, non-connective nodes are discarded as parse errors or otherwise ungeneralizable flukes.

The graph for Dreyfus-Wagner includes a directed edge for each dependency in the sentence's parse tree. For each edge, a back-edge of equal weight is added, unless the back-edge already exists (which UD allows). This allows Dreyfus-Wagner to find a subtree even when it has to follow an arc in reverse to do so.

Most edges have unit weight. However, for nodes with multiple parents (which UD also allows), the algorithm would often choose the wrong dependency path, leading to poor generalization. Accordingly, on paths of the form $x \xrightarrow{\text{xcomp}} y \xrightarrow{\text{csubj} \mid \text{nsubj}} z$ (where xcomp indicates open clausal complements), edge costs are slightly decreased. This helps the algorithm prefer the xcomp path connecting x, y, and z, rather than a path such as $x \xrightarrow{\text{nsubj}} z \xleftarrow{\text{nsubj}} y$. Similarly, acl (adjectival clause) and expl (expletive) edges are slightly penalized.

A semi-formal grammar for Causeway-S patterns To more formally describe the space of possible Causeway-S patterns, we can specify a grammar over dependency tree fragments. A fully rigorous specification would require the machinery of GRAPH GRAMMARS (see Rozenberg, 1997), a generalization of formal language theory in which production rules operate on graphs. In graph grammars, rules incrementally replace nodes, edges, or even entire subgraphs to derive a more complex graph structure. However, the dependency graphs relevant to Causeway-S are simple enough that we can suffice with a simpler, semi-formal notation borrowed from more familiar string grammars: we will allow graph edges, graph nodes, or the strings of node labels to appear freely on the left side of a rule.

A semi-formal grammar for Causeway-S patterns can be found in Figure 7.2. $\mathscr{P}$, $\mathscr{W}$, and $\mathscr{D}$ denote the sets of available POS tags, word lemmas, and dependency labels, respectively. The em dash symbol (—) indicates a dependency arc of unspecified direction with an unspecified label. The T nonterminal represents an as-yet unspecified node in the parse tree. x and y are variables representing ⟨Cause⟩, ⟨Effect⟩, or any T node. Thus, the first production rule template indicates that an unspecified arc can turn into a directed, labeled arc (the final ellipsis elides the remaining UD dependency labels). The second rule allows an unspecified edge to be interrupted with an as-yet unspecified node. The third allows the path to branch off from an existing node to an as-yet unspecified one, without modifying or

Pattern so far	**Rule(s) applied (in order of application)**
$\langle\textsf{Cause}\rangle — \langle\textsf{Effect}\rangle$	(Start)
$\langle\textsf{Cause}\rangle \xrightarrow{\textsf{advcl}} \langle\textsf{Effect}\rangle$	$x — y \Rightarrow x \xrightarrow{\textsf{nsubj}} y$
$\langle\textsf{Cause}\rangle \xrightarrow{\textsf{advcl}} \langle\textsf{Effect}\rangle — T$	$x \Rightarrow x — T$
$\langle\textsf{Cause}\rangle \xrightarrow{\textsf{advcl}} \langle\textsf{Effect}\rangle \xrightarrow{\textsf{mark}} \textit{in}:\textsf{IN}$	$x — y \Rightarrow x \xrightarrow{\textsf{mark}} y,\ T \Rightarrow \langle\text{lemma}\rangle : \text{POS},$ $\langle\text{lemma}\rangle \Rightarrow \textit{in},\ \text{POS} \Rightarrow \textsf{IN}$
$\langle\textsf{Cause}\rangle \xrightarrow{\textsf{advcl}} \langle\textsf{Effect}\rangle \xrightarrow{\textsf{mark}} \textit{in}:\textsf{IN} \xrightarrow{\textsf{mwe}} \textit{order}:\textsf{NN}$	$x — y \Rightarrow x \xrightarrow{\textsf{mwe}} y,\ x \Rightarrow x — T,\ T \Rightarrow \langle\text{lemma}\rangle : \text{POS},$ $\langle\text{lemma}\rangle \Rightarrow \textit{order},\ \text{POS} \Rightarrow \textsf{NN}$
$\begin{tikzcd}\langle\textsf{Cause}\rangle \arrow[r, "\textsf{advcl}"] & \langle\textsf{Effect}\rangle \arrow[r, "\textsf{mark}"] \arrow[dr, "\textsf{mark}"'] & \textit{in}:\textsf{IN} \arrow[r, "\textsf{mwe}"] & \textit{order}:\textsf{NN} \\ & & \textit{to}:\textsf{TO} & \end{tikzcd}$	$x — y \Rightarrow x \xrightarrow{\textsf{mwe}} y,\ x \Rightarrow x — T,\ T \Rightarrow \langle\text{lemma}\rangle : \text{POS},$ $\langle\text{lemma}\rangle \Rightarrow \textit{to},\ \text{POS} \Rightarrow \textsf{TO}$

Table 7.1: Deriving a Causeway-S pattern for the *in order to* construction. Some steps are collapsed for brevity.

interrupting any existing paths in the parse fragment. The ellipses in the last two rules represent the remaining POS tags and the full set of possible English lemmas. (Note that in practice, POS labels are only added for connective nodes.)

Table 7.1 shows how the pattern for the connective *in order to* would be derived using this grammar. Note that this is entirely a theoretical construction to demonstrate the space of possible patterns, not the actual mechanism by which patterns are produced.

7.2.1.3 Syntax-based argument identification

Each syntactic pattern inherently encodes the positions in the tree of the Cause and Effect heads. Thus, matching these patterns is the first step of argument ID, in addition to (tentative) connective discovery.

The second step of argument ID is to expand the argument heads into complete spans. In general, most syntactic dependents of an argument's head are included in its span. There are two exceptions:

1. **Connective words.** Under the UD scheme, words that form part of the connective sometimes appear as dependents of the argument head. For example, in *A prevents B from C*, *from* appears as a dependent of *C*, but it is really part of the construction for the verb *prevent.* Following the annotation scheme, we therefore exclude such connective words (and any of their dependents) from the argument span.
2. **Words below the head of the other argument.** For example, in *A because B*, *B* will be a dependent of *A* (see Figure 7.1). Obviously, however, *B* should not be included in the Effect span.

The full algorithm for expanding argument heads into spans is given in Algorithm 1.

7.2.2 Causeway-L: Lexical pattern-based tagging

Syntactic parsers often make mistakes, which becomes especially relevant for syntactic patterns that examine multiple dependencies. This is particularly problematic for the syntax-based pipeline, for which parse errors, either in training or at test time, can prevent patterns from matching. Additionally, the exact syntactic relations present in a given instance may be altered by the presence of other constructions. For example, if a verb appears as a complement of another verb, the path to the subject will have an additional dependency link.

```
procedure ExpandArgumentToken(token, arg_head, visited, connective, other_arg)
    expanded ← {token}
    for edge_label, child in GetChildren(token) do
        if $child \notin visited$ and ($token \neq arg\_head$ or $edge\_label \notin$ {conj, cc, parataxis}) then
            visited ← visited ∪ {child}
            if $child \neq other\_arg$ and ($child \notin connective$ or child's lemma was seen in training
              within an argument for connective) then
                e ← ExpandArgumentToken(child, arg_head, visited, connective, other_arg)
                expanded ← expanded ∪ e
            end if
        end if
    end for
end procedure
```

Algorithm 1: Algorithm for expanding argument heads into full argument spans. *other_arg* is the head token of the other argument.

$$\begin{aligned}
\langle \textit{Pattern} \rangle &\Rightarrow \langle \textit{TokenStart} \rangle \ \big(\langle \textit{ArgStart} \rangle \ \texttt{|} \ \langle \textit{ConnStart} \rangle\big) \\
\langle \textit{ArgStart} \rangle &\Rightarrow \langle \textit{PossibleArg} \rangle \ \langle \textit{ConnWord} \rangle\texttt{+} \ \big(\langle \textit{PossibleArg} \rangle \ \langle \textit{ConnWord} \rangle\texttt{+}\big)* \\
\langle \textit{ConnStart} \rangle &\Rightarrow \langle \textit{ConnWord} \rangle\texttt{+} \ \langle \textit{PossibleArg} \rangle \ \big(\langle \textit{ConnWord} \rangle\texttt{+} \ \langle \textit{PossibleArg} \rangle\big)* \\
\langle \textit{TokenStart} \rangle &\Rightarrow \text{‘}\texttt{(\^{}|␣)}\text{’} \\
\langle \textit{PossibleArg} \rangle &\Rightarrow \text{‘}\texttt{([\textbackslash S]+␣)+?}\text{’} \\
\langle \textit{ConnWord} \rangle &\Rightarrow \text{‘}\texttt{(}\text{’} \ \langle \textit{Lemma} \rangle \ \text{‘}\texttt{/}\text{’} \ \langle \textit{POS} \rangle \ \text{‘}\texttt{)␣}\text{’}
\end{aligned}$$

Figure 7.3: A grammar over Causeway-L regular expression patterns.

.

We therefore implemented a second algorithm, Causeway-L, that performs connective discovery based on the sequence of word lemmas and parts of speech: instead of extracting and matching parse patterns, it extracts and matches regular expressions. It then uses a conditional random field to label the argument spans, using features from the parse in a more probabilistic way.

7.2.2.1 Connective discovery with regular expression patterns

At training time, we generate regular expressions that will match sequences of connective and argument lemmas. The regexes also make sure that there are tokens in the correct lexical positions for arguments. For instance, upon seeing an example like *A because B*, it would generate a pattern that matches an optional sequence of lemmas (the Effect range), followed by the lemma *because* with POS tag IN (the connective), followed by any other optional sequence of lemmas (the Cause range).[3] Each subpattern is given its own capturing group in the regular expression. At test time, each new sentence is turned into a string of lemmas with POS tags for matching. Matching lemmas can be recovered from the capturing groups.

[3]The actual regular expression that encodes this is: (^|)([\S]+)+?(because/IN) ([\S]+)+?.

- The lemma of w_i
- The POS tag of w_i
- Whether $w_i \in C$
- The dependency parse path between w_i and the token in C that is closest in the parse tree
- The absolute lexical distance between w_i and the lexically closest token in C
- The signed lexical distance between w_i and the lexically closest token in C
- Whether w_i is in the parse tree (false for punctuation and several other non-argument token types)
- The regex pattern that matched C
- The cross-product of the regex pattern feature and the parse path feature
- The position of w_i relative to C (before, overlapping, or after)
- Whether the lemma of w_i is alphanumeric

Table 7.2: Features used for CRF argument ID. For each possible connective C (a set of tokens), features are extracted from each word w_i in the sentence. All non-numeric features are binarized.

A grammar for Causeway-L patterns Since lexical patterns are actually just regular expressions, it is much easier to define a formalized grammar that describes what patterns are possible. The regular grammar in Figure 7.3 spells out the space of connective-matching regular expressions that can be generated by Causeway-L. Literals are given in `typewriter` font between 'single quotes,' with visible␣spaces. ⟨*Lemma*⟩ and ⟨*POS*⟩ of course expand to any lemma and any POS tag, respectively. The induced patterns are matched over space-separated, space-terminated sequences of tokens, where each token is of the form *lemma*/POS.

7.2.2.2 Argument identification with a CRF

Unlike syntactic patterns, regular expressions cannot pinpoint the Cause and Effect arguments; they can only give ranges within which those arguments must appear. Thus, the argument ID stage for this pipeline starts with much less information.

We treat the task of argument ID as a sequence labeling problem: given a particular regex-derived connective, the system must label each token as part of the Cause, part of the Effect, or neither. For this task we use a LINEAR-CHAIN CONDITIONAL RANDOM FIELD (LCCRF), which models the probability of each label in a sequence as a function of the label value, the previous label, and all the inputs in the sequence. More precisely, an LCCRF models a sequence of labels $\mathbf{y}$ for a sequence of inputs $\mathbf{x}$ as :

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t) \right\},$$

where each f_k is a feature function, θ_k is a weight for feature f_k, t represents a timestep in the sequence, and Z is a normalizing factor. We use standard implementations for CRF training and prediction from the CRFsuite library.[4]

The features for the argument ID CRF are listed in Table 7.2.

7.2.3 Voting classifier for filtering

[4]`http://www.chokkan.org/software/crfsuite/`. We use the python-crfsuite wrapper (`https://github.com/tpeng/python-crfsuite/`).

[5]For this purpose, we represent the tense of a verb as the POS of the verb plus the string of auxiliaries attached to it. The tense of a non-verb is null. For copulas, both the POS of the copula and the POS of its object are included.

Connective features:
- The label on the dependency from h to its parent
- The part of speech of h's parent
- The sequence of connective words*
- The sequence of connective lemmas*
- Each pattern that matched the connective*

Argument features:
- The POS tags of c and e
- The generalized POS tags of c and e (e.g., N for either NNP or NNS)
- The tenses[5] of c and e, both alone and conjoined
- The label on each child dependency of c and e
- For verbs, the set of child dependency labels
- The number of words between c and e
- The dependency path between c and e
- The length of the dependency path from c to e
- Each closed-class child lemma of e and of c
- The domination relationship between c and e (dominates, dominated by, or independent)
- The sets of closed-class child lemmas of e and c
- The conjoined NER tags of c and e
- Initial prepositions of the Cause/Effect spans, if any
- Each POS 1-skip-2-gram in the Cause/Effect spans
- Each lemma 1-skip-2-gram in the Cause/Effect spans that was seen at least 4 times in training
- Each WordNet hypernym of c and e†

Table 7.3: Features for the causal language candidate filter. c indicates the Cause head, e the Effect head, and h the connective head. All non-numeric features are binarized.
* Used only for per-connective classifiers.
† Used only for the global classifier.

The pattern-matching stage overgenerates for two reasons. First, due to ambiguity of both words and constructions, not all instances of a given pattern are actually causal. *Since*, for example, has both causal and temporal senses, which are not distinguished either by surface position or by their dependency tree configurations. Second, the patterns do not filter for important constructional elements like tense and modality (e.g., example 8 in Table 1.1 requires modality of necessity in the Cause). Thus, pattern matching alone would yield high recall but low precision.

To account for this, the final stage of both pipelines is a filter that determines whether each possible connective instance, along with its arguments, is in fact being used in a causal construction. The machine learning device we use for this is the logistic regression classifier, which takes a feature vector **x** and assigns it the following probability for its binary label y:

$$p\left(y = \text{true} \mid \mathbf{x}\right) = \frac{1}{1 + \exp\left\{-\left(\theta_0 + \boldsymbol{\theta}^\top \mathbf{x}\right)\right\}},$$

where θ_0 and $\boldsymbol{\theta}$ are the model parameters. A final label is assigned by comparing the probability estimate to some threshold.

Our classification task is somewhat, but not entirely, heterogeneous. Some aspects are universal—there are regularities in what typically causes what—while others are construction-dependent. To incorporate both kinds of information, a separate SOFT-VOTING classifier is created for each unique sequence of connective words. Each such classifier averages the probability estimates of three less-reliable classifiers: a global logistic regression classifier, which is shared between all connectives; a per-connective logistic regression classifier; and a per-connective classifier that chooses the most frequent label for that connective.[6] Our classifier thus differs slightly from typical voting classifiers: rather than the ensemble consisting of multiple algorithms trained on the same data, our ensemble includes one generalist and two specialists.

We use the `scikit-learn` 0.17.1 (Pedregosa et al., 2011) implementation of logistic regression with L_1 regularization and balanced class weights. The logistic regression classifiers consider a variety of features (see Table 7.3) derived from the matched pattern, the connective words matching, the argument heads, and the parse tree. ("1-skip-2-gram" refers to a trigram missing its second word—i.e., a bigram that skips 1 word.) An instance is tagged as causal if the soft vote assigns it a probability above 0.45. This cutoff, close to the prior of 0.5, was tuned for F_1 on a different random split of the data than was used in the experiments below. In our experiments, the cutoff made little difference to scores.

7.3 Experiments

7.3.1 Experimental design

The methods above were implemented within the NLPypline framework for NLP pipelines.[7] The full Causeway codebase is available on GitHub.[8]

See §6.3 for a description of the simple benchmark system against which we compare Causeway-S and Causeway-L, and §6.4 for a description of our evaluation metrics. As noted in §6.4, because Causeway is designed only to find instances with both a Cause and an Effect, we evaluate it only on such instances.

In addition to an end-to-end evaluation, we wished to test how helpful our three-way voting classifier is. For each pipeline, then, we also compare that classifier against a simple most-frequent-sense baseline that chooses the most frequent label (causal or not) for each connective, with connectives differentiated by their lemma sequences.

7.3.1.1 Experiment 1: Pipeline comparison

In this experiment, we measured the performance of Causeway-S, Causeway-L, and the benchmark system on the tasks of connective discovery and argument ID. We also tried taking the union of each system's outputs with the benchmark system's. Because of the small size of BECAUSE, we report averaged metrics from 20-fold cross-validation, with fold size measured by sentence count. All pipelines were run on the same folds.

7.3.1.2 Experiment 2: Ablation studies

Our second experiment explores the impact of various design choices by eliminating individual design elements. We report results from using the global and connective-specific classifiers on their own, without the soft-voting ensemble. (The MFS classifier is tested alone as part of Experiment 1.) We also report results from the ensemble classifier without using any features that primarily reflect world knowledge: NER tags, WordNet hypernyms, and lemma skip-grams. To the extent that these features

[6]We explored a weighted average, with the weights learned automatically for each connective, but our dataset was not large enough for this to be effective.

[7]`https://github.com/duncanka/NLPypline`

[8]`https://github.com/duncanka/causeway`

	Connectives			Causes			Effects		
Pipeline	**P**	**R**	**F_1**	**F_1**	**F_1@.5**	**J**	**F_1**	**F_1@.5**	**J**
Causeway-S w/o filter	6.3	*72.3*	11.5	8.2	9.2	81.8	4.0	7.0	64.1
Causeway-S + MFS	59.0	37.7	45.8	33.6	38.1	84.5	22.1	34.4	75.5
Causeway-S + MFS + SC filter	*62.2*	37.7	46.7	34.4	38.9	84.3	22.5	35.2	75.7
Causeway-S + cls	55.5	46.0	50.1	36.1	40.6	82.1	23.0	36.4	73.2
Causeway-S + cls + SC filter	59.6	45.9	*51.7*	*37.3*	*42.0*	82.3	*23.7*	*37.8*	73.5
Causeway-L w/o filter	7.2	***92.7***	13.4	7.8	9.0	68.0	5.9	8.4	64.0
Causeway-L + MFS	62.4	35.2	44.6	32.7	37.5	84.0	24.4	32.7	75.8
Causeway-L + MFS + SC filter	*64.3*	34.9	44.8	33.0	37.5	84.2	24.5	32.7	75.7
Causeway-L + classifier	58.9	45.0	50.7	*38.3*	*43.0*	85.2	*29.8*	39.7	78.4
Causeway-L + cls + SC filter	60.4	44.2	*50.8*	38.2	*43.0*	85.1	*29.8*	*39.8*	78.5
Benchmark	***84.1***	19.7	31.6	24.0	26.7	86.6	16.4	23.8	77.6
Benchmark + Causeway-S	61.3	*50.0*	***54.9***	39.9	44.7	82.8	25.5	40.2	73.8
Benchmark + Causeway-L	62.2	49.0	54.6	***40.8***	***45.9***	85.1	***31.6***	***42.5***	78.5

Table 7.4: Results for Experiment 1. F_1@.5 indicates F_1 score for partial match of at least 50% of tokens. J indicates the average Jaccard index (given a correct connective). "SC filter" indicates the filter for smaller overlapping connectives. For the combinations of the benchmark with Causeway, the union of the benchmark's and Causeway's outputs was passed to the SC filter. The best scores within each group are italicized, and the best scores in each column are also bolded (except for Jaccard indices, which are not directly comparable between pipelines).

correlate with causality, it is presumably because of regularities in what types of real-world objects are more or less likely to be Causes and Effects.

7.3.1.3 Experiment 3: Effects of parse errors

In our third experiment, we examined the effects of parse errors on our pipelines' performance. We compared each pipeline's performance with and without gold-standard parses, using only the Penn Treebank portion of BECAUSE. For gold-standard runs, we used the Stanford dependency converter (De Marneffe et al., 2006) to convert the gold parses into dependency format. We report averaged results from 20-fold cross-validation on this subcorpus.

7.3.2 Experimental results and analysis

7.3.2.1 Experiment 1 results

Results from Experiment 1 are shown in Table 7.4.

Connective discovery Our most important conclusion from these results is that a classifier can indeed learn to recognize many of the subtleties that distinguish causal constructions from their similar non-causal counterparts. Even our end-to-end benchmark offers moderate performance, but Causeway-L outperforms it at connective discovery by over 19 F_1 points, and Causeway-S outperforms it by over 20 points.

Causeway-S bests Causeway-L by a slim margin, but contrary to our expectations, the difference is far from statistically significant ($p < 0.51$, paired t-test over folds). This does not support our initial hypothesis that parse information would be essential for identifying constructions of causal language. Of course, Causeway-L does still have access to much of the parse information in the form of syntax-related classifier features, but even with those features removed its F_1 remains nearly as high.

The design of the filter is a significant contributor to Causeway's dominance over the benchmark. The MFS classifier alone substantially underperforms the voting classifier. The small connectives filter makes up a bit of the gap, but the full pipeline still beats the MFS filter by 5 points for the syntax-based system and 6 points for the lexical system.

When our pipelines are combined with the end-to-end benchmark, the results are better still, beating the benchmark alone by a resounding 25 points. This supports our hypothesis that causal construction recognition rests on a combination of both shallow and deep information.

As expected, both pipelines show high recall but low precision for the connective discovery stage. (Much of the remaining gap in recall came simply from the long tail of constructions—about half of connective types never saw a pattern match.) The filter does balance out precision and recall for a better F_1. However, as the filter's steep drop in recall suggests, more work is needed to upweight positive instances. Examining the classifier scores reveals that the filter is doing a good job of assigning low probability to negative instances: the vast majority of false pattern matches are clustered below a probability of 0.5, whereas the positives are peppered more evenly over the probability spectrum. Unfortunately, the positives' probabilities are not clustered on the high end, as they should be.

Significant leverage could be gained just from improving classification for the connective *to*. For both pipelines, this one connective accounted for 15–18% of end-to-end false positives and false negatives, and a third of all misclassifications by the filter. Many of the remaining errors (about 45%) came from just a few common, highly ambiguous/polysemous connectives, including *for*, *when*, *allow to*, *if*, and *so*.

On most simple words, though, the classifier did fairly well: for verbs, for example, F_1 jumped from 23% to 52% in Causeway-S after the filter, and from 8.9% to 48% in Causeway-L. Similarly, for subordinating conjunctions it increased from 24–26% to 72–76%. For complex constructions (MWEs or more complex syntactic structures), the category of most interest for our purposes, Causeway achieved 35–37% F_1 compared to 2–3% before the filter. The only POS types for which the classifier did not produce a dramatic jump in F_1 were those with very few training examples: nouns and adjectives. Overall, then, it seems that the classifier helps even with the cases that would challenge typical semantic parsing systems, but it needs some features to help it to upweight positive instances of a few challenging words. More data would not hurt, either.

Argument identification Both versions of Causeway do reasonably well at recovering exact argument spans, given the precision and recall limitations imposed by the first stage (they cannot correctly tag arguments for erroneous connectives). Even when the exact argument spans do not match, both pipelines achieve high Jaccard indices,[9] suggesting that both sequence tagging and head expansion do a good job of identifying argument words even when the full argument spans do not match perfectly.

Effects seem to be harder to recover than Causes. We initially hypothesized that this was because of situations where the Effect really has two distinct heads, such as ditransitive causative verbs. For example, in *I made him apologize*, *him* and *apologize* are two distinct subtrees that are both part of the Effect argument. However, less than 8% of gold-standard instances have connectives that can take two heads, which belies this explanation.

One likely contributor is the difference in lengths between the two types of arguments. The distribution of Cause lengths is skewed toward low numbers, with a peak at 1 and a median of 5, while Effects have a less sharp peak at 5 and a median of 7 (Figure 7.4). The difference makes it harder for the system

[9]Earlier descriptions of this work (Dunietz et al., 2017a,c) reported much lower Jaccard indices on the BECAUSE 1.0 corpus. These were due to a bug in the evaluation code.

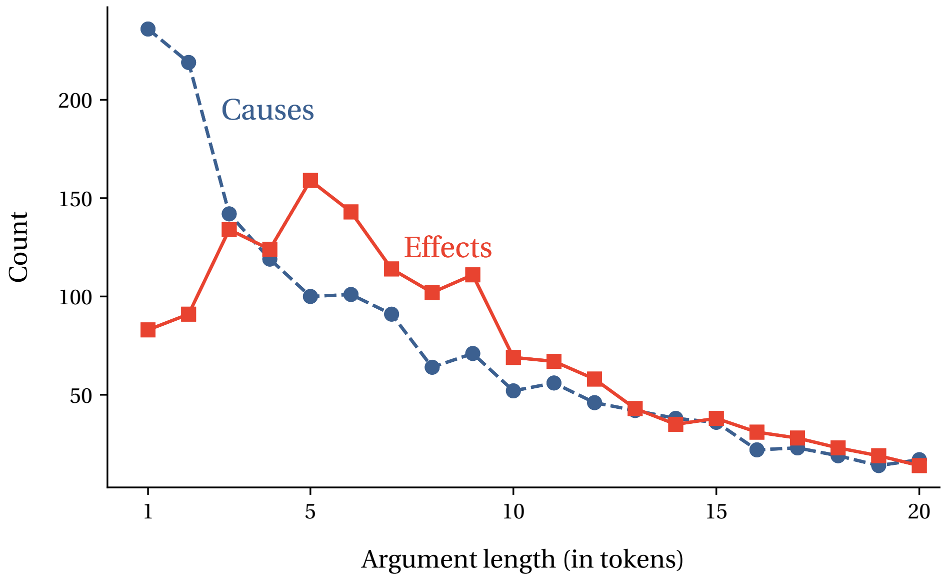


Figure 7.4: Distributions of Cause and Effect span lengths in the BECAUSE corpus (for instances with both arguments).

to guess full Effect spans. That explains why Causeway-L does significantly better on exact spans for Effects despite being close in span overlap: expanding the syntactic head is likely to get a few words wrong when the span is long. The length disparity, in turn, is probably due to the fact that Causes are likely to be subjects (19%) or nominal modifiers (17%), which skew short and tend to be simple noun phrases, whereas most Effects are main clauses (28%), clausal complements (30%), or direct objects (11%), which are often longer and more internally complex.

Still, length does not explain the entire disparity in argument performance; for any given length, both systems perform worse for Effects than for Causes. The disparity is weaker for the lexical pipeline, though, which therefore dominates the others on F_1 (and Jaccard index) for Effects. We leave further exploration of why pattern-based methods struggle more with Effects to future research.

7.3.2.2 Experiment 2 results

Results for Experiment 2 are shown in Table 7.5.

The results using single classifiers, rather than an ensemble, uphold our design of combining multiple information sources. Even the best non-ensemble filter, the global-only filter for Causeway-S (i.e., both per-connective filters ablated), underperforms its ensemble counterpart by 2 points.

The effects of world knowledge features are more ambiguous.[10] For both pipelines, removing these features hurts precision but helps recall, ultimately producing a nearly identical F_1. It appears that world knowledge features establish correlations about what causes what, causing the classifier to be more conservative about whether two things can be in a cause/effect relationship. That reduces false positives, but at the expense of adding some false negatives. In other words, world knowledge features are a lever to slightly bias the system toward recall or precision.

[10]On BECAUSE 1.0, world knowledge appeared to improve classification (Dunietz et al., 2017a), though the apparent improvement may have been an anomaly.

		Connectives			Causes			Effects		
Pipeline	**Classifiers Ablated**	**P**	**R**	**F_1**	**F_1**	**F_1@.5**	**J**	**F_1**	**F_1@.5**	**J**
Causeway-S	–	59.6	45.9	51.7	37.3	42.0	82.3	23.7	37.8	73.5
Causeway-S	Both per-conn.	42.9	49.2	45.7	32.4	36.8	81.5	20.0	32.9	71.9
Causeway-S	Global/most-freq.	48.2	47.9	47.8	33.9	38.2	81.0	20.6	33.8	71.2
Causeway-S	Knowledge feat.s	57.2	46.5	51.0	36.9	41.7	82.1	22.9	37.2	72.3
Causeway-L	–	60.4	44.2	50.8	38.2	43.0	85.1	29.8	39.8	78.5
Causeway-L	Both per-conn.	43.0	44.0	43.2	32.6	36.7	85.2	24.9	34.0	78.1
Causeway-L	Global/most-freq.	49.0	43.8	46.0	34.2	39.0	85.3	25.4	35.5	77.3
Causeway-L	Knowledge feat.s	57.2	46.5	51.0	38.4	43.3	85.1	29.1	39.8	78.2

Table 7.5: Results for Experiment 2. Unablated full-pipeline results from Table 7.4 are included for comparison.

	Connectives			Causes			Effects		
Pipeline	**P**	**R**	**F_1**	**F_1**	**F_1@.5**	**J**	**F_1**	**F_1@.5**	**J**
Automatic parses; no filtering	12.0	64.3	20.1	14.8	16.7	84.9	9.4	14.2	73.0
Automatic parses; cls + SC filter	64.0	44.5	51.8	40.0	44.3	86.8	26.2	38.3	75.7
Gold parses; no filtering	10.5	69.8	18.2	14.5	16.5	90.3	10.9	15.5	83.4
Gold parses; cls + SC filter	63.9	51.5	56.6	45.8	51.1	89.8	36.1	50.3	86.8

(a) Causeway-S

	Connectives			Causes			Effects		
Pipeline	**P**	**R**	**F_1**	**F_1**	**F_1@.5**	**J**	**F_1**	**F_1@.5**	**J**
Automatic parses; no filter	9.4	86.4	16.8	10.6	11.6	69.2	7.5	11.0	65.3
Automatic parses; cls + SC filter	60.7	43.2	50.0	39.7	42.4	86.4	28.9	40.9	78.8
Gold parses; no filter	9.3	87.0	16.7	10.9	12.1	72.9	7.7	11.0	65.8
Gold parses; cls + SC filter	65.9	45.4	53.2	43.7	46.4	89.0	31.1	43.3	81.1

(b) Causeway-L

Table 7.6: Results for Experiment 3.

7.3.2.3 Experiment 3 results

Results from Experiment 3 are shown in Table 7.6.

As expected, Causeway-S improved significantly with gold-standard parses, whereas Causeway-L gets a smaller boost. The improvements did not come only from better matching of connective patterns: recall is better with gold parses, but both precision and F_1 for connective matching from the first stage were the same or worse. Instead, a significant portion of the improvement appears to come from argument identification: better parses made it easier to identify argument heads, which in turn made the many features based on those heads more reliable. This is supported by the fact that when we ran the dependency-based benchmark on the PTB subcorpus with and without gold parses, we saw a similar improvement. In fact, with gold-standard parses on the PTB subcorpus, the benchmark is on par with Causeway-L (53.4% F_1, statistically indistinguishable at $p < 0.84$).

Thus, although the limited data and the classifier's failure to upweight positives are still the primary handicaps, better parses would be somewhat helpful for at least the syntax-based approach, and simpler techniques could be quite effective if parse errors were not at all an issue.

7.4 Conclusion

With this work, we have demonstrated the viability of tagging a wide variety of causal constructions. Both versions of our pattern-based pipeline, Causeway-S and Causeway-L, significantly outperform our simple benchmark. We did not see as much of a difference as we expected (even hoped) between the two, though, suggesting that parse information may be less critical than we thought. Our experiments do uphold our pipeline design choices, with the possible exception of world knowledge features: information about tentative arguments helps inform decisions about which causal language instances to retain, and an ensemble of specialist and generalist classifiers is important to achieving good accuracy.

In short, then, pattern-based methods work—but they still leave plenty of room for improvement.

CHAPTER 8

Deep Learning-Based Tagging Methods

8.1 Why deep learning and transition-based tagging?

The pattern-based methods of Causeway discussed in Chapter 7 offer substantial coverage for constructions of cause and effect. Still, the system design has several drawbacks, most importantly the preciseness of the patterns. Causeway avoids some of the brittleness of the handcrafted, rule-based AI systems of yore by inferring lexical and lexico-syntactic patterns from the corpus, but the patterns are still constrained to exactly match the configurations seen in training. If an instance uses a causal construction never seen in training or a synonym of a known connective word, the system stands no chance of tagging it. Likewise, parsing or POS tagging errors can throw off the pattern-matching unless the same errors appear in both training and test data.

Even Causeway's statistical learning mechanisms suffer from similar issues. The system uses a carefully calibrated ensemble of classifiers which depends on extracting hand-engineered features from each potential instance of causal language (see Table 7.3). Improving Causeway relies on re-architecting the ensemble or improving the feature set, and adapting the system to a new task would require starting almost from scratch on feature engineering.

A more flexible approach is that of DEEP NEURAL NETWORKS (DNNs) and DEEP LEARNING. In principle, deep learning is particularly well-suited to tasks of construction recognition. Constructions are often characterized by subtle combinations of word forms, word meanings, and context. These characteristics are hard to pin down explicitly—e.g., to distinguish the different uses of *given* in *Given his stature, he can go overtime* (causal) and *Given a layout, we can adjust to either font size* (not causal), one must incorporate cues from the noun phrases' definiteness with knowledge about human social interactions, among other factors. It can be hard even to decompose such characteristics into the features that traditional machine learning systems rely on. Furthermore, constructions rely on linguistic phenomena at multiple scales, from prefixes to syntactic configurations to discourse contexts. Such tasks are precisely where deep learning excels: it can mine the low-level training data to extract patterns and patterns of patterns to attend to.

The challenge, however, is that a neural network typically has a fairly rigid output structure—e.g., one label for each word. But for constructions such as causal language, the output we need is more complex: each sentence must be tagged with up to several causal language instances, each with its own connective and argument spans. A connective span may overlap with its arguments, and a connective or argument span may overlap with spans from other instances.

To produce such rich structures from a single neural network, this chapter presents a TRANSITION-BASED technique. A transition-based tagger operates much like robot assembling a tower of Tinker Toys using a predefined set of operations: adding a strut, rotating a joint, joining two boxes together, etc. Similarly, a transition-based tagger defines a set of TRANSITIONS, also called ACTIONS, which can be performed in sequence to construct the desired output. At each step, the tagger must select the next transition, terminating when it has produced the final output structure.

In the remainder of this chapter, I present **DeepCx**, a neural, transition-based construction tagger for causal language. I describe a **transition scheme** that allows reconstructing instances of causal language from the BECAUSE corpus (§8.3). I then describe a DNN, specifically a **LONG SHORT-TERM MEMORY (LSTM) network**, that applies the transition scheme to tag causal constructions (§8.4). I describe (§8.5) and analyze (§8.6) several **experiments** testing this tagger, and suggest how an approach like DeepCx could be applied to **other kinds of constructions** (§8.8). But before discussing the details of DeepCx, it will be helpful to review its conceptual lineage.

8.2 Background and related work

This section covers the essentials of the background literature on neural networks and transition-based systems for semantic parsing, with an eye toward what is necessary to understand the workings of DeepCx. See §A.3 for a review of deep learning in general and LSTMs. For a more complete survey of deep learning methods in NLP, see Goldberg and Hirst (2017).

8.2.1 Non-transition-based related work using neural networks

A variety of non-transition-based neural networks have been employed for semantic parsing.

Several systems performed PropBank-style semantic role labeling using a sequence tagging approach: for each verb, each word is labeled with its argument status (None, Arg0, Arg1, etc.), typically based on information from a fixed context window around the word. This approach was pioneered by SENNA (Collobert et al., 2011), which computed embedding and feature vectors for a fixed window around each word, then combined those vectors into one fixed-size vector representing the entire sentence. The window vectors were merged by taking the max over all windows for each vector position, and the result was then passed to a conventional classifier network. For each verb, this process was repeated for every word in the sentence to produce a tag for that word. The Daisy system (Foland and Martin, 2015) extended SENNA to include additional features based on dependency parses and to handle nominal predicates. Zhou and Xu (2015) took a similar approach, but using a BIDIRECTIONAL LSTM (Graves and Schmidhuber, 2005)—a pair of LSTMs, one reading the sentence forward and one reading it backward—to encode information from the entire sentence rather than a context window. (They did still use a context window as one of the features for each token's vector.)

DeepCx resembles these systems in that it scans through all possible argument words for each connective. But through the technique of transitions, it unifies this procedure with the process of identifying connectives, including fragments of connectives that are interspersed with arguments. Also, as noted in §6.1.1.1, the PropBank taggers take it as a given that PropBank predicates are easy to identify; the taggers are saddled only with argument identification, which is easily construed as a sequence tagging task. DeepCx, on the other hand, must perform the more difficult task of connective discovery, as well.

Other groups have taken a neural tack to tagging arguments as a whole, rather than tagging word by word. The role of the neural network has varied: Täckström et al. (2015); FitzGerald et al. (2015), who tested their system on both FrameNet and PropBank, use an ANN to score possible argument slot fillers, where each potential argument span is represented by a hand-engineered feature vector. The best overall assignment of slot fillers to slots is then computed with a constrained optimization algorithm. The PathLSTM system, which also has been applied to both PropBank (Roth and Lapata, 2016) and FrameNet (Roth, 2016), tags possible argument head words, then reranks candidate arguments to find the single best assignment of argument labels. For classifying arguments, it uses four distinct neural networks—one for each permutation of predicate type (verbal/nominal) and stage of argument identification (finding argument spans/labeling their argument classes). Finally, open-SESAME (Swayamdipta et al., 2017) takes a segmenting approach to argument identification for FrameNet: its "semi-Markov" model scores entire spans, using a bidirectional LSTM module to embed each contextualized token, another to embed each span of tokens, and LSTM-based embeddings for the contextualized trigger and pre-labeled frame.

These systems differ from ours in four key ways. First, they are again limited to argument identification; even when tagging for FrameNet, they assume another model or an oracle has provided the triggers and predicate labels. Second, their neural networks depend on decomposing contextualized words and spans into manually defined features. Third, whereas we focus on causal connectives, they aim to tag all predicates in either the PropBank lexicon or the FrameNet lexicon (or both). This gives them the broad propositional coverage but also the restrictions on linguistic triggers discussed in §3.3.1. Finally, we rely on a transition-based tagger rather than span- or head-oriented classifiers or scorers.

In addition to its overall system architecture, Roth and Lapata (2016) introduce an important concept that we borrow in DeepCx: the idea of DEPENDENCY PATH embeddings. A dependency path is simply a path through the sentence's dependency parse tree from one token to another. Symbolic dependency paths have long been used in semantic analysis tasks, but PathLSTM produces LSTM-based embeddings of the dependency paths between a predicate and a possible argument word as part of the representation for the latter. We include a similar path embedding in DeepCx's state representation; see §8.4.2.3.

8.2.2 Transition-based systems

The second pillar on which DeepCx rests, transition-based parsing, traces back to SHIFT-REDUCE PARSERS for context-free grammars. Most commonly used for parsing programming languages (see Aho et al., 1986), a shift-reduce parser parses a document one token at a time, incrementally merging each one into the collection of parse tree fragments the parser has built up. Because shift-reduce parsers process each token only once, never backtracking, they are very efficient at building complex data structures from the bottom up.

In the context of natural languages (as opposed to programming languages), shift-reduce parsers have primarily been used for dependency parsing, starting with Yamada and Matsumoto (2003) and Nivre (2003). (Earlier researchers proposed similar approaches for other forms of linguistic analysis; see, e.g., Ratnaparkhi, 1997 and Briscoe and Carroll, 1993.) Shift-reduce dependency parsers break the problem of syntactic parsing down into a series of incremental decisions, each deciding between two basic types of actions. A SHIFT action reads the next word from the sentence and adds it to a STACK data structure. A REDUCE action merges the two most recent words or subtrees on the stack into a single tree fragment by adding a dependency relation between them.[1] (Of course, there are separate actions for each dependency direction—i.e., for making the top of the stack a dependent of its predecessor and vice versa.) The formal task facing a shift-reduce dependency parser is then to examine the state of the stack, the remaining words to be processed, and any other relevant information at each step, and decide which action to take.

[1]The variant of shift-reduce dependency parsing described here is what Nivre calls ARC-STANDARD. See Nivre (2008) for other common variants.

Because of its speed—linear time in the number of words in the sentence—the shift-reduce approach has become extremely popular for dependency parsing. A huge variety of methods have been proposed for deciding between actions (see, e.g., Nivre et al., 2007; Nivre, 2008; Zhang and Nivre, 2011; Chen et al., 2014), as well as alternative transition schemes (e.g., Nivre, 2008; Choi and Palmer, 2011a; Qi and Manning, 2017) and mechanisms for overcoming the greediness of typical transition-based approaches (e.g., Choi and Mccallum, 2013; Johansson and Nugues, 2007a).

Occasionally, transition-based systems have been used for NLP tasks beyond syntactic parsing, as well. Titov et al. (2009) and Henderson et al. (2008) explored extensions of transition-based dependency parsing that interleave actions for semantic parsing with the traditional actions for syntactic parsing. More directly relevant to DeepCx is Choi and Palmer's (2011b) work, which defines a novel transition scheme designed expressly for semantic parsing in the PropBank scheme (specifically tagging verb arguments assuming oracle predicates). We build on this more specialized scheme with our similar one for parsing causal constructions. The key differences in our transition scheme are that it is designed to find triggers as well as arguments and that it assumes that triggers may be complex constructions. See §8.3.3 for further elaboration on these differences.

8.2.3 Neural transition-based systems

Given RNNs' power as a machine learning paradigm and transition-based systems' versatility with complex output structures, it is only natural that NLP systems would begin to use RNN-driven transition-based taggers. Chen and Manning (2014) were among the first to develop a neural transition-based syntactic parser, and an updated version of that Stanford Parser model is still in widespread use. More recently, Google released its highly accurate SyntaxNet parser (Andor et al., 2016), whose English version was released under the moniker Parsey McParseface.

These models extract as features vector representations of the top k words on the stack and buffer, their POS tags, and their already-parsed dependency children. Ideally, however, parsing behavior would depend on the entire contents of the stack and buffer, not just a fixed-size subset of it. Dyer et al. (2015) enabled embedding the full stack and buffer by introducing an LSTM enhancement, the STACK LSTM, that can efficiently embed a dynamically changing list. The idea is to augment an LSTM with a TOP pointer that indicates which item t from the LSTM's history is currently on the top of the stack, i.e., the last in the list. When an LSTM input i is added via a push operation, the system moves TOP to point to i, but it also adds a back-pointer from i to t, the previous TOP item. If i is later removed with a pop operation, the back-pointer is followed to rewind the TOP pointer to t (which of course has its own backpointer to its predecessor, in case another pop is performed). On the next push, the LSTM uses its state from t, rather than i, as the preceding cell state and output. We use stack LSTMs for all of DeepCx's list embeddings, since our transition scheme involves frequently moving items between lists.

We are aware of only two other attempts to use neural networks for transition-based semantic parsing. The first is Swayamdipta et al.'s (2016), who roughly follow Henderson et al.'s (2008) transition scheme for joint syntactic and PropBank-style semantic parsing, using a second stack that holds semantic arcs rather than dependency arcs. They augment Dyer et al.'s stack LSTM parser with this larger set of transitions. As noted by Choi and Palmer (2011b), such joint syntactic/semantic schemes were adapted from existing syntactic transitions, rather than being designed expressly for semantic parsing.

The second system is SLING (Ringgaard et al., 2017), a transition-based frame parser that uses bidirectional LSTMs to encode a sentence, then repeatedly applies an RNN that proposes actions. The transition scheme is custom-designed for frame-based parsing, although it is not specific to FrameNet. In principle it can be applied to any representation where phrases trigger frames with roles that are filled by spans in the sentence, and in fact PropBank-derived data was used for the experiments reported. Like other semantic parsers, SLING assumes that triggers are consecutive sequences of words.

8.3 Causal language transition scheme

The transition scheme used for DeepCx is closely based on that of Choi and Palmer (2011b), modified to suit the additional needs of causal construction tagging. The most important element we borrow is the notion of "bidirectional top-down search": the algorithm first tries to find a connective word, and once it has found one, it compares it with each word both to the right and to the left. In each comparison, the algorithm selects a transition that best fits the word's relationship with the current connective word. The available transitions label the word either as unrelated to the current connective word, as another connective word (or "fragment"), or as a member of one or more of the argument spans. When the algorithm has finished comparing each word in the sentence to the current connective word, it advances to the next potential connective word. In the worst case, then, processing a sentence can take $O(n^2)$ transitions.

The full set of transitions is shown in Table 8.1. The transitions employ a number of auxiliary variables representing the current state of the algorithm. a is the index of the current possible "connective anchor"—the word being tentatively treated as the initial (i.e., leftmost) word of a connective. s is a boolean variable indicating whether we are in the process of comparing all words in the sentence to a, i.e., whether a has been confirmed as a connective anchor. λ_1 is the list of word indices to the left of a that have not yet been compared with it, and λ_2 represents the words to the left of a that have already been compared with it. Likewise, λ_3 and λ_4 contain the indices of compared and uncompared words, respectively, to the right. Thus, words move from λ_1 to λ_2 and from λ_4 to λ_3 as they are compared with a.

The algorithm also maintains a set of partially-constructed instances of causal language. Each instance consists of a set of connective word indices, plus one set of argument word indices for each of the three argument types (Cause, Effect, and Means). For the purpose of formally describing the transitions, we represent the causal instances as a set A of labeled arcs. The head a of each arc is the connective anchor of a causal language instance i (an arbitrary identifier). The label of the arc indicates what role the tail t plays with respect to i: Cause, Effect, or Means if t is a member of the corresponding argument span, and Frag if t is a fragment of the connective other than a. No arc is included in A if t plays no role in i.

As the algorithm scans from left to right, assigning each word index to a in turn, it chooses whether or not to proceed under the assumption that a is the anchor for a connective. The NO-CONN transition indicates that a is not the start of a connective. If the algorithm determines that a does start a connective, it issues a NEW-CONN action. It then proceeds to compare each word to the left of a, in right-to-left order (i.e., starting from the word closest to a), then each word to the right (in left-to-right order). For each comparison, it issues a LEFT-ARC/RIGHT-ARC, CONN-FRAG, or NO-ARC action, depending on whether the comparison word is deemed part of an argument, part of the connective, or neither. After all words have been compared with a (i.e., once λ_1 and λ_4 are empty), the algorithm automatically issues a SHIFT transition to advance a to the next connective anchor candidate.

The initial state is: $\lambda_1 = \lambda_2 = \lambda_3 = []$, $a = 1$, $\lambda_4 = [w_1 \mathinner{..} w_n]$, $s = \text{f}$, and $A = \varnothing$ (the empty set), where w_1 is the first word in the sentence and w_n is the last. The algorithm terminates when $a = n$ and either $\lambda_3 = \lambda_4 = []$ or s is false—i.e., when no words remain to the right of a, and either a is not a connective anchor or all words in the sentence have been compared with it.

An example transition sequence for a sentence is shown in Table 8.2.

Transition schema	Effect and preconditions
NO-CONN	$(\boldsymbol{\lambda_1}, \lambda_2, \boldsymbol{a}, \lambda_3, [\boldsymbol{w} \mid \boldsymbol{\lambda_4}], s, A) \Rightarrow ([\boldsymbol{\lambda_1} \mid \boldsymbol{a}], \lambda_2, \boldsymbol{w}, \lambda_3, \boldsymbol{\lambda_4}, s, A)$ $\lambda_2 = \lambda_3 = [\,]$, $s = \mathsf{f}$
NEW-CONN	$(\lambda_1, \lambda_2, a, \lambda_3, \lambda_4, \boldsymbol{s}, A) \Rightarrow (\lambda_1, \lambda_2, a, \lambda_3, \lambda_4, \mathbf{t}, A)$ $s = \mathsf{f}$
NO-ARC-LEFT	$([\boldsymbol{\lambda_1} \mid \boldsymbol{w}], \boldsymbol{\lambda_2}, a, \lambda_3, \lambda_4, s, A) \Rightarrow (\boldsymbol{\lambda_1}, [\boldsymbol{w} \mid \boldsymbol{\lambda_2}], a, \lambda_3, \lambda_4, s, A)$ $\lambda_3 = [\,]$, $s = \mathsf{t}$
NO-ARC-RIGHT	$(\lambda_1, \lambda_2, a, \boldsymbol{\lambda_3}, [\boldsymbol{w} \mid \boldsymbol{\lambda_4}], s, A) \Rightarrow (\lambda_1, \lambda_2, a, [\boldsymbol{\lambda_3} \mid \boldsymbol{w}], \boldsymbol{\lambda_4}, s, A)$ $\lambda_1 = [\,]$, $s = \mathsf{t}$
LEFT-ARC$_x$	$([\boldsymbol{\lambda_1} \mid \boldsymbol{w}], \boldsymbol{\lambda_2}, a, \lambda_3, \lambda_4, s, \boldsymbol{A}) \Rightarrow (\boldsymbol{\lambda_1}, [\boldsymbol{w} \mid \boldsymbol{\lambda_2}], a, \lambda_3, \lambda_4, s, \boldsymbol{A} \cup \{\boldsymbol{a} \xrightarrow{\boldsymbol{x_i}} \boldsymbol{w}\})$ $\lambda_3 = [\,]$, $s = \mathsf{t}$
RIGHT-ARC$_x$	$(\lambda_1, \lambda_2, a, \boldsymbol{\lambda_3}, [\boldsymbol{w} \mid \boldsymbol{\lambda_4}], s, A) \Rightarrow (\lambda_1, \lambda_2, a, [\boldsymbol{\lambda_3} \mid \boldsymbol{w}], \boldsymbol{\lambda_4}, s, \boldsymbol{A} \cup \{\boldsymbol{a} \xrightarrow{\boldsymbol{x_i}} \boldsymbol{w}\})$ $\lambda_1 = [\,]$, $s = \mathsf{t}$
CONN-FRAG	$(\lambda_1, \lambda_2, a, \lambda_3, [w \mid \lambda_4], s, \boldsymbol{A}) \Rightarrow (\lambda_1, \lambda_2, a, \lambda_3, [w \mid \lambda_4], s, \boldsymbol{A} \cup \{\boldsymbol{a} \xrightarrow{\mathsf{FRAG}_i} \boldsymbol{w}\})$ $\lambda_1 = [\,]$, $s = \mathsf{t}$, $a \neq w$
SHIFT	$(\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \boldsymbol{a}, [\boldsymbol{w} \mid \boldsymbol{\lambda_3}], \boldsymbol{\lambda_4}, \boldsymbol{s}, A) \Rightarrow (\boldsymbol{\lambda_2}, [\,], \boldsymbol{w}, [\,], \boldsymbol{\lambda_3}, \mathbf{f}, A)$ $\lambda_1 = \lambda_4 = [\,]$, $s = \mathsf{t}$
SPLIT	See text (§8.3.2)

Table 8.1: The transitions available in the DeepCx transition scheme. The pre- and post-transition states are expressed as tuples $(\lambda_1, \lambda_2, c, \lambda_3, \lambda_4, s, A)$. x stands for any one of Cause, Effect, Means, or any combination thereof. i indicates the current causal instance under construction; thus, x_i denotes an argument or fragment arc specific to instance i. Elements that differ between the pre-transition and post-transition states are bolded. The preconditions ensure that no rightward action is ever performed before all leftward actions are completed, yielding a consistent transition order.

8.3.1 Overlapping arguments

Occasionally, a word may be part of multiple arguments of the same connective (e.g., both the Cause and the Effect). For example, in *This equipment is newer and thus safer*, the Cause and Effect of *thus* would respectively be annotated as *this equipment is newer* and *this equipment is safer*. When processing such a shared word, the algorithm issues an arc transition that includes both argument names (e.g., LEFT-ARC$_{\text{Cause,Means}}$). From the tagger's standpoint, this is an entirely separate transition from, say, LEFT-ARC$_{\text{Cause}}$ or LEFT-ARC$_{\text{Means}}$.

When executing an action with multiple argument types, a separate arc is added to A for each argument type—i.e., the word is added to both argument spans.

8.3.2 SPLIT transitions

In BECAUSE, a word from one connective can also be part of another connective. This most commonly occurs with conjoined arguments where portions of the connective are repeated. For example, in *we'll need better cooperation* ***if*** *we* ***are to*** *succeed or even* ***to*** *escape catastrophe*, *succeed* and *escape catastrophe* would be considered the Causes of two different causal instances, each with a connective starting with *if are* (see §4.1.5.3). The SPLIT action handles such cases. A SPLIT transition completes the current causal language instance and starts a new one, copying over all connective and argument words up

Transition	λ_1	λ_2	a	λ_3	λ_4	s	A
–	[]	[]	1	[]	[1..7]	f	∅
NO-CONN	**[1]**	[]	**2**	[]	**[2..7]**	f	∅
NO-CONN	**[1,2]**	[]	**3**	[]	**[3..7]**	f	∅
NO-CONN	**[1..3]**	[]	**4**	[]	**[4..7]**	f	∅
NEW-CONN	[1..3]	[]	4	[]	[4..7]	**t**	{**because$_4$ (•, •)**}
LEFT-ARC$_{\text{Effect}}$	**[1,2]**	**[3]**	4	[]	[4..7]	t	{because$_4$ (**moved$_3$**, •)}
LEFT-ARC$_{\text{Effect}}$	[1]	**[2,3]**	4	[]	[4..7]	t	{because$_4$ (**they$_2$** moved$_3$, •)}
NO-ARC-LEFT	**[]**	**[1..3]**	4	[]	[4..7]	t	{because$_4$ (they$_2$ moved$_3$, •)}
NO-ARC-RIGHT	[]	[1..3]	4	**[4,5]**	**[6,7]**	t	{because$_4$ (they$_2$ moved$_3$, •)}
CONN-FRAG	[]	[1..3]	4	**[4,5]**	**[6,7]**	t	{because$_4$**/of$_5$** (they$_2$ moved$_3$, •)}
RIGHT-ARC$_{\text{Cause}}$	[]	[1..3]	4	**[4..6]**	**[7]**	t	{because$_4$/of$_5$ (they$_2$ moved$_3$, **the$_6$**)}
RIGHT-ARC$_{\text{Cause}}$	[]	[1..3]	4	**[4..7]**	**[]**	t	{because$_4$/of$_5$ (they$_2$ moved$_3$, the$_6$ **schools$_7$**)}
SHIFT	**[1..4]**	**[]**	**5**	**[]**	**[5..7]**	**f**	{because$_4$/of$_5$ (they$_2$ moved$_3$, the$_6$ schools$_7$)}
NO-CONN	**[1..5]**	[]	**6**	[]	**[6,7]**	f	{because$_4$/of$_5$ (they$_2$ moved$_3$, the$_6$ schools$_7$)}
NO-CONN	**[1..6]**	[]	**7**	[]	**[7]**	f	{because$_4$/of$_5$ (they$_2$ moved$_3$, the$_6$ schools$_7$)}
NO-CONN	–	–	–	–	–	–	–

Table 8.2: The sequence of algorithm states for the oracle transition sequence for *Well*$_1$, *they*$_2$ *moved*$_3$ *because*$_4$ *of*$_5$ *the*$_6$ *schools*$_7$. Words have been replaced by their indices (indicated in the previous sentence by subscripts). Elements that differ between the pre-transition and post-transition states are bolded. Instances are notated as connective(Cause, Effect).

to the point of the repeated connective word. Further actions are applied to this new causal language instance.

8.3.3 Differences from the PropBank Tagging transition scheme

The overall structure of this scheme is similar to that of Choi and Palmer, but with four important modifications:

1. Choi and Palmer assume that PropBank predicates are provided by an oracle, leaving the transition scheme to handle just the argument identification. In contrast, DeepCx does not have oracle connectives, so it uses the NEW-CONN transition to initiate a new causal language instance. This difference also necessitates storing s, which tracks whether or not such a connective-initiating transition has occurred.

2. Unlike PropBank, BECAUSE allows a target—a connective, in our parlance—to consist of multiple content words.[2] Our transition scheme therefore includes a CONN-FRAG transition. To simplify the algorithm, we always consider the leftmost connective word to anchor the connective. All CONN-FRAG transitions are therefore between a connective word and a word to its right.

3. The connective word can also be part of an argument, as in *enough food to survive*, where *enough* is part of both the connective and the Cause. Our transition scheme therefore compares each

[2]In PropBank, verbs can have attached particles (e.g., *pick up*, but these are not reflected in Choi and Palmer's transition scheme. The particles (which are pre-identified by the predicate oracle) are simply treated as non-arguments of the verb, which is not sufficient for our purposes.

connective anchor with itself in addition to the other words in the sentence. (This is why for each new connective anchor a, λ_4 starts out with a as its first element. This is also why the CONN-FRAG action does not advance to the next potential argument word: a connective fragment can also be part of an argument.)

4. PropBank never posits two predicates for a single verb, but BECAUSE allows one connective word to be shared by multiple connectives. As described in §8.3.2, the SPLIT transition was introduced to handle this case.

8.3.4 Constraints on transitions

In addition to the preconditions listed in Table 8.1, several transitions have constraints on what order they may appear in. These are enforced at each step, both at training and test time, by eliminating violating transitions from the tagger's set of available next actions. The following constraints apply:

- A CONN-FRAG may not immediately follow a SPLIT or another CONN-FRAG.
- A SPLIT may not immediately follow a CONN-FRAG or another SPLIT.
- A SPLIT may be performed only if the connective currently under construction has at least one fragment, i.e., the connective span so far is at least two words.
- NO-CONN is forbidden if s is true, i.e., if a has already been determined to be a connective anchor.

8.3.5 Transition scheme variants

In addition to the transitions described above, we also experimented with eliminating the separate NEW-CONN transition. Rather than setting s to t in a separate step, in this variant a NO-ARC, LEFT-ARC, or RIGHT-ARC transition automatically initiates a new connective and sets s. The version of the scheme with NEW-CONN performed marginally better and is easier to describe, so we retained this variant.

The treatment of overlapping arguments, while serviceable, completely divorces multiple-argument transitions from their single-argument cousins (see §8.3.1). To allow the tagger to learn about such arguments alongside non-overlapping arguments, it may be more effective to insert a NEXT-ARG transition between each pair of arc transitions to indicate that all roles for the current possible argument token have bene added. Thus, to tag a word that belongs to just one argument (as is most common) would then require not just, say, a single LEFT-ARC$_{\mathrm{Cause}}$, but a LEFT-ARC$_{\mathrm{Cause}}$ followed by a NEXT-ARG. If the same token were also part of the Effect, this would allow a LEFT-ARC$_{\mathrm{Effect}}$ to be inserted before the NEXT-ARG. (Of course, some additional constraints would be necessary to ensure that the tagger only ever considers an arc transition it has not already performed on that argument word.) On the other hand, it may be that having the option of waiting to move on will encourage the parser to overlap argument spans too often. Overlapping arguments are rare in our corpus, so we did not explore alternative ways of handling them, but such an extension is an interesting direction for future work.

8.3.6 Edge cases not covered by the scheme

When a SPLIT transition forks one instance into two, the algorithm needs a policy for which connective and argument words from the original instance to preserve, which such words to drop from the fresh instance, and when/whether to apply further actions to which of the instances. Currently, DeepCx assumes that all connective or argument words preceding the repeated connective word are shared, but all actions on words to the right of the repeated word apply only to the new instance. No cases in our corpus violated this assumption, but it is possible in principle. Such cases would require more careful definition of the semantics of a SPLIT, or possibly additional transitions that would produce instance-specific policies.

One edge case that does occur in our corpus is instances that are split across sentences. BECAUSE does not actually allow inter-sentence connectives or extrasentential arguments, so such splits are normally a non-issue. However, the training and testing data are all read into DeepCx from the transition oracle's outputs, and our oracle uses the Stanford CoreNLP sentence splitter, which occasionally chokes on cases like mid-sentence abbreviations (e.g., *Lt. Smith*). Additionally, it splits quotes containing sentence-final punctuation into multiple sentences, even though the entire multi-sentence quote may be an argument to a speaking verb. In such sentences, it may be impossible for even oracle transitions to represent causal instances properly: spans from a single instance may appear to come from multiple sentences, whereas the transitions act on just one sentence at a time. To ensure accurate evaluation, the oracle appends to each sentence's transition list a count annotated instances that crossed the automatically detected sentence boundaries.

8.4 DeepCx neural network architecture

With the transition scheme in place, the next step is to build a system that can select a next action based on its history and current state. Given the experience of previous taggers, we predicted that syntactic information would play a key role in informing these decisions (despite its relative unimportance for pattern-based methods). We therefore built our system on top of Dyer et al.'s (2015) stack LSTM parser, allowing us to directly incorporate the parser's embeddings as inputs to our system. For example, part of the embedding for a token can be an embedding of the parse subtree rooted at that token. For this purpose, we do not need to have our network learn a new embedding for subtrees; we can simply reuse the embedding of that tree used internally by the parser. (Of course, a side benefit is that we could reuse the parser's architecture for DNN-based transition tagging.)

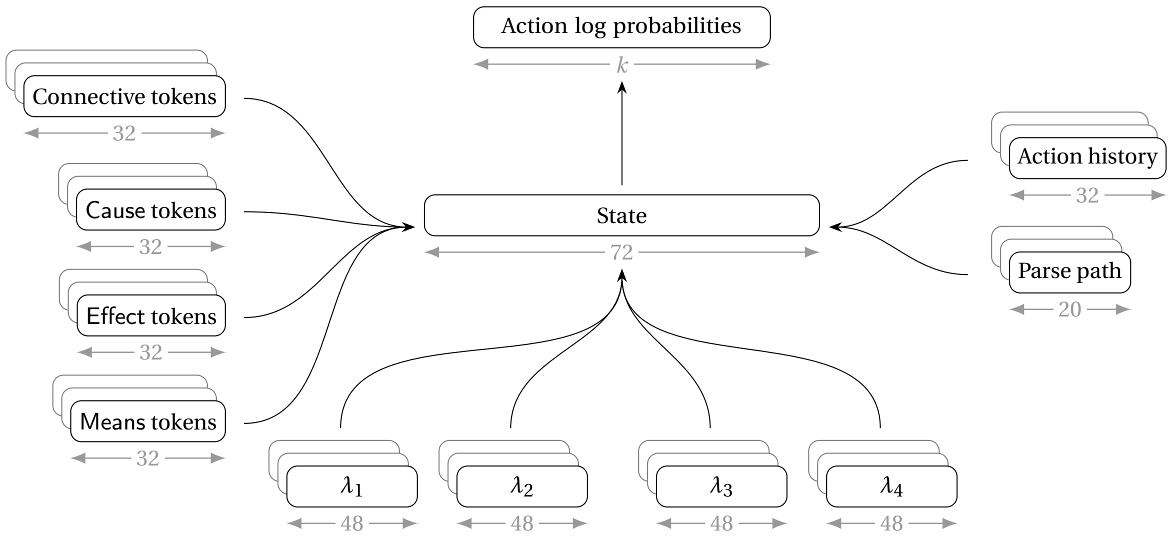
Like the stack LSTM parser, at each step the network computes a high-dimensional vector representing the current state of the internal data structures (λ's, the causal instance currently under construction, etc.). It then uses that state to predict a next action. The output layer of the network is a length-k vector, where k is the number of distinct transitions seen in training, with each vector component giving the predicted log probability that the corresponding transition is the correct next action. At test time, the highest-scoring predicted action is taken. At training time, the gold-standard action is executed instead.

Figure 8.1 shows a schematic of the neural network structure. We elaborate on the components of this schematic below.

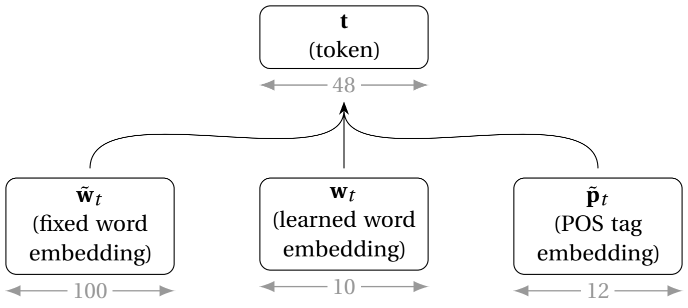
8.4.1 Final state and prediction layers

The inputs to the state vector, as shown in the schematic, are the following:

- a, the current connective anchor token.
- λ_{1-4}, the lists of tokens to the left and right of a that have and have not yet been compared with it.
- h, the history of actions performed thus far for the sentence.
- d, the path in the dependency parse tree between a and the token currently being compared with it.



[a] The overall neural network architecture.



[b] Schematic of the architecture for embedding a token.

Figure 8.1: Schematics of various components of the neural network. Each lone box represents a vector. Stacked boxes represent LSTMs: at any given time, the state is a single vector, but that state encodes information from a series of inputs.

- The lists of tokens making up the connective (o), Cause (c), Effect (e), and Means (m) spans for the causal language instance currently under construction (all empty if no instance is currently being constructed).

The parser state **s** at a given timestep is defined as:

$$\mathbf{s} = \max\{\mathbf{0}, \mathbf{W}_s[\boldsymbol{\lambda}_1;\boldsymbol{\lambda}_2;\mathbf{a};\boldsymbol{\lambda}_3;\boldsymbol{\lambda}_4;\mathbf{o};\mathbf{c};\mathbf{e};\mathbf{m};\mathbf{d};\mathbf{h}] + \mathbf{b}_s\},$$

where $\mathbf{b}_s$ is a bias term, **W** is a learned parameter matrix, and any other bold variable **x** indicates a neural embedding of a variable x. We discuss the construction of each input's embedding in the next subsection (§8.4.2). The max function implements a component-wise ReLU.

The final predicted probability of each transition T is computed from **s** using a softmax unit, which computes the nonlinear function:

$$p(T \mid \mathbf{s}) = \frac{\exp\left(\mathbf{g}_T^\top \mathbf{s} + q_T\right)}{z},$$

where $\mathbf{g}_T$ is a learned neural embedding of T, q_T is a bias for T, and z is a normalizing constant.

8.4.2 Computing embeddings of inputs to the state

8.4.2.1 Embedding a token

Following Dyer et al. (2015), each token t is represented as a concatenation of 3 vector inputs:

- $\tilde{\mathbf{w}}_t$, a fixed word embedding for t's surface form, drawn from a neural language model.
- $\mathbf{w}_t$, a small additional word embedding of t, which allows the network to learn task-specific representations of words related to causality. This is the only component of a token's representation that is trained specifically for this task (i.e., that does not use an embedding from a pretrained language model or a syntactic parsing model).
- $\mathbf{p}_t$, the LSTM parser's embedding of the POS tag it assigned to t in preprocessing. Because we build directly on the parser's code, we can simply access the parser's internal embeddings for POS tags.

The concatenated vector is passed through a linear transformation **V** (with an associated bias term **d**) and a component-wise ReLU. The full expression for embedding a token t is:

$$\mathbf{t} = \max\left\{\mathbf{0}, \mathbf{V}\left[\tilde{\mathbf{w}}_t; \mathbf{w}_t; \mathbf{p}_t\right] + \mathbf{d}\right\}$$

This mathematical operation is shown schematically in Figure 8.1[b].

8.4.2.2 Embedding a list of tokens

Most of the inputs to the final state vector are not just tokens, but lists of tokens. For each such list, we add an LSTM cell to the network.

In the case of the spans for the instance under construction—i.e., the connective (**o**), Cause (**c**), Effect (**e**), and Means (**m**) spans—generating token list embeddings is straightforward. Whenever a transition adds a token to one of these lists, that token's embedding is added as the next element in the corresponding LSTM's input sequence. The LSTM's updated output is then used for all subsequent actions until another transition modifies the span.

The procedure is slightly more complicated for embedding the λ's. As transitions are taken, tokens may need to be moved between the different lists. For example, the argument token is moved from λ_1 to λ_2 after a LEFT-ARC transition, and the connective anchor token is moved from λ_4 to λ_1 on a NO-CONN.

We implement these transfers using the stack LSTM data structure. Initially, all tokens' embeddings are added as inputs to λ_4, but in **reverse order**, so that the leftmost token is the last to be added. Whenever it is time to remove the leftmost token from λ_4—i.e., when advancing to a new potential connective anchor t (SHIFT/NO-CONN) or when a rightward token t has just been compared to a (RIGHT-ARC/NO-ARC-RIGHT)—the stack LSTM for λ_4 is simply rewound one step so that it returns to the state before t was added. Storing λ_4 in reverse order has the additional advantage that, because LSTMs favor more recently added inputs, tokens closer to the connective anchor have greater sway. Meanwhile, tokens are added in their natural order to λ_3, so the most recently compared tokens are most accessible to the λ_3 LSTM.

λ_1 and λ_2 are a mirror image of λ_4 and λ_3, respectively. Tokens are added to λ_1 on either a SHIFT or a NO-CONN, in the order that they are encountered in the sentence. Thus, the λ_1 LSTM ends up representing an in-order list of tokens up to the current a. If a is then determined to be a connective anchor, tokens to the left of a are moved from λ_1 to λ_2 as they are compared with a. The rightmost token t in λ_1 is the first to be compared, so we can rewind the λ_1 stack LSTM to remove t. The embedding of t is then added to λ_2, leaving λ_2 with a reverse-order list of tokens that have been compared. Once again, the closest uncompared word to the anchor and the most recently compared word are the items most accessible to the λ_1 and λ_2 LSTMs, respectively.

8.4.2.3 Embedding a parse path

As demonstrated by Causeway (Chapter 7), one of the key signals about the appropriate action to take, given a connective anchor a and a candidate argument word t, is the syntactic relationship between a and t. We encapsulate this relationship as a dependency path between a and t. The parse path consists of the sequence of dependency labels on the arcs along the path. For instance, in the example sentence from Figure 7.1 (*I worry because I care*), the path between the first *I* and *because* would be $\bigcirc \xleftarrow{\text{nsubj}} \bigcirc \xrightarrow{\text{advcl}} \bigcirc \xrightarrow{\text{mark}} \bigcirc$ (where the blank nodes take the place of the words *I*, *worry*, *care*, and *because* in the dependency graph).

To embed a parse path, we again use the output of an LSTM cell, where each input is an embedding of a dependency label. We reuse the LSTM parser's pretrained embeddings of dependency arc actions: for a dependency label x, we use the parser's embedding for the syntactic parse action LEFT-ARC(x), if available, and RIGHT-ARC(x) otherwise. We add one extra bit to this embedding to indicate whether a given dependency on the path was traversed forward or backward.

This embedding is similar to that proposed by Roth and Lapata (2016). In their representation, however, the dependency path includes the words encountered along the way and their part-of-speech tags. We experimented with adding these elements to our dependency paths, but found that they consistently reduced performance.

8.4.2.4 Embedding the action history

During training, DeepCx learns vector representations of each action, much as language models learn vector representations of words. To embed the action history, these action embeddings are simply fed as inputs into yet another LSTM cell. This LSTM's output is the embedding of the history thus far.

8.4.3 Implementation details

DeepCx is implemented using a refactored version of the LSTM parser codebase.[3] The neural network framework, which also lies at the core of the LSTM parser, is an early version of DyNet (Neubig et al., 2017). The LSTM parser model is pretrained on the usual Penn Treebank (Marcus et al., 1994) sections (training: 02–21; development: 22).

For $\tilde{\mathbf{w}}$, we use the same word embeddings as the LSTM parser, which employs a variant of "structured skip n-gram" embeddings. See Dyer et al. (2015) for details about the embedding approach, hyperparameters, and training corpora. DeepCx gives no special treatment to out-of-vocabulary items, other than using the $\mathbf{0}$ vector for words not included in the pretrained embeddings.

The full code for DeepCx is available on GitHub.[4]

[3] `https://github.com/clab/lstm-parser/tree/easy-to-use`
[4] `https://github.com/duncanka/lstm-causality-tagger`

Embedding or neural network layer	**Dimensionality**
Action embeddings (inputs to h)	8
Task-specific word embeddings ($\mathbf{w}$)	10
Token embeddings ($\mathbf{t}$)	48
Hidden states of λ LSTMs	48
Hidden state of parse path LSTM (d)	20
Hidden state of action history LSTM (a)	32
Hidden states of connective and argument span LSTMs (o, c, e, m)	32
Full tagger state ($\mathbf{s}$)	72

Table 8.3: DeepCx neural network dimensionalities.

8.4.3.1 Dimensionality

The pretrained LSTM parser model uses the same dimensionality settings as the original LSTM parser (Dyer et al., 2015).

The DeepCx neural network dimensionalities used in the experiments reported below are shown in Table 8.3. All LSTM cells use two layers of LSTMs before the final output. These values were chosen as an intuitive balance between values that worked well for other projects and what we could reasonably expect to train with the amount of data we have. The dimensionalities are proportional to how much information needs to be captured by each of the embeddings. (We did tinker with these values in early experiments to observe their effects, but the effects were minimal, and we settled fairly early on the dimensionalities reported here.)

8.5 Experiments

Our experiments with DeepCx evaluated the system's overall performance and probed the effects of making different kinds of information available to the tagger.

8.5.1 Experimental setup and training procedure

Because of the small corpus size, all experiments were performed using 20-fold cross-validation, with folds split by sentence. Within each fold, the training procedure randomizes the available data (i.e., everything but the held-out test set) and splits it into 80% training and 20% development. After each sentence has been completely fed through the network, taking gold-standard transitions at each step (see §8.4), backpropagation is performed on all the predictions for that sentence. Performance on the development set is evaluated every 2500 iterations. Once all training sentences have been seen—i.e., after each full epoch—the sentences are re-randomized and split again into training and development.[5] Training terminates either when the connective-level F_1 score[6] on the development data hits 0.999 or when 85% of evaluations in the past five epochs have yielded lower scores than their immediate predecessors.

We followed the training parameters of Dyer et al. (2015): we used gradient descent for parameter optimization, with an initial learning rate of $\eta_0 = 0.1$ and updates of $\eta_t = \eta_0/(1+0.8t)$ after each epoch

[5]This reuse of the development data means that the network can end up memorizing all the training data. However, early experiments with keeping development data out of training resulted in a drop in scores, presumably because it removed too much data from training in each fold. Of course, our final evaluation is still performed on the fold's held-out data.

[6]For the experiment with oracle connectives, action-level prediction accuracy is used instead of F_1 score.

t; we clipped the ℓ_2 norm of the gradients to 5; and we applied an ℓ_2 penalty of 10^{-6} to all weights. We also used Glorot initialization (Glorot and Bengio, 2010) for all parameters. Each fold took about 40 minutes to train on a single core of a 3.10-GHz processor.

Unlike in Chapter 7, in this chapter we do not restrict the evaluation to instances with both Cause and Effect arguments. The more powerful DeepCx system is able to handle a wider range of causal language instances, so we evaluate it on the full tagging task. (Even on the pairwise evaluation, DeepCx still does about 3.5 points better on connective F_1 than Causeway, despite not being trained directly on the pairwise subtask.)

8.5.2 Network variants tested in experiments

Ablation studies: effects of eliminating pieces of the model In addition to simply testing the vanilla configuration described above, we examined which non-essential model components and inputs to the final state vector actually contribute to performance. We were particularly interested in the effects of including or omitting parse information. We tested the following variants of the DeepCx architecture:

1. Eliminating **w**, the task-specific word embeddings. Given the small dataset, learning such embeddings could contribute to overfitting.
2. Eliminating **a**, the action history.
3. Eliminating **d**, the parse path between the connective anchor and the current comparison token.

Argument identification alone Much work in semantic parsing, including Choi and Palmer (2011b), assumes that triggers are already provided, so that the automated tagger needs only to identify argument spans for each trigger. This setup is slightly trickier to arrange in a transition-based system like DeepCx, where there is no separate argument identification phase. We tested performance on the subtask of argument identification alone by providing the tagger with oracle transitions only for actions that act on the connective—i.e., NO-CONN, NEW-CONN, and CONN-FRAG transitions. The system was then responsible for deciding between NO-ARC, LEFT-ARC, and RIGHT-ARC transitions.

Forbidding generalization to novel connectives One of the strengths of the transition-based approach is its ability to recognize new forms of causal language that it has not previously seen, but which resemble known connectives semantically and/or linguistically. Given our relatively small dataset, however, it seemed possible that the system would not have enough data to make meaninful generalizations, tagging previously unseen connectives that were all false positives. We therefore implemented a switch that caused DeepCx to record connectives seen in training. The system would then refuse to allow a test-time NEW-CONN or CONN-FRAG transition unless adding the putative connective word would match the initial word sequence of some known connective.

8.6 Performance of DeepCx and variants

The experimental results for all of these variants are shown in Table 8.4. For comparison, we also include scores for the best-performing Causeway systems on the complete task (i.e., not limiting evaluation to pairwise instances). All systems were run on the same set of folds.

The remainder of this section focuses on evaluating overall metrics for DeepCx. An extensive error analysis and an in-depth dissection of the differences from Causeway are taken up in the next section. All significance tests below are paired, two-tailed t-tests on the results from all 20 folds.

	Connectives			Causes			Effects			Means	
System variant	**P**	**R**	**F_1**	**F_1**	**F_1@.5**	**J**	**F_1**	**F_1@.5**	**J**	**F_1**	**F_1@.5**
Benchmark	***84.1***	19.7	31.6	24.0	24.7	86.6	16.4	22.4	77.6	0.0	0.0
CW-S + benchmark	62.8	46.2	*53.1*	37.9	42.5	81.0	24.8	38.7	73.3	0.0	0.0
CW-L + benchmark	63.4	45.1	52.5	*38.8*	*43.5*	83.7	*30.4*	*40.7*	78.4	0.0	0.0
Vanilla DeepCx	*63.4*	55.8	*59.2*	*43.9*	*50.6*	83.0	41.0	*51.7*	82.2	***0.3***	***0.3***
No **w** (1)	62.7	56.2	59.1	42.8	49.0	81.9	*41.8*	*51.7*	82.5	0.1	0.1
No **d** (3)	62.7	***56.5***	*59.2*	42.7	49.1	80.9	39.6	51.1	80.5	0.1	0.2
No **a** (2)	61.3	54.1	57.3	40.3	48.3	81.6	36.9	50.4	81.1	0.0	0.1
No novel connectives	65.4	**56.5**	**60.5**	**44.7**	***51.0***	82.6	**42.8**	***52.6***	82.1	0.1	**0.3**
Oracle connectives	–	–	–	73.5	80.9	79.7	67.8	82.8	80.7	5.8	13.6

Table 8.4: Results for all variants of DeepCx tested. As before, J indicates Jaccard index. The top section shows Causeway and the benchmark for comparison, the middle gives DeepCx variants, and the final two sections give performance with special constraints. For $P/R/F_1$ scores, the best non-oracle results across all sections are bolded, and the best results within each of the top two sections are italicized.

8.6.1 Overall performance

The vanilla configuration of DeepCx unmistakably eclipses Causeway (and the benchmark, by transitivity) at connective discovery, with a margin of 6.1 F_1 points. Both F_1 scores have relatively high standard deviations across folds (3.6 points for DeepCx, 3.8–4.7 points for Causeway), but the scores vary together; some folds are simply harder. DeepCx usually leads Causeway by at least 5 points, making the difference highly statistically significant ($p \ll 0.001$). The gap appears to come primarily from recall, where DeepCx averages 9.6–10.7 points higher than Causeway.[7] Of course, the hyper-conservative benchmark approach still bests both DeepCx and Causeway on precision, but its miserable recall leaves its F_1 far behind the others'.

It is no great surprise that DeepCx's advantage comes from recall. After all, one of Causeway-S's biggest challenges is that patterns often won't match when they should. Even for Causeway-L, the classifier is so swamped with false positives that it is hard for it to achieve reasonable precision without dragging down recall tremendously. The fact that recall is where DeepCx has an easier time supports our conclusion (§7.3.2.1) that Causeway most needs ways of upweighting positive instances.

On end-to-end argument identification, DeepCx again outperforms Causeway, particularly on recall, producing a 5–6-point gap in F_1. The Jaccard indices for Causes and Effects are in the low 80's, indicating a high degree of overlap with the gold-standard argument spans. For Causes they are on par with Causeway; for Effects they are notably higher, despite the fact that DeepCx's higher recall gives it more opportunities to be docked for mismatches. Thus, DeepCx does about as good a job as Causeway at finding Cause words, given a correct connective, and substantially better at finding Effect words.

On Means arguments, all systems appear to do extremely poorly. This likely stems from the small number of gold-standard Means instances: less than 2% of instances have Means arguments, so any given fold would have just a few training examples and one or two test examples. Causeway and the benchmark have no mechanism for even detecting Means arguments, so they deserve their 0 scores. DeepCx, on the other hand, actually did better than the metrics suggest. Among its true positive connectives, it had only about a 12% recall rate and a 25% precision rate for exact Means matches.

[7] Recall should perhaps have even been higher: in at least three cases, DeepCx was penalized for correctly flagging connectives that had been missed by annotators.

But on further inspection, at least six of the supposed false positive arguments were in fact correct—the tagger had done a better job of finding Means than the annotators. If these had been counted correct, precision would be 63%. Furthermore, at least one error was merely an extra word added to the beginning of the span. The neural network thus succeeded fairly well at generalizing from the Means instances available, and might have done even better had the examples it found been marked correctly when used as training data in other folds.

8.6.2 Argument identification alone

Argument identification scores remain high when oracle connectives are provided. Naturally, the end-to-end argument scores improve dramatically compared to non-oracle connectives, but the more important question is what fraction of the previous errors remain when connective discovery is no longer a source of error. DeepCx achieves 73.5% F_1 on Causes and 67.8% on Effects with oracle connectives, which implies that the vanilla configuration's argument error (as measured by F_1) was split roughly half and half between connective discovery failures and argument ID failures.

However, the F_1 metrics reflect exact span matches; it is counted as a mismatch if even a single word is missing from or added to the argument. Because in this experiment the system's entire task is to tag arguments, the Jaccard indices give an absolute measure of how closely the predicted argument spans overlap with the gold-standard ones. By that measure, the neural network's treatment of argument identification transitions looks quite robust. Jaccard indices do drop by a few points across the board compared to non-oracle connectives, which is to be expected: with oracle connectives, the system is evaluated on the arguments of every gold-standard instance, including more difficult ones that the vanilla configuration missed. But despite the more exhaustive assessment, DeepCx maintains Jaccard indices of ~80% for Causes and Effects.

8.6.3 Model ablation studies

No pieces of the model beyond the bare essentials contributed much to the final F_1 scores. Removing task-specific word vectors and dependency paths yielded essentially identical precision, recall, and F_1, and while the F_1 without action history was nearly two points lower, the difference was not statistically significant ($p < 0.11$).

Removing these model components did lower scores on argument identifaction by a few points, however. The only statistically significant effects were those of removing action history on Causes ($p < 0.02$ for F_1, $p < 0.19$ for Jaccard indices) and Effects ($p < 0.004$ for F_1). Still, with the possible exception of task-specific word vectors, which marginally reduced Effect F_1, these model components do not seem to result in more overfitting, and may help in some cases.

The meager effects of parse paths came as a surprise; indeed, our entire reason for building on top of the LSTM parser was to make use of its parse information. We expected that embeddings of parse paths would be highly informative for determining when a syntax-dependent construction was in play. The fact that it made little difference to DeepCx suggests that the bulk of the information that syntactic parses would provide is available in some isomorphic form from simpler inputs. How and in what form the network extracts that information remains an open question.

8.6.4 Constraining to known connectives: a mixed bag

Constraining DeepCx to known connectives yields an interesting tradeoff. On the one hand, it improves precision substantially ($p < 0.036$), since the system is forbidden from hypothesizing novel, potentially spurious connectives. As expected, the effect on recall is far less pronounced, averaging out to a statistically insignificant half-point improvement.[8] The improvement in precision results in a 1.3-point increase in F_1, although the difference is not quite statistically significant ($p < 0.09$).

Inspecting some of vanilla DeepCx's false positives reinforces the notion that it might be unwise to allow the system to run wild with new connectives. Some of its odder proposals included *an unfair effort to*, *is insanity*, *eight*, and the dollar sign.

On the other hand, some of its generalizations were startlingly perceptive. For instance, one sentence contained the sequence *allowing states greater opportunity to regulate*, which was not marked by annotators because *allowing* here seems to mean "providing" rather than "permitting." But DeepCx proposed *allowing opportunity to* as a connective, which, if it proved to be a conventionalized construction, would be a plausible candidate for inclusion in the constructicon. Similarly, in other sentences DeepCx tagged *catalyst for* and *fuel* (as in *fueled skepticism*). An even stronger case can be made for these words as connectives; indeed, omitting them from the constructicon was probably a mistake. It seems a shame to artificially handicap the tagger from discovering such cases.

Ultimately, then, whether to permit novel connectives depends on the priorities of the user. If precision is paramount, or if it is important that the tagged connectives match the gold standard exactly, it may be best to restrict connectives to known ones. But if discovering new connectives is more important and some imprecision in the tagged connectives can be tolerated, then the best bet may be to allow the tagger to flex its creativity.

8.7 Where and why DeepCx outshines Causeway

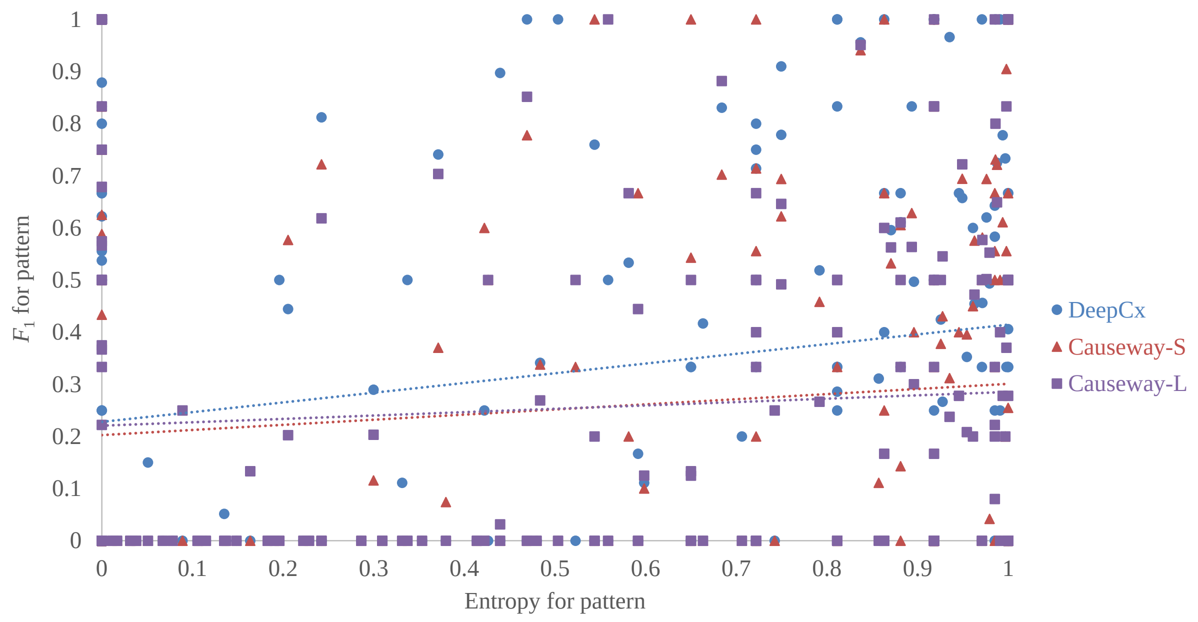
On the whole, DeepCx clearly has the edge over Causeway. It would be reckless, however, to draw sweeping conclusions about the superiority of neural networks for this task without examining what cases drive the difference. It is entirely possible—indeed, we thought it likely—that there would be classes of language that would be better modeled by pattern-based methods, with their explicit syntactic representations and their ability to evaluate an entire putative instance of causal language at once. This section lists some of the hypotheses we tested to explain the differences, and our conclusions about each. All analyses in this section were performed on the vanilla variant of DeepCx.

Ambiguity It seemed likely that the systems would exhibit different behavior depending on how ambiguous a connective is. For example, DeepCx might maintain higher performance as as connectives become more ambiguous, which would help explain why it performs better overall. To test this, we assigned each lexicalized pattern (see §5.2) an ambiguity score, as follows: for each connective pattern c, we counted k_c, the number of instances marked with connective c. We also counted n_c, the number of candidates for that pattern in the corpus, defined as the number of times the same lemmas appear in the correct order in a sentence. The empirical probability of a pattern being causal is thus $p_c = \frac{k_c}{n_c}$. The connective pattern's unpredictability can then be measured as the Shannon entropy of a Bernoulli random variable:

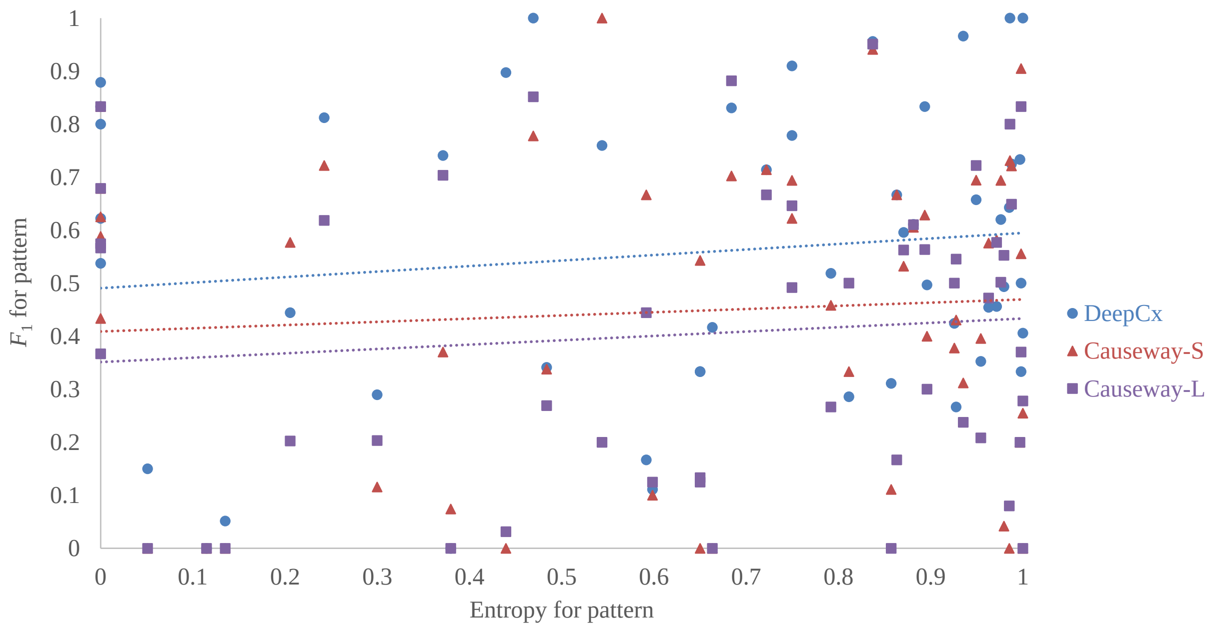
$$H_c = -p_c \log_2(p_c) - (1 - p_c) \log_2(1 - p_c)$$

H_c ranges between 0 (predictably either causal or not causal) and 1 (a 50/50 chance of being causal). See Appendix D for the computed entropy scores for the observed patterns.

[8] To the extent that recall does improve, it is presumably because some false positives in the vanilla configuration were actually true positives with an extra word or two in the connective. For example, in a sentence containing *brought on as the effect of*, DeepCx marked *brought effect* as a connective, whereas the gold standard marked only *effect*. This error resulted in both a false positive and a false negative.



[a] For all patterns



[b] For patterns with 6 or more instances in the corpus

Figure 8.2: Entropy of individual connective patterns vs. F_1 on those patterns, averaged across all folds where the pattern appeared in the test set. Dashed lines indicate linear fits.

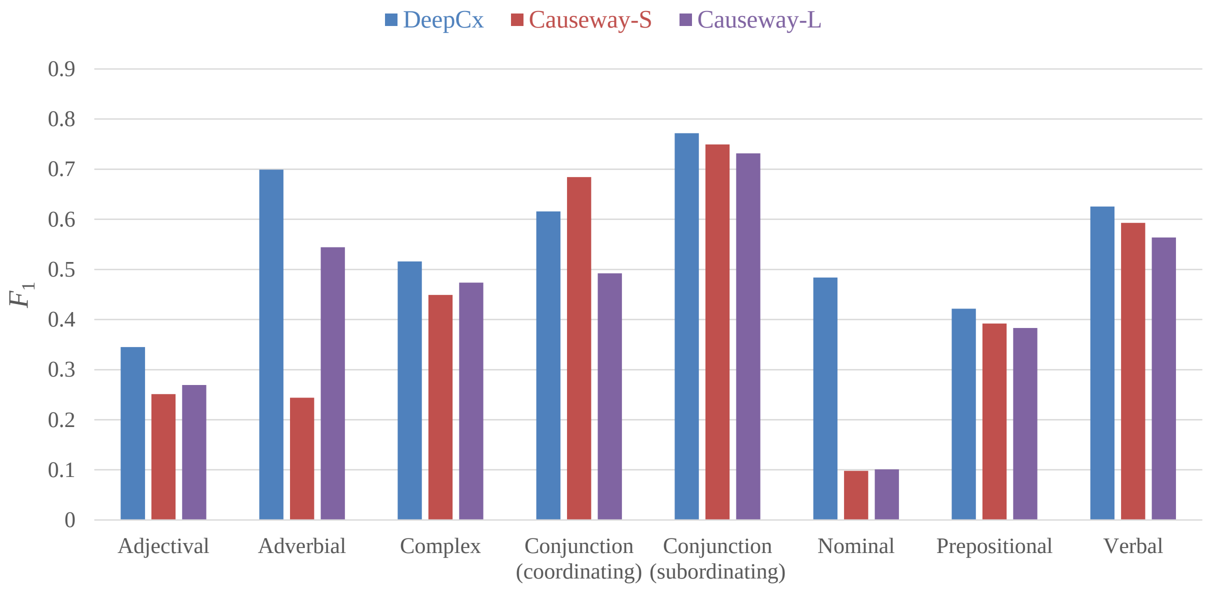


Figure 8.3: F_1 by connective type for DeepCx and Causeway.

.

The relationship between performance and this entropy score is shown in Figure Figure 8.2[a]. To our surprise, we observed little correlation between ambiguity and performance. The weak trend that was present was actually toward **increasing** F_1 with more ambiguous connectives. Given the large cluster of low-entropy points with 0 F_1, we suspected that the plot was skewed by low-frequency patterns: patterns attested only two or three times in the corpus are likely to be unambiguous, but also unlikely to be tagged correctly. To rule out this effect, we re-ran the analysis only on connective patterns that appeared at least 6 times in the corpus. The lack of effect persisted (Figure 8.2[b]), suggesting that ambiguity is not a systematic factor in either system's performance.

Connective POS type The most obvious categorical distinction to check for differences between Causeway and DeepCx is different types of connective: nominal, prepositional, etc. Particularly interesting is performance on complex connectives, one of the driving motivations for the project. F_1 on connective discovery for different types of connectives is shown in Figure 8.3. Recall that these categories were assigned per lexical pattern, so they may be slightly imprecise; see §5.2.

It is clear from the figure that DeepCx's biggest advantage lies in nominals and adverbials. (It also outperforms on adjectives, although there are only 15 adjectival instances in the corpus, so the net effect is much smaller.) Indeed, excluding nominal and adverbial connectives from the systems' scores tightens the F_1 gap by 2.9 points.

In the case of nominals, the issue seems to be a combination of high ambiguity—the average entropy of a nominal pattern is 0.9—and a low number of samples from which to learn the correct generalizations (39). Thanks to the LSTM's ability to generalize from other types of connectives, it can better handle these cases. The explanation for adverbials is more straightforward. Between 8 and 9 points of it stems from the fact that Causeway-S managed to miss every single instance of *why*. Causeway also missed most of the *when*s, the most common adverbial connective. Combined, these two connectives explain most of the gap in adverbials.

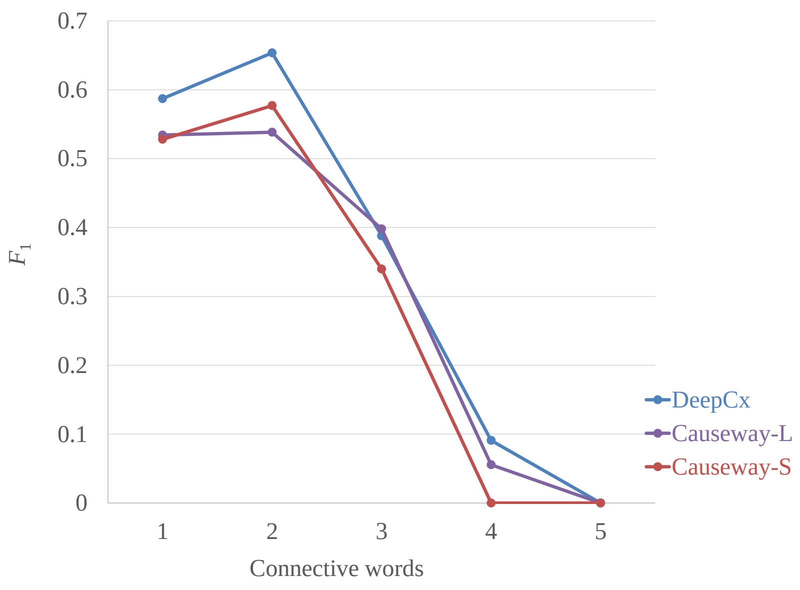


Figure 8.4: F_1 by connective length for DeepCx and Causeway.

.

Specific connectives The connective *to* is an outlier in its impact on the scores. Not only is it the most common connective in the corpus, it is also among the most ambiguous, with an entropy of 1. DeepCx does far better on both precision and recall for *to*, resulting in an average F_1 of 40.6% instead of 25.5%. This connective alone accounted for a third of the F_1 gap between Causeway-S and DeepCx; if the two systems are evaluated on all other connectives, the divide between their overall scores shrinks from 6.1 to 4.2 points. Discounting *to* for Causeway-L produced a smaller but still notable difference of 0.8 points.

We suspected that *for*, the third most common connective and another unusually ambiguous one, would have a similar impact. However, eliminating it from the evaluation made no noticeable impact on the scores; Causeway and DeepCx both do equally well at finding and disambiguating it. The same was true of *because* (which has a high ambiguity score because *because of* can be mistaken for it).

Connective length Most connectives are one or two words, with some patterns ranging up to five (e.g., *in order to* __, __ *have to* __). More words means more opportunities for a tagger to be confused by intervening words or parse links.

Figure 8.4 shows the performance of each system as it scales with connective length. The plot shows that two-word connectives are generally easier than one-word connectives, perhaps because the second word helps disambiguate the sense of the connective. Still, the overall pattern holds that longer connective words hurt performance, as expected. Much of the dropoff on four- and five-word connectives comes from sheer scarcity of data; longer connectives are less common. Scarcity is especially harmful for Causeway-S because it treats different syntactic patterns as different connectives even for the same lexical items, so the connective discovery stage learns more slowly.

DeepCx outstrips both Causeway implementations at all connective lengths, but its advantage does shrink as connectives get longer. On 3-word connectives, Causeway-L is actually on par with it. This may be because DeepCx is processing word by word, whereas Causeway-L looks for regex matches in entire sentences, making its connective detection less confusable by intervening words.

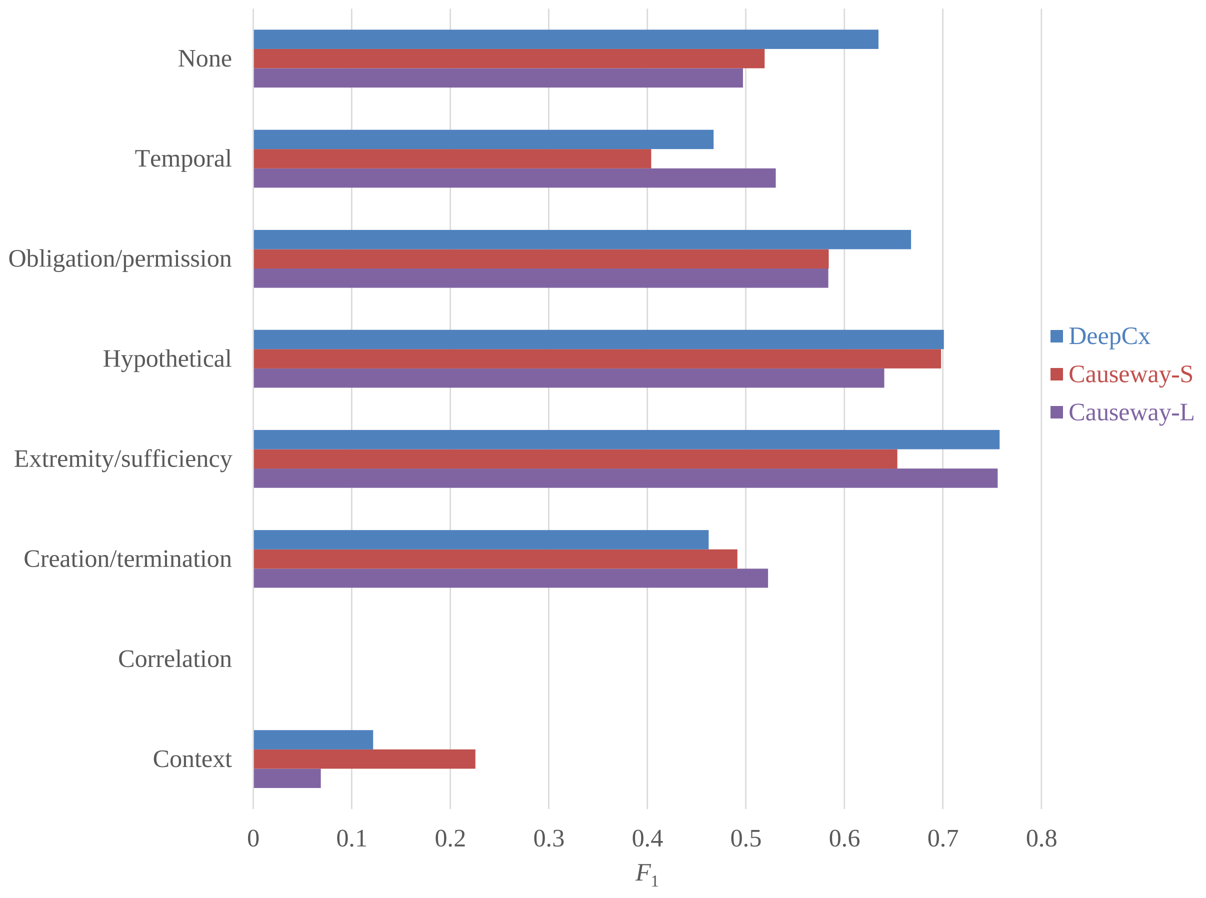


Figure 8.5: F_1 by overlapping relation for DeepCx and Causeway.

.

Overlapping relations Each connective pattern has at most one commonly associated overlapping relation (though a few instances in the corpus are annotated with multiple such overlaps; see §5.1). Figure 8.5 shows each system's performance broken out by the connectives' most common overlapping relations. None indicates patterns that are never associated with an overlapping relation.

The systems show similar trends regarding the relative difficulties of different overlapping relations. No system marks any Correlation constructions as causal (there are only a few in the corpus), and Context relations are also rare and difficult. Extremity/sufficiency relations are the most reliably tagged or close to it by all systems, presumably because the common expressions in this category that indicate causality are very consistent (e.g., *so X that Y* has few non-causal look-alikes). DeepCx generally matches or beats Causeway, though its superiority is statistically significant only for the None category ($p \ll 0.001$).

In the only places where either Causeway system bests DeepCx—on Creation/termination, Context, and Temporal—the difference is not statistically significant, though it comes close for Causeway-S on Context ($p < 0.07$) and for Causeway-L on Temporal ($p < 0.08$).

Degree Although all systems performed somewhat better on average on Inhibit instances than Facilitate instances, the performance differences between different systems were about the same within each category. The lack of effect persisted when we split Inhibit into finer-grained labels of Facilitate and

Enable to match Wolff et al.'s (2005) categories. (As described in §4.1.4.2, our annotators had trouble distinguishing these finer-grained degrees. We approximated them by assigning each connective pattern one of the three labels depending on its most intuitively obvious meaning.)

Amount of training data The amount of training data available for a given pattern obviously affects performance. As Figure 8.6[a] shows, F_1 scores do increase as the amount of data for a given pattern increases, though they only begin to rise in earnest after around the 15 sample mark. The gap between Causeway-S and DeepCx stays fairly constant, indicating that they improve about equally well from more samples, but Causeway-S does start to catch up toward the right of the graph. To our surprise, the system that improves fastest with more training data is Causeway-L, despite its low scores on the left of the plot. It would be interesting to explore whether this trend continues with larger corpora.

Plotting the difference in F_1 between DeepCx and the Causeway systems (Figure 8.6[b]) demonstrates even more conclusively that DeepCx outperforms at all amounts of training data. This implies one of two things: either DeepCx is better able to generalize from a few samples to others like them, or it is better able to generalize from one connective to others. The second explanation is likely to be the bigger factor, since DeepCx surpasses Causeway by 5–10 points even on connectives with only one or two samples in the corpus, which account for about 9% of all instances. These cases cannot be explained by better intra-pattern generalization, suggesting that inter-pattern generalization is the key. This hypothesis is also consistent with the fact that the neural network treats related words as nearby values in the same vector space, rather than as completely disparate symbols.

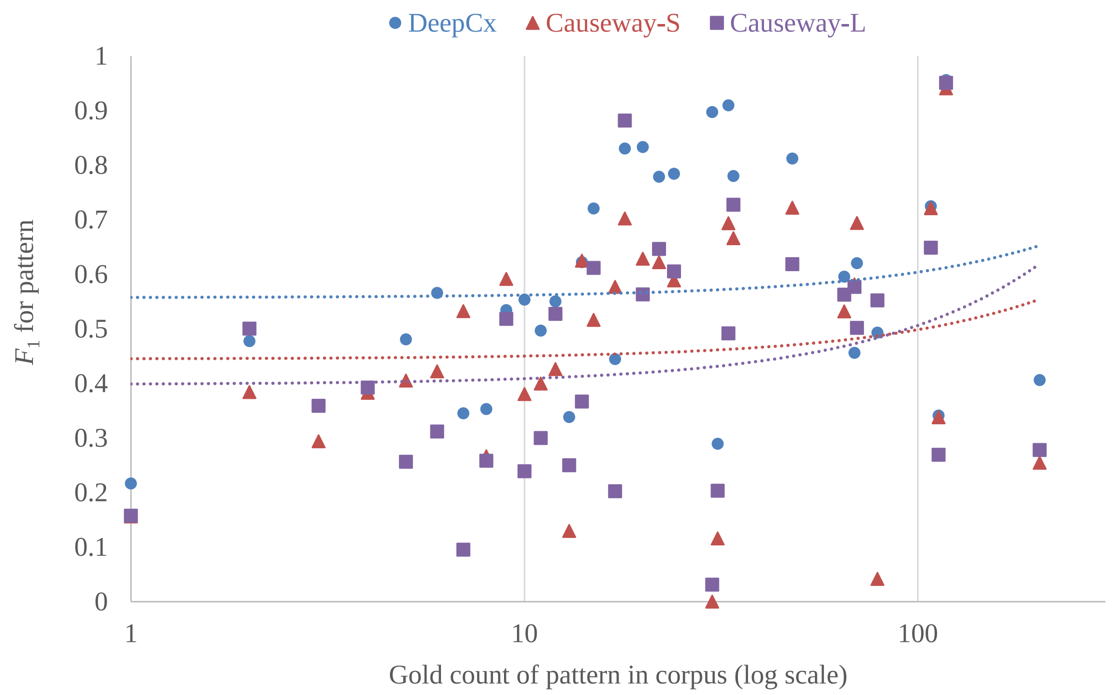
Argument length While argument length does not predict F_1 for any of the systems, we expected longer arguments to be harder to tag correctly. For both Causes (Figure 8.7[a]) and Effects (Figure 8.7[b]), this hypothesis proved weakly true for Causeway-S, but not for Causeway-L or DeepCx. DeepCx consistently edges out Causeway-L on Jaccard index, and both do better than Causeway-S, particularly as arguments grow longer. Apparently the sequence-tagging approach is more reliable than dependency subtrees for argument identification, showing little impact from lengthier arguments.

8.8 What would be needed to handle other kinds of constructions?

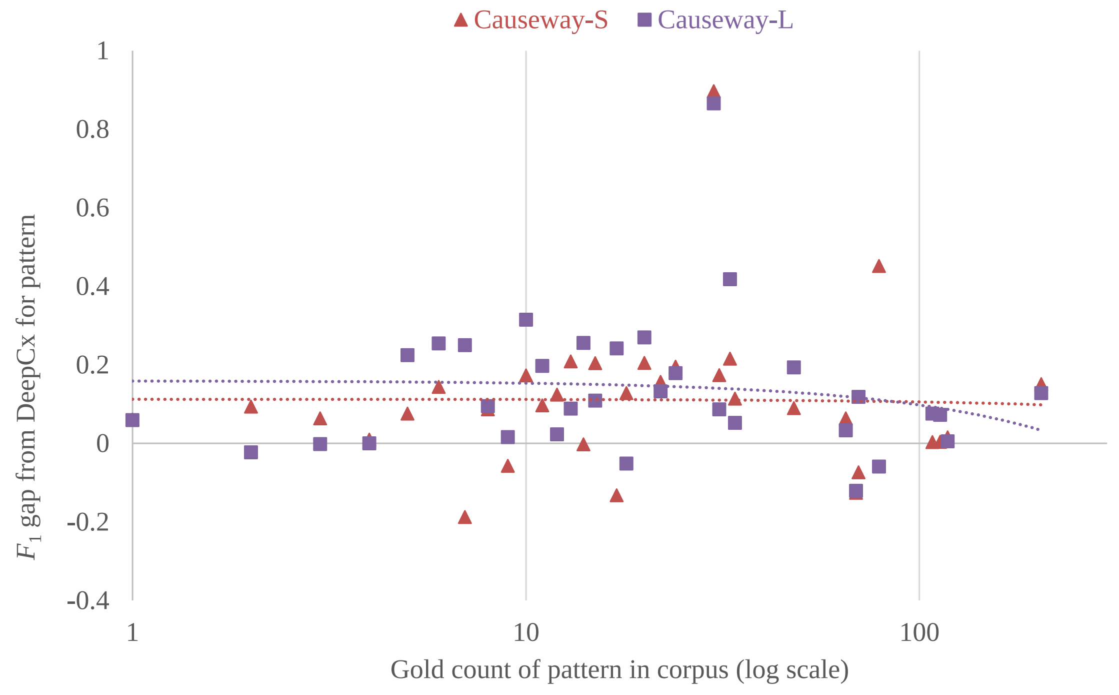
The DeepCx transition scheme was designed specifically for tagging with the BECAUSE annotation scheme. Nonetheless, the design is quite general, and we believe it has the potential to form the basis of more general construction tagging in the SCL paradigm. In addition to handling straightforward instances of the sort PropBank or FrameNet taggers can process, the DeepCx transition scheme allows for discontinuous trigger and argument spans, overlaps between arguments, overlaps between trigger words and arguments, and overlaps between different triggers. Such flexibility is enough to cover most types of constructional quirks that might otherwise interfere with shallow semantic parsing.

Still, there do remain several issues that would require additional consideration for tagging some kinds of common constructions. Most notably, our scheme operates on words as units, but plenty of constructions are sub-lexical. For example, comparatives—another semantic domain where complex constructions abound—make regular use of morphemes like *-er* and *-est*. In many languages, such morphemes are a crucial part of causal language, as well as other semantic relations. In Semitic languages, for instance, causitives are often expressed by merging a verb root with a causative morphological template.[9] One option for handling such constructions is to split words into morphemes and let the tagger

[9]Arguably, even English possesses such causative morphemes: the suffixes *-ize* and *-ify* can sometimes be used as productive causatives.

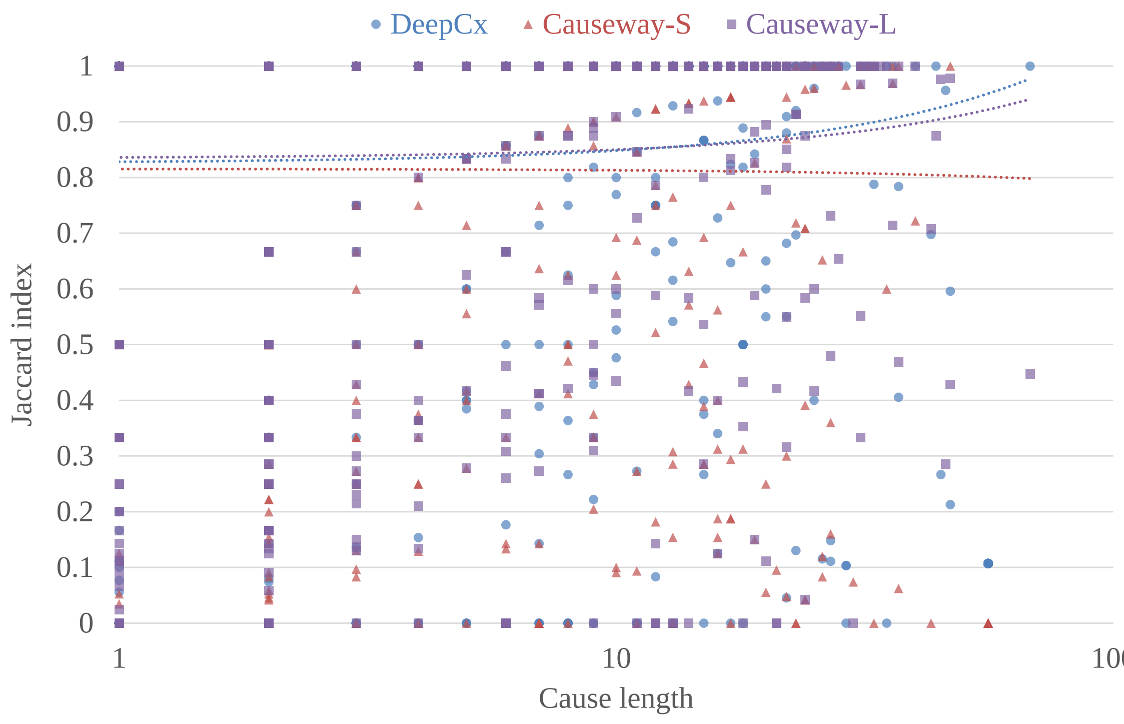


[a] Raw F_1 scores.

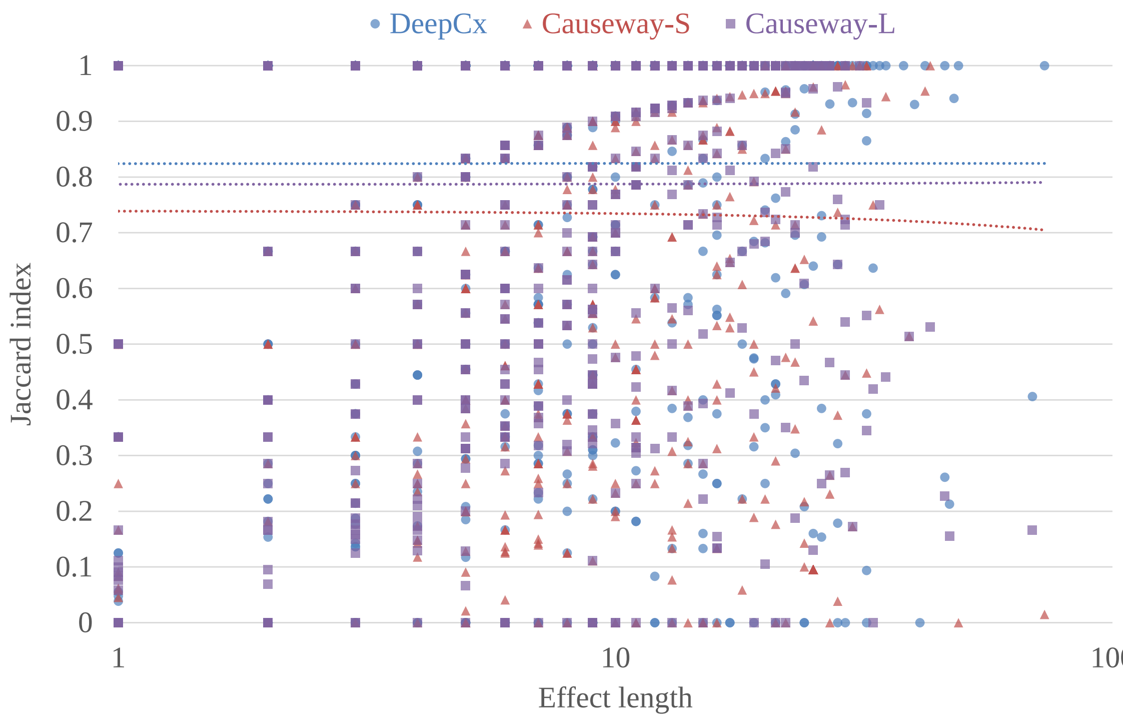


[b] Difference between F_1 score for DeepCx and those for Causeway. Positive scores signify a DeepCx advantage.

Figure 8.6: F_1 by pattern frequency for DeepCx and Causeway. Dashed lines indicate linear fits.



[a] Causes



[b] Effects

Figure 8.7: Jaccard index by argument length (log scale) for all true positives. Dashed lines indicate linear fits. For Means, there were not enough examples to obtain a meaningful plot.

operate on morphemes instead of words. Unfortunately, tagging would then be subject to frequent errors and ambiguities in morphological analysis. Additionally, some kind of morpheme embeddings (or some variant of character-based word embeddings; see, e.g., Ballesteros et al. 2015) would be needed to replace word embeddings in a neural model. An easier-to-implement but less elegant solution would be to simply tag the entire word containing the morpheme (e.g., *bigger*) as part of the construction.

A second challenge for applying a DeepCx-like scheme elsewhere is constructions with no lexical trigger. The simplest way to adapt DeepCx to these constructions would be to add a JUXT transition as a sibling of NEW-CONN. This transition (short for "juxtaposition") would indicate that a new relation instance should be anchored at the boundary between the words currently being processed, but without using either word as a trigger.

Cross-sentential constructions pose yet another challenge: the current scheme operates on one sentence at a time, and does not recognize the possibility of sentential juxtaposition or even cross-sentential grammatical relations as construction possibilities. While it would not be exceptionally difficult to alter the scheme to allow, say, arguments in the previous k sentences, it might make randomized training more difficult.

Finally, as noted in §8.3.6, SPLIT transitions currently make strong assumptions about how one connective will interact with a previous one that shares some of its connective words. Constructions violating these assumptions may require more drastic surgery on the scheme.

8.9 Contributions and takeaways

This chapter has demonstrated DeepCx, a neural transition tagger for causal language. By examining one word at a time and deciding on an appropriate action, DeepCx is able to perform connective discovery and argument identification in one unified framework. As we have seen, the tagger achieves good performance on these very difficult tasks, significantly outshining Causeway by most metrics. The transition scheme, though designed specifically for causal language, could be profitably applied to many other kinds of constructions, as well, as it offers great flexibility with respect to how different triggers and arguments interact.

Our error analysis and comparison yields several conclusions. First, the main advantage DeepCx has over Causeway is recall, particularly for adverbs and nominals. Second, sequence tagging looks to be a stronger choice than syntax-based methods for argument identification. Third, DeepCx's advantage comes largely from generalizing from seen to unseen connectives, rather than, say, learning individual patterns from less training data. Indeed, Causeway-L may even outdo it with more training data, given the lexical pipeline's rate of improvement as training data for a pattern increases. Finally, few of the other factors we expected to correlate with performance actually do: ambiguity, connective length, and overlapping semantic relations all had little impact.

Part III

Looking Forward

CHAPTER 9

Conclusion

9.1 Summary of contributions and lessons learned

This thesis has laid out a novel approach to shallow semantic annotation and parsing that draws on the principles of construction grammar (CxG). Specifically, it has demonstrated the practicality of annotating the wide variety of linguistic patterns used to express causal relations in English, and of automatically reproducing such annotations with data-driven techniques. Our approach expands the traditional notion of semantic parsing, allowing predicates to be borne not just by individual words, but by multi-word expressions and beyond.

The most significant contributions are methodological ones, namely:

1. The **"surface construction labeling" approach** (SCL; Chapters 1 & 2), in which the surface forms of arbitrarily complex constructions can be tagged either by humans or by machines. Rather than explicitly modeling every construction that gives rise to a sentence, in SCL we mark just the fixed lexical components of a construction, allowing linguistic analysis and NLP systems to leverage some key advantages of CxG principles even without a full construction grammar.

2. A comprehensive, SCL-based approach to **annotating causal language and overlapping semantic relations** (Chapter 4), which lends itself to both reliable human annotation and automated tagging.

3. **Pattern-based methods** for parsing complex constructions and their argument spans (Chapter 7) in which tentative construction instances are fully fleshed out before filtering false positives.

4. A **transition scheme** that for the same semantic parsing task, along with a **neural network architecture** that can apply the transitions automatically (Chapter 8).

All but the second of these can be applied to domains beyond English causal language.

In addition, our work has produced three freely available artifacts which future linguistic analysis and NLP work can build on:

1. The **BECAUSE annotated corpus** of causal language, along with some initial insights derived from it (Chapter 5).

2. The **Causeway software** for pattern-based tagging (Chapter 7).

3. The **DeepCx software** for neural transition-based tagging (Chapter 8).

9.1.1 Takeaways for annotation efforts

As discussed in the concluding sections of previous chapters, our experience offers a number of lessons for future annotation work.

Regarding annotation specifically for causality, the BECAUSE scheme and corpus confirm that focusing on causal language can achieve much higher inter-annotator agreement than annotating for real-world causal interactions (§4.4). Additionally, the diversity of constructions we observed shows that to capture the full range of such language, we do indeed need more flexibility than previous lexical unit approaches have provided (§5.2.2).

Generalizing beyond causality, the BECAUSE project affirms that shallow semantic tagging need not restrict itself to particular parts of speech or types of language as triggers. FrameNet pioneered starting from a semantic relation and annotating many types of language that express it. With the SCL approach, we have expanded that paradigm to include multi-word expressions and even more complex constructions as triggers, as well as overlapping relations.

SCL does make some aspects of annotation more difficult. Because any pattern can be a trigger, many of the annotators' decisions keep revisiting whether a given pattern should be annotated at all (§4.3). This problem can be mitigated with the same kind of corpus lexicography that prior semantic tagging schemes have used, and the same approach would be applicable for any annotation task where most decisions are repeated (§4.5.1). Codifying previous decisions also mitigates the difficulty of individuating constructions, a difficulty which is incovenient but not a theoretical threat for SCL (§4.5.3).

BECAUSE also demonstrates that adding overlapping semantic phenomena can make for substantially broader coverage for a given relation type (§5.2) without compromising the reliability of the annotation scheme. Furthermore, make the annotation guidelines less arbitrary, though some semi-arbitrary lines will still need to be drawn (§4.5.4). Annotating such overlaps additionally provides insights into semantic competition for linguistic machinery (§5.2).

9.1.2 Takeaways for automatic tagging efforts

On the automated tagging side, our work on Causeway and DeepCx affirms that automated construction tagging is possible even without deep constructional analysis. Pattern-based methods can tag a wide variety of constructions, but are somewhat brittle (§7.3.2), while neural transition-based methods are more robust (§8.6, §8.7). Neural methods also make it possible to generalize to previously unseen triggers, although they should be permitted to do so with caution (§8.6.4). From our results, it seems that sequence tagging outperforms dependency head expansion for identifying argument spans.

Parse information seems to be less essential to SCL automation than one might think; apparently whatever information would be gleaned from it is represented isomorphically in other forms. In a sense, this is good news for SCL: skipping parsing will make construction tagging methods less reliant on conventional NLP tools, and therefore easier to adapt to more theoretically rigorous construction-based systems.

9.2 Future extensions

9.2.1 Extensions to other linguistic phenomena

Specifically in the realm of causality, it would be interesting to attempt **more fine-grained distinctions** based on Wolff et al.'s (2005) categories of causal language (see §3.1). That includes distinguishing Cause, Enable, and Prevent categories, as discussed in §4.6. It could also include extending the annotations to both the vaguer and the more specific ends of Wolff et al.'s spectrum of causal language, discussed in §3.1. Doing so would require major changes to the annotation scheme, such as allowing for an unspecified Degree and even direction of causation. It would also necessitate finding a way to annotate the effect even when it is implicit in lexical connectives like *kill*.

Beyond English causality, the current annotation scheme is limited with respect to **morphological constructions** such as the comparative *-er*, which would currently have to be anchored to full words. We would like to experiment with different ways of handling such constructions to determine how practical it would be to tag individual morphemes as construction triggers. Likewise, we hope to explore handling constructions with **null triggers**, where simple juxtaposition or linear order can convey meaning. In addition to expanding the range of phenomena that can be annotated in English, these changes would make it possible to do SCL-style annotation in **other languages**, where morphology and implicit relations may play a more prominent role.

Additionally, we would like to extend our annotation approach to **other relation types**. Relations where a similar approach would be appropriate include:

- **Concessions**. A meaning of "despite" can be expressed with diverse constructions, as demonstrated by the examples in Table 4.9: *We headed out **in spite of** the bad weather*; *We value any contribution **no matter** its size*; and *Strange **as** it seems/**though** it **may** seem, there's been a run of crazy dreams!*

  In these constructions, the mapping to our work is straightforward: the concessive connectives take the place of our causal connectives, and the potential obstacle and unprecluded outcome replace the Cause and Effect. A concessive always has exactly two arguments (though one may be in another sentence).

- **Comparisons**. Comparatives take even more wide-ranging forms (e.g., *You're **as** bad **as** my mom*, ***More** boys wanted to play **than** girls*, *I'm two years old**er than** Johnny*), and also frequently interact with ellipsis structures.

  Comparative constructions like *as* __ *as* __ correspond naturally to our causal connectives. However, the argument structure for comparisons is a bit more complex than that of causality: the roles proposed by Bakhshandeh et al. (2016) include Figure, the item being compared; Ground, what it is being compared to; Scale, what attribute is being compared; Differential, an indication of the magnitude of the difference (e.g., *two years*); Domain, an expression of the scope of a superlative (e.g., *the tallest **girl***); and Domain-Specifier, a further restriction of a superlative's scope (e.g., *the tallest girl **in the class***). Hasegawa et al. (2010) suggest a similar repertoire of roles: Attribute, Difference, Item, Item_value, Standard, and Standard_value.

- **Caused motion constructions (CMCs)**. Hwang and Palmer (2015) have already developed annotation guidelines and a corpus for CMCs, and built classifiers that can detect such constructions in text with gold constituency parses. However, they did not attempt to extract the arguments from such constructions. Thus, it would be interesting to apply our approach to more fully analyze each CMC instance.

Because CMCs allow the verbs to vary so widely, the prepositions that introduce the motion paths (e.g., *talk them* ***into*** *coming*) would have to take the place of our causal connectives. This should be feasible, since there is a limited set of such prepositions. Alternatively, if machinery is added for null triggers, a CMC could be marked on the boundary of the verb and the preposition. The arguments could include Agent (the force inducing motion), Patient (the object induced to move), Action (the verb indicating how the motion is caused), and Path (the direction or result of motion).

- **Temporal relations**. Temporal language is far more complex than any of the above categories, and has earned its own suite of annotation schemes, most prominently TimeML (Pustejovsky et al., 2003). Although many temporal assertions are expressed using single-word prepositions (*before*, *after*, *during*, etc.), they also show up as verbs (e.g., *followed*), nouns (*successor*), adverbs (*now*, *yesterday*), and multi-word expressions (*in advance of*). They can even occasionally be gappy, as in ***within*** *two months* ***of*** *her arrival, she had already moved on.*

  It would therefore be interesting to see how much of the temporal language annotated in TimeML could be captured by a frame-oriented, constructional representation like ours, and how much flexibility such a representation would add beyond what is already available in FrameNet and/ or TimeML. The triggers would be temporal constructions like those listed above. Following FrameNet's Time_vector frame, arguments could include Event (the event being anchored), Landmark (the time or event it is anchored to), and Distance (how far away the Event is from the Landmark in a relation of asynchronicity). Instances would of course need to be annotated with additional metadata, akin to our causation types and degrees, indicating what type of temporal relation is indicated (precedence, succession, contiguity, partial contiguity, etc.).

  One complication is that temporal expressions come in several flavors, not all of which relate multiple events or times to each other. Sometimes, for example, they simply set up time anchors that are subsequently treated like nouns, as in *After lunch is the best time for a nap.* Such cases could either be included as construction variants, perhaps with additional metadata indicating their non-relational role, or treated as having entirely separate semantics.

Once SCL is proven for relations rich in constructional variety, a similar approach would be a helpful addition across all semantic domains: while most domains exhibit less variety than causality and comparison, many have the odd idiosyncratic construction that is difficult to capture with conventional tagging methods. Examples of unpredictable constructions in other domains include *what's X doing Y?* (incongruity); *for Y, X is the way to go* (superiority/preference); *jog X's memory* (reminding); verb-*way*, as in *hacked their way out of the jungle* (motion); and *an* in phrases like *60 miles an hour* (rate). As the CxG community has long recognized, such constructions constitute a substantial portion of real-world language use. See Fillmore et al. (2012) for other examples, some reaching into the more grammatical end of the construction spectrum.

In the long run, it would be best for SCL projects not to produce a plethora of standalone, relation-specific corpora. Rather, we would like to see the approach **incorporated into existing broad-coverage annotation schemes**. To that end, we hope to work with the creators of resources like FrameNet and the PDTB to see where aspects of our approach could be helpful for future releases.

9.2.2 Extensions of automation techniques

In this work, we have automated the tagging of only the most important elements of the annotation scheme. Our systems do not produce annotations for degree or type of causation, nor do they mark the presence of overlapping relations. Adding **degree and type tagging** are essential next steps for a fully automated causal language parser. These could be addressed fairly easily by adding either new classifiers, in the case of the pattern-based taggers, or adding transitions that conclude a causal language instance by marking its category (e.g., SHIFT-Motivation). Overlapping relations might best be put off until causal language annotations are more fully integrated into schemes like FrameNet that tag other semantic relations whenever they appear, rather than when they happen to be expressed with language that can also indicate causality.

As discussed in §8.8, we believe the transition-based approach to construction tagging could be easily adapted to most kinds of constructions. Still, as for annotation (§9.2.1) and as discussed above (§8.8), **adapting the transitions to morphological, non-lexical, and cross-sentential constructions** would require significant further research. In addition, a larger repertoire of argument types, as would be necessary for some of the relations proposed above, might make it harder for the tagger to predict argument transitions. This would be especially problematic for phenomena where one word can participate in multiple arguments, in which case it would be impractical to create separate LEFT-ARC$_{x,y}$/RIGHT-ARC$_{x,y}$ transitions for each possible set of overlapping arguments, especially if more than two arguments can overlap. Instead, multiple argument transitions could be applied to the same word, with a separate NEXT-ARG transition to move on to the next word (see §8.3.5).

With regard to improving the neural network's performance, whether on causal constructions or on other domains, three directions stand out as promising for future research:

- **Borrowing techniques from other successful DNNs.** In particular, bidirectional LSTMs (biLSTMs; Graves and Schmidhuber, 2005), which encode a sequence by reading it both forward and backward, have been shown to outperform unidirectional LSTM encodings (e.g., Ling et al., 2015). Neural ATTENTION mechanisms (Bahdanau et al., 2015), in which the network learns to identify which information from a sequence is most relevant at each processing step, are another prominent mechanism that could improve performance.

  It would be particularly worth investigating whether transitions like ours could be integrated into the SLING transition system (Ringgaard et al., 2017), which also happens to use biLSTMs and attention. Such integration would augment a proven semantic parsing framework with the ability to deal with gappy constructions as triggers.

- **Data expansion.** One of the major limitations on our systems currently is the paucity of training data. Performance might improve substantially with training data that has been derived by automatically perturbing our existing data. Similarly, it might help to automatically search an unlabeled corpus for instances that are very similar to existing ones, as Mihăilă (2014) did for BioCause.

- **Detailed analysis of continuing sources of error.** To the extent that the above efforts still yield lackluster performance, it will be essential to figure out where the remaining gap is coming from. Our experience suggests that ambiguity does not seem to be a crucial factor, at least for causal language, but there are many other possible explanations for low semantic parsing scores: lack of world knowledge about what types of entities are typically described as having these relationships; linguistic fluff like conjunctions or LIGHT NOUNS (e.g., *a **bucket** of mussels*) obscuring the true arguments or trigger components; or perhaps just a long tail of diverse triggers. Assessing which factors are important will be crucial to improving generalization.

A final future avenue of work is **automatic construction discovery**. Much work has been done on recognizing novel multi-word expressions (e.g., Schneider and Smith, 2015). This work could be extended to identify new candidate constructions for a given semantic relation, perhaps using a ranked clustering approach. New patterns would need to be identified only at a level of detail at which a linguist could examine them and decide whether to add them to a constructicon.

9.3 Future applications

As elaborated in §4.5.2, the most unambiguous test for how worthwhile an annotation task is is whether the presence of the annotations improves some downstream task (see Riezler, 2014). Below I list a number of such tasks where we hope our approach and annotations will serve useful.

9.3.1 Linguistic applications

In §4.5.3, we saw that attempting to codify and annotate a particular semantic phenomenon can produce collections of "organized observations" (Fillmore et al., 2012), which give linguists the raw material for crafting full construction grammars (or other types of grammars). We hope that our efforts and future SCL projects will continue to help identify the patterns that fuller grammars need to explain.

Our corpus and annotation approach can also help to discover more insights of the sort discussed in §5.2. In particular, further SCL-style annotations would provide valuable information about what kinds of semantic relations tend to be grammaticalized in what ways, both synchronically and diachronically. In addition, the annotations could reveal patterns of linguistic overlaps between different semantic domains (see §5.2.1). This could even lead to automated induction of the SEMANTIC MAPS that are believed by some construction grammarians (e.g., Croft, 2001) to govern how meanings cluster around a finite set of linguistic forms.

9.3.2 NLP applications

On the NLP side, many applications could benefit from better extraction of causal information. Automating analysis of causal language would help question answering (QA) systems identify what snippets of a document to surface when asked a "why" question, or even allow them to extract and stitch together an answer that does not rely on having the user read selections from the original document. We hope such technology will also improve the question interpretation side of QA, helping systems to determine what causes and effects users are asking for or about.

In machine translation, we would like to investigate whether causal language annotations could improve alignments and improve the odds that an appropriate construction is chosen given the causation type and degree of the original. This becomes particularly important when a causal relation was piggybacked on an overlapping relation such as temporal precedence, in which case the translator should ensure that the same implications are conveyed in the target language. Similarly, paraphrasing and simplification systems should retain as much as possible of the causal intent of the original. We hope to explore whether incorporating our annotations and systems can improve scores in these systems.

Finally, much work has been done on cause and effect specifically in the biomedical domain (Mihăilă, 2014). It is worth exploring how domain-sensitive this work is: it is possible that our domain-general approaches capture the vast majority of the same information, but require less expert input. On the other hand, biomedical texts and other specialized fields may have too much context-specific jargon about causality that makes it hard to apply general tools.

Generalizing beyond causal language, we plan to incorporate the "surface construction labeling" approach into other application areas. As previously discussed, relations like concession and comparison are ripe for constructional analysis, and such analysis would improve any application that relies on semantic parsing, including information extraction, question answering, dialogue systems, and more.

As this thesis has shown, systems based on SCL can handle a wider range of linguistic forms than conventional approaches. Accordingly, we hope the approach will ultimately empower NLP tools to interpret an ever-expanding landscape of meanings, improve performance on existing applications, and perhaps even open up new ways that language technologies can help us live our lives.

Appendices

APPENDIX A

Background Terminology and Concepts

A.1 Notes on inter-annotator agreement

In linguistic annotation, the equivalent of shooting from the hip is annotating haphazardly—marking up text with categories and relationships based on gut instinct, rather than taking care to be systematic and consistent. Inconsistent annotations yield misleading linguistic conclusions, and systems trained on such annotations are not only unreliable but also unhelpful, as their outputs may not correspond to any real phenomena.

To ensure that humans, at least, have converged on a consistent analysis of the phenomenon under study, annotation scheme designers measure INTER-ANNOTATOR AGREEMENT, also known as INTER-CODER or INTER-RATER AGREEMENT. Inter-annotator agreement is simply the degree of overlap between two or more annotators' independent attempts at marking up the same text. (How essential or informative such measures really are is an issue discussed in §4.5.2.) The same measures can sometimes also be used to evaluate how closely a system's outputs correspond to the correct answers.

Typical metrics treat one set of annotations as the "gold standard" against which the other annotator is compared. It makes no difference which is which; most measures are symmetric.

Different agreement metrics are appropriate depending on the type of data being compared. **Accuracy**, applicable to any kind of data, is simply the percentage of correct answers. But for NLP, we typically want to drill down to more fine-grained measurements. In particular, there are often a large or even infinite number of annotations an annotator could have made but did not; nobody is impressed when, say, dozens of non-name words are correctly left untagged as names.

One common type of annotation data is where two annotators must find instances of some phenomenon—e.g., personal names or uses of causal language. The most widely used metrics for such data are calculated from three simple counts: true positives (TPs), false positives (FPs), and false negatives (FNs). Common metrics include:

- **Precision:** the fraction of the instances tagged by the annotator that were correct, defined as:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

  Precision measures how well an annotator avoided FPs.

- **Recall:** the fraction of the instances the annotator should have tagged that they did tag, defined as

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

  Recall measures how well an annotator avoided FNs.

- **F_1 measure:** the harmonic mean of precision and recall, defined as:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$

  F_1 reflects a balance between avoiding FNs and avoiding FPs, and is therefore usually seen as the most important metric for this type of data.

A second type of annotation is classifying items into one of k categories. In categorization, accuracy can look artificially high if many of the instances are from the same category: annotators may be agreeing often simply by chance, rather than because they both make the same distinctions between categories. It is more informative to check how much better they do than chance, a value measured by Cohen's kappa:

$$\kappa = \frac{A - p_e}{1 - p_e},$$

where A is accuracy and p_e is the probability of chance agreement given each annotator's distribution of labels. Kappa scores range from -1 (full disagreement) to 1 (perfect agreement), with 0.6 usually considered moderate agreement and 0.8 or above considered excellent.

The final type of metric we use measures how well two annotated spans overlap with each other when they do not match exactly. The Jaccard index (Jaccard, 1912), less commonly used than P, R, and F_1, is defined as

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

where A and B are the sets of tokens in the two annotated spans.[1] Unlike some previous works, we report Jaccard indices not on sets of unique word forms but on sets of word indices (i.e., their positions within their containing sentence). Thus, repeated words within a span do not artificially alter the Jaccard index.

A.2 Standard NLP tools

A.2.1 NLP tagging tools we rely on

The backdrop to our work is a large constellation of existing NLP tools for basic linguistic analysis. These tools are foundational to the operation of many modern NLP systems, to the extent that systems that do not rely on them advertise that fact as an unusual efficiency feature. Below I review some of these key terms and tools.

[1]The Jaccard index was developed to measure overlap between two sets—originally, in fact, sets of vegetation species in the alpine zone. But groups of words in a text make perfectly good sets, too.

Basic machine learning terminology While much of the terminology for machine learning techniques described in the thesis can be glossed over if it is unfamiliar, one term deserves special attention. The word FEATURE has a unique meaning in the context of machine learning: it describes how a discrete, potentially complex object like a word is broken down into a vector of numerical inputs for an algorithm (see, e.g., Fu, 1968). If the task is to tag, say, personal names in context, the algorithm will not be able to generalize over examples if each word is represented just as one formless entry among hundreds of thousands in the English dictionary. Instead, each word is FEATURIZED, or converted into a vector of "features": its length in characters, a binary variable for whether the word's first letter is capitalized, another for whether it is in a known list of names, the fraction of the way through the sentence where the word appears, and so on. These feature vectors are what machine learning algorithms such as classifiers actually operate on. A function that produces an entry in the feature vector for a given input is known as a FEATURE FUNCTION or FEATURE EXTRACTOR.

Sentence splitting and tokenization What defines a word or a sentence may seem intuitively obvious, but as with most linguistic phenomena, real-world examples quickly defy expectations. As mentioned in §6.2.1.1, we use the Stanford CoreNLP pipeline (Manning et al., 2014) to automatically determine where words and sentences start and end. The conventions used are generally straightforward, but two aspects are worth noting. First, the TOKENIZER—the subsystem that splits a document into individual words—splits contractions and punctuation off as their own "words" (e.g., *"You can't"* would become the list [*", You, ca, n't, "*]). Second, the sentence splitter treats sentence-final punctuation within a sentence as splitting the sentence in two. For example, *"No way!" he complained* would be treated as two separate sentences, even though the quotation is a sentence within a sentence.

Part-of-speech tagging One of the most fundamental pieces of information about a word is its part of speech (POS): singular noun, plural noun, present-tense verb, etc. Again, the categories here are blurrier than they might seem, but the most popular inventory of POS tags is that of the Penn Treebank (Marcus et al., 1994). Although ambiguous words sometimes still trip up POS taggers, they have achieved very high accuracy (frequently above 97%) on automatically tagging English words in context.

Lemmatization A given word can have many different morphological variants depending on factors like tense or number (singular/plural): *run* has variants like *running*, *ran*, *runs*, and so on. Rather than treating these variants as completely separate words, it is often useful to bring them under a single umbrella as variants of the base form. That base form—e.g., *run*—is the word's LEMMA.

An automated LEMMATIZER looks at each word in a sentence and tries to determine the most appropriate lemma for each given its context. English lemmatizers are also highly accurate, although they can occasionally be fooled by homonyms (e.g., the lemma of *left* could be *left* or *leave*).

Named entity recogition One of the most well-studied tasks in NLP is finding names of people, places and things (see, e.g., Nadeau and Sekine, 2007)—NAMED ENTITY RECOGNITION, or NER for short. Most commonly, the task is formulated as finding and labeling each proper noun, including multi-word names, with one of Person, Location, Organization, and Other. NER systems are quite reliable, with the best algorithms reaching performance levels competitive with humans. NER tags are often useful as features for downstream algorithms.

Dependency parsing for syntax The most complex pieces of NLP machinery we rely on are tools that automatically analyze the syntactic structure of a sentence. This analysis is known as SYNTACTIC

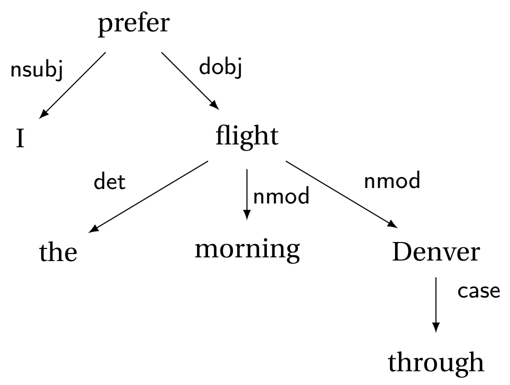


Figure A.1: A UD parse tree for the sentence *I prefer the morning flight through Denver.* (Example from Jurafsky and Martin [2017].)

PARSING, or sometimes just PARSING. For decades, efforts to analyze syntax used Chomsky's theories of phrase structure grammar, which treat sentences as being built out of modular phrases. In recent years, however, a different paradigm called DEPENDENCY PARSING has become far more popular in NLP, particularly for applications where identifying semantic relations in a given text is important.

In dependency parsing, or DEPENDENCY GRAMMAR, the structure of a sentence is modeled as a set of dependency relationships between individual words. The syntactic formalism we use throughout this work is Universal Dependencies (UD; Nivre et al., 2016), which aims to maximize the relationships between content words and to relegate grammatical function words (prepositions, conjunctions, etc.) to lower levels of the tree. An example UD parse is shown in Figure A.1. At the root of the tree is the main verb of the sentence.[2] The children of that verb are the main content words, or HEADS, of the verb's arguments and modifiers (in this case *I* and *flight*). Likewise, *flight* is modified by its children *the*, *morning*, and *Denver*, and *Denver* is modified in turn by *through*. Thus, the tree represents the fact that *flight* is the head of the phrase *the morning flight through Denver*. With such tree structures, UD parse trees capture the syntactic structure of a sentence in a manner that highlights the key semantic relationships—e.g., the relation prefer(I, flight).

Each edge in the tree, or DEPENDENCY ARC, is usually assigned a label that describes the relationship between the head and the dependent. For example, in Figure A.1, the nsubj label between *prefer* and *I* indicates that *I* modifies *prefer* as its nominal subject. See the UD documentation for the scheme's complete inventory of English dependency relation types.

A.3 Background on deep learning

DEEP NEURAL NETWORKS, the approach that underlies DeepCx (Chapter 8), is behind many recent advances in artificial intelligence. Loosely inspired by the brain, a DNN contains thousands or even millions of simplified artificial "neurons." Each neuron takes some inputs, performs a simple calculation based on a few internal numerical parameters, and passes on the result to the next layer of neurons.

The network is trained to perform a particular task—say, recognizing company names—by looking at thousands of training examples. As it sees each example, the software twiddles the parameters of

[2]For computational convenience, a symbolic ROOT node is often placed as the parent of the main verb at the actual root of the tree.

each neuron to improve the output of the overall network. With sufficient training, the interactions between neurons enable the network as a whole to spot patterns like name capitalization or the rarity of strings of adjacent company names. The network essentially learns to extract high-level features from the raw input without extensive feature engineering. The training process, and the use of DNNs more generally, is known as DEEP LEARNING.

A.3.1 Neural networks and LSTMs

The first major line of research underlying DeepCx is artificial neural networks (ANNs), which go back to the simple "perceptrons" of the 1950's and 60's (Rosenblatt, 1958). The basic idea is simple: in an extremely stylized analogue of brain cell networks, each "neuron" in an ANN takes in a set of numerical inputs. It computes a weighted sum of those inputs, including a bias term, then passes the sum through some nonlinear function. The result is output to the next layer of nodes in the network (or, in the case of the final layer, makes up one of the network's overall outputs). The network's behavior is determined by all the weights used in the individual neurons' weighted sums. Accordingly, the network is trained by tuning these weights.

The earliest ANNs used a step function for the nonlinearity, while later versions used sigmoids or hyperbolic tangents as differentiable approximations of a step function. Modern neural networks, including DeepCx, frequently use nonlinearities consisting of RECTIFIED LINEAR UNITS (ReLUs; Nair and Hinton, 2010), defined by the equation:

$$f(x) = \max(0, x)$$

A key feature of the nonlinear function is that it must be differentiable. The reason for this is the algorithm that made multi-layer networks tractable to train: BACKPROPAGATION (Rumelhart et al., 1988). The backpropagation algorithm runs the network on an input, then updates each weight in the network according to its contribution to the error in the final layer's output. Backpropagation works by finding the partial derivative of the error with respect to each parameter, essentially tracing responsibility for the errors back through each layer of the network. Mathematically, the algorithm is equivalent to gradient descent on the model parameters with respect to the error.

Over the past few decades, as computers' processing power improved and datasets have grown, researchers have been able to add more and more layers to neural networks while retaining the ability to train them. Such "deep" networks now set the standard of performance for many AI tasks, including in NLP. Figure A.2 shows a sample DNN architecture.

By analyzing these networks both mathematically and empirically (see, e.g., Hinton and Salakhutdinov, 2006), researchers have found that one key to their success is that each layer re-represents the input in a new way. A set of n numbers (e.g., pixel intensities) fed into to the network can be thought of as an n-dimensional vector representation of the input. When the first layer processes this vector and outputs its own set of m numbers, it has effectively re-represented the input in a new, m-dimensional vector space. As the information approaches the final layer, the vector representations, or EMBEDDINGS, get successively higher-level: in a facial recognition network, for example, layers come to represent the image not as a vector of raw pixel intensities, but as a vector of intensities of local facial features like eyebrows and nostrils. The network learns to make the embeddings closest to the output layer highly informative for the actual task the network is trying to perform. From the 10,000-foot view, learning to generate these embeddings is the true task of a neural network.

Often, an embedding learned for one task will turn out to capture core properties of the inputs being embedded, and thus be useful for other problems, as well. In NLP, words are frequently embedded as different vectors in a high-dimensional space (usually several hundred dimensions). Word embeddings

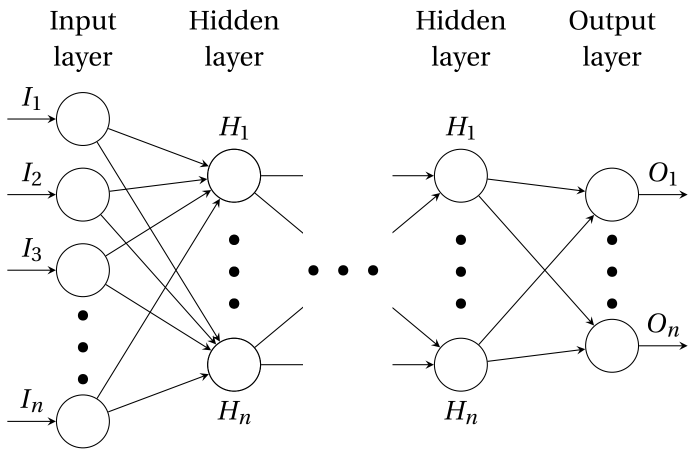


Figure A.2: A schematic of the architecture of a deep neural network.

.

learned for, say, predicting the linguistic contexts in which a word appears end up reflecting many aspects of the words' semantics: the vectors of words with similar meanings are close to each other in the embedding space (see, e.g., Mikolov et al., 2013). This makes word embeddings an excellent way to represent a discrete word as a continuous input to a new neural network. Similarly useful embeddings can be learned for the members of other discrete sets, such as part-of-speech tags.

The description above presents neural networks in their most generic form; of course, there are all manner of variations on the theme. Instead of weighted sums, network nodes can perform multiplications or take the max of their inputs. Instead of fully connecting every node in one layer to every node in the next layer, a node can pull from a limited pool of neurons in the previous layer, or even from shortcut connections to earlier layers. Various sets of nodes in the network can all share the same weight parameters. Any differentiable function of inputs and previous layers is fair game, and much of modern AI research is a cottage industry of designing and tweaking neural network architectures.

A.3.2 Recurrent neural networks: an essential tool for NLP

Applying a neural network to a fixed-size input, such as $k \times k$-pixel image, is relatively straightforward. But when the input is an arbitrary-length sequence, such as a time series or a natural-language sentence, we need some additional machinery. We would like to be able to apply the same input processing to each sequence element (e.g., each word) in turn, but also to have the network respond differently depending on what it has seen so far. The class of neural network architectures that accomplishes this is known as RECURRENT NEURAL NETWORKS (RNNs), visualized schematically in Figure A.3.

Conceptually, an RNN has a repeating slice of network which is applied to each input. This is represented by the h's in Figure A.3. Each time that slice is applied a new input (a new x), the network is extended by another copy of h. The repeating module produces an output for the current input (a y), as well as some state information for future layers.

Two key features distinguish this temporally "unfolded" network structure from a non-recurrent ANN. First, in addition to the next sequence input, each slice receives some information from the previous slice. This information constitutes the network's "memory" of what it has seen so far, enabling it to alter its behavior depending on previous sequence elements. Second, every slice shares the same

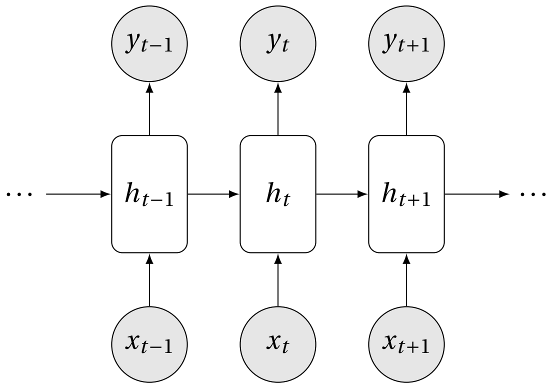


Figure A.3: A simple schematic of an RNN architecture.

.

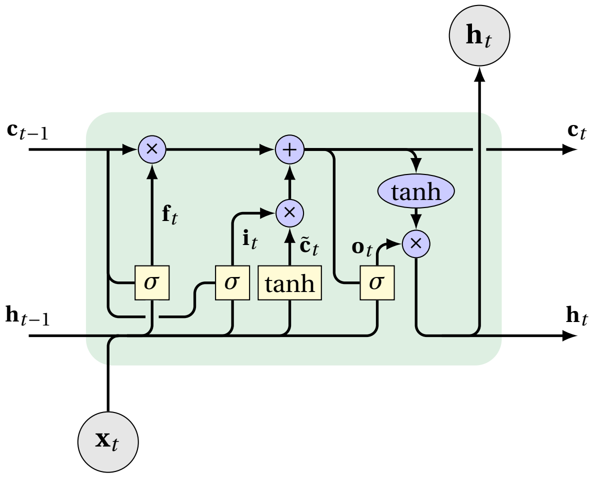


Figure A.4: A schematic of an LSTM neural network cell, based closely on Chris Olah's LSTM walk-through at http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Blue circles represent pointwise operations, while yellow boxes represent neural network layers (i.e., weighted sums with nonlinearities).

weight parameters, which keeps the number of parameters fixed and finite and allows the network to respond similarly to similar inputs at different locations in the sequence.

A.3.2.1 LSTMs allow RNNs to learn long-distance patterns

RNNs are effectively very deep networks, adding at least one layer per input item. In principle, their memory allows them to react to information even from many steps back in the input sequence. In practice, however, their depth subjects them to the VANISHING GRADIENT PROBLEM (see Hochreiter et al., 2001): as responsibility for errors during a training cycle propagates backwards through the network, the earliest layers, corresponding to the earliest inputs, receive an ever-smaller portion of the blame. Inevitably, the network either ignores previous inputs in favor of the most recent ones, eliminating the benefits of recurrent structure, or it takes an extremely long time to learn from infinitesmal error signals how to encode the relevant extended history. Thus, the network becomes unable to, say, conjugate a verb to agree in number with a long subject.

The type of RNN used by DeepCx and many other NLP projects, known as a LONG SHORT-TERM MEMORY (LSTM) network (Hochreiter and Schmidhuber, 1997), overcomes the vanishing gradient problem by allowing information to propagate forward until it is explicitly replaced. An LSTM maintains a "cell state," a fixed-size vector that carries forward information from previous steps, frequently without much modification. An LSTM's repeating module takes in three inputs at each step t: $\mathbf{c}_{t-1}$, the cell state after the previous step; $\mathbf{h}_{t-1}$, the cell's output from the previous step; and $\mathbf{x}_t$, the input at position t in the input sequence.[3] The LSTM cell's computation proceeds in six steps (shown graphically in Figure A.4):

1. A FORGET GATE decides what entries in the previous state should be forgotten. The forget gate is a neural network layer characterized by the equation

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f\left[\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_{t-1}\right] + \mathbf{b}_f\right),$$

   where σ indicates a componentwise logistic sigmoid function ($f(x_i) = (1 + e^{-x_i})^{-1}$) and $\mathbf{W}_f$ is a weight matrix. The sigmoid outputs a continuous value between 0, which indicates that the corresponding cell state entry will be completely replaced, and 1, which indicates that it will be left unchanged.

2. The cell's INPUT GATE determines which entries in the cell state should be updated. The equation for the input gate is:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i\left[\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_{t-1}\right] + \mathbf{b}_i\right)$$

3. Based on the current input and previous output, the cell computes the new values that might be used to update the cell state:

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_c\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}_c\right)$$

4. The results of the previous steps are used to remove old information from the cell state and add the new information ($\tilde{\mathbf{c}}_t$) to it:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \tilde{\mathbf{c}}_t,$$

   where $\odot$ indicates the componentwise (Hadamard) product.

5. The OUTPUT GATE decides on selections from the cell state that should constitute the cell's output for the current timestep:

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o\left[\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_t\right] + \mathbf{b}_o\right)$$

6. Finally, the LSTM's output is computed by applying the output gate to the cell state:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right)$$

The variant of LSTM gates presented here is that of Gers and Schmidhuber (2000), which is also used by Dyer et al. (2015), on which we base our work. Minor variations abound, including restricting the gates from seeing the cell state.

In our work, we interpret the output of an LSTM as an embedding of the list of items it has seen so far. This gives us a fixed-size, continuous-space vector summary of a list. Thanks to the LSTM model, this embedding is able to prioritize information from early in the input sequence when appropriate. Often such embeddings are used to summarize a sequence of words from the input sentence—e.g., an English sentence that is about to be translated into French. But an LSTM is not limited to summarizing

[3]Throughout this chapter, we use the standard notation of bolding vector and matrix values.

segments of input text; it can also encode, say, the list of tokens that have been added so far to an argument span, or the list of dependency arcs between two nodes in the parse tree. See §8.4 for more on how DeepCx uses such summaries.

APPENDIX B

BECAUSE Annotation Guidelines

B.1 Purpose and sketch of the corpus

The corpus produced by this annotation effort will capture the cause-and-effect relations that are part of the textual meaning of the documents in the corpus. This information will be useful for two related lines of work:

- On the computational linguistics side, the corpus will yield insights as to how causality is expressed in English—e.g., how often certain types of constructions are used. Because causality is not a linguistically isolated phenomenon, the corpus will also indicate the presence of semantic relationships that share the same linguistic machinery as causation.
- The corpus will also serve as a gold standard for a computational system that identifies and classifies causal relations in text. The tool is intended to enable downstream systems to combine semantic representations of short clauses or phrases in a meaningful way.

  For instance, consider the sentence *Lack of oversight led to the economic collapse.* The system should be able to identify that the relationship between *lack of oversight* and *economic collapse* is presented as one of natural cause and effect, and that the cause is presented as leading directly to the effect. On the other hand, the sentence *his ambiguous responses kept her from giving up* should be interpreted somewhat differently: the system should identify the relationship between the two clauses as one of motivation (still causal, but explicitly involving an agent's decision-making powers), and with the cause inhibiting the effect.

To define the set of possible linguistic patterns that can indicate causation by our definition, we use an approach based on corpus lexicography: we maintain a catalog of all linguistic patterns known to indicate causation. The primary task of annotators is thus to identify and mark up instances of those known linguistic patterns. A secondary task is to look for language that matches our definition of causal, but that is not currently captured by any pattern in the catalog. Annotators should propose new patterns based on such instances.

B.2 Anatomy of a causality annotation

Once an instance of a known causal pattern has been identified, up to four components should be annotated: the Cause, the Effect, the Means of causation, and the language that indicates the cause-and-effect relationship (the connective). Each annotated instance should include some additional classification metadata. It should also indicate the presence of any overlapping phenomena of interest; see §B.2.4.

B.2.1 Annotatable units

For each instance of causal language, one element is mandatory:

1. **The causal connective:** the word or series of words that encodes the causal relationship. In general, the connective will consist of a fixed construction with some open slots (e.g., __ *because* __). For each pattern, the words that should be included in the connective are indicated in the catalog.

   Any modifiers of the connective should not be included in the annotation. For example, if the connective is *severely inhibits*, only *inhibits* should be annotated.

   Every connective should also be annotated with the instance's classification information—the type (see §B.2.2) and the degree of the causation (see §B.2.3)—as well as the overlapping semantic relationships that are present (see §B.2.4).

The arguments of the causal connective, each of which may or may not be present, form the rest of each annotation.

2. **The Effect:** the span of the sentence presented as an outcome. It should be either a complete phrase or clause, though the clause may be non-finite (i.e., the subject may be missing and the verb may be a participle, gerund, or infinitive).

   See §B.4.2.9 for the one case in which the Effect argument need not be a complete phrase or a clause.

3. **The Cause:** the span of the sentence presented as an event, state, entity, or action that produces the effect. This should also be either a complete phrase or a clause.

4. **The Means of causation:** when the Cause (explicit or implicit) is an agent or entity, the activity by which the Cause produced the Effect may also be specified. This is the Means. For example, in *The men kept the fire from spreading by clearing large ditches around it*, *the men* should be annotated as the Cause, and *clearing large ditches around it* as the Means.

   Conceptually, the Means is part of the Cause (e.g., it was the men clearing large ditches that kept the fire from spreading). However, we annotate it separately because Means clauses make use of distinct linguistic machinery.

   Means arguments should be annotated only for:

   - *by* or *via* clauses
   - dependent clauses like ***Singing loudly**, she caused pain for everyone around*
   - *how* phrases (e.g., ***That**'s how I almost caused a war*)
   - *when* phrases (e.g., *I started a fire when **I dropped the match***)

  - Purpose instances where the Cause (i.e., the motivation) is itself a causal instance (e.g., *I **spoke with him** to prevent him from getting mad*, where *I* and *spoke with him* are the Cause and Means, respectively, for *prevent*)
  - constructions like *use* ⟨*means*⟩ *to cause* ⟨*effect*⟩ and ⟨*cause*⟩ ⟨*does means*⟩ *to cause* ⟨*effect*⟩ (e.g., *The stripes and polka dots **combine** to create a sense of discord*).

  Other cases that seems like Means should be raised for discussion.

In general, no portions of the connective will need to overlap with one or the other of the arguments. For example, *caused the glass to fall* should be annotated with *cause/to* as the two fragments of the connective, and *the glass/fall* as the two fragments of the Effect. See §B.4.2.9 below.

B.2.2 Classifying types of causal language

We distinguish three different types of causal language. The decision tree below should be used to classify them, but here we give the rough intuition for each. A causal statement where *C* (either a cause or a causer) is presented as the cause of *E* is an instance of:

- Consequence if it asserts that *C* happens/exists/is the case, which naturally leads to *E*.

  In an instance of Consequence, the Effect is presented as the end of some chain of events starting with the Cause, without the deliberate intervention of any agent. There may in fact be an agent in this chain of events, but the language used does not highlight that entity's agency. For instance, *John drinks a lot because he had a troubled childhood* presents the drinking as a natural consequence of John's past, even though John could in theory choose to drink or not drink.
- Motivation if it asserts that some agent *A* perceives *C*, which leads to *A* choosing *E*, and *E* describes *A* taking some action; or if it asserts that *A* perceives *C*, which leads to *E*, and *E* describes *A* consciously feeling or thinking a particular way. (*A* need not be explicitly mentioned in *E*; for example, *E* may be a passive clause.)

  For instance, *John drinks a lot because he loves the taste of beer* would generally be interpreted to imply a causal sequence where John perceives the taste of beer and accordingly chooses to drink a lot of it.
- Purpose if it asserts that some agent *A* chooses to do *E* out of a desire to make *C* happen/be the case. (Again, *A* need not be explicitly mentioned.)

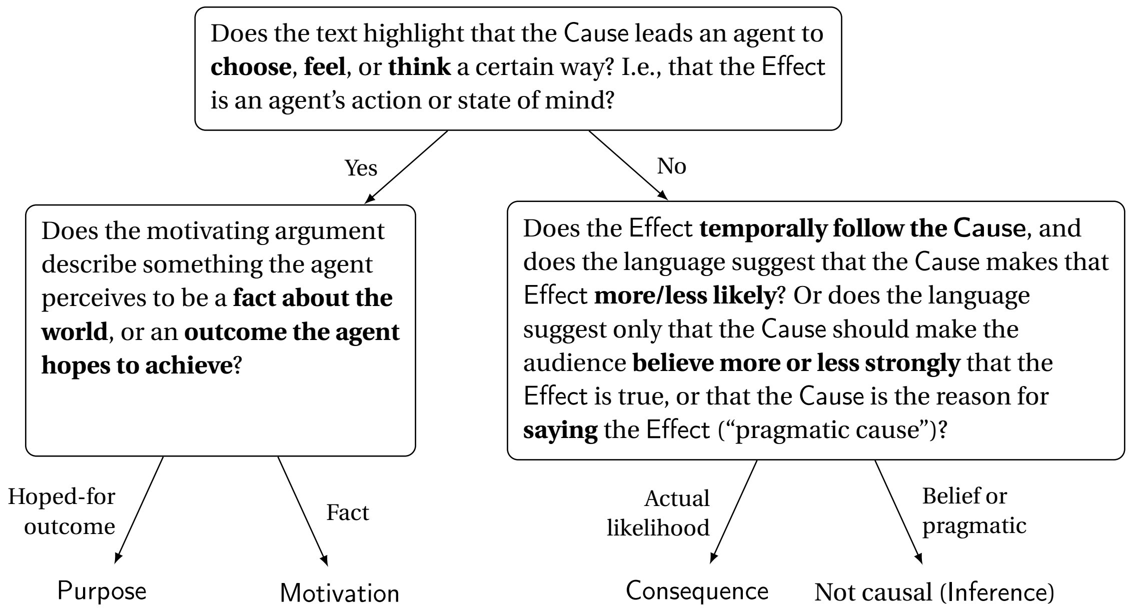
  For example, *John drinks a lot to drown his sorrows* is an instance of Purpose. In general, it is felicitous to say of an instance of Purpose that the desired effect did not in fact take place (e.g., *John drinks a lot to drown his sorrows, but his sorrows persist nonetheless*).

In some cases, of course, the Cause will be described as making the Effect **less** likely, e.g., when the connective is a verb like *prevent* or *inhibit*.

See §B.4.1.6 for cases where language of causation is used for a non-causal meaning.

B.2.2.1 Decision tree for causation classification

Once the causal connective and the arguments have both been identified, the following decision tree should be followed to determine the causal class a given instance belongs to:


Does the text highlight that the Cause leads an agent to **choose**, **feel**, or **think** a certain way? I.e., that the Effect is an agent's action or state of mind?
Yes
No
Does the motivating argument describe something the agent perceives to be a **fact about the world**, or an **outcome the agent hopes to achieve**?
Does the Effect **temporally follow the Cause**, and does the language suggest that the Cause makes that Effect **more/less likely**? Or does the language suggest only that the Cause should make the audience **believe more or less strongly** that the Effect is true, or that the Cause is the reason for **saying** the Effect ("pragmatic cause")?
Hoped-for outcome
Fact
Actual likelihood
Belief or pragmatic
Purpose
Motivation
Consequence
Not causal (Inference)

Note that for the purposes of the first question, the agent need not be explicitly mentioned in the text. What matters is whether the presence of an agent **and their capacity for making decisions or having mental states** is emphasized.

A good sanity check for whether Motivation or Purpose is appropriate is whether the sentence can be rephrased as *X motivates Y* (for Motivation) or *the desire for X motivates Y* (for Purpose).

When applying this decision tree, annotators should keep in mind two facts about arguments:

- For cases where the Cause is hindering the Effect, references to "the Effect" should be taken to refer to the lack of effect.
- When only the causer is present, without an explicit cause, it is the time of the **implicit cause** (the action) that should be considered for the right side of the tree.

B.2.2.2 The structure of Purpose

Determining the Cause and Effect in an instance of Purpose can be confusing. Consider the statement *John drinks to drown his sorrows*. In this case, *John drinks* is the Effect, and *drown his sorrows* is the Cause. This may be counterintuitive, given that drowning his sorrows is the hoped-for effect of drinking. To understand the causal relationships in this sentence, we need a clearer picture of the structure of a purposeful action.

In a typical case of purposeful action, there is some agent A who desires some state S. Because of this desire, A takes some action X. Subsequently, if this action achieves its intended purpose, then S in fact obtains. The structure is depicted below:

$$A \text{ desires } S \text{ and } A \text{ believes } X \text{ will cause } S \rightarrow A \text{ performs action } X \rightarrow S \text{ may obtain}$$

A statement may focus on either causal link in this chain. The statement *John drinks to drown his sorrows* is emphasizing the first link—i.e., it is stating that there was a causal link between John's desire to drown his sorrows and his drinking. It is for this reason that *drown his sorrows* is considered the

Cause in this case—it describes a desired state, not an actual state of the world. The second link could be captured by a statement such as the unlikely *John drank a lot, so his sorrows vanished.*

Accordingly, when considering the criteria for determining the degree of causation, annotators should consider the link between the desire for S and the action: if desiring S encourages A to do X, then it is facilitation; etc.

B.2.2.3 Distinguishing Motivation from Purpose

Motivation and Purpose are often semantically very similar—both highlight an agent's choice and the reason(s) for it—but they are expressed with slightly different language:

- In an instance of Motivation, the event or state that leads to the agent's choice (possibly including the agent's state of mind) is stated explicitly, and that event/state has happened/is the case already at the time the agent experiences the motivation.
- In an instance of Purpose, the agent's reason must specifically be a desire to obtain a certain outcome, and this desire must be implied only by the syntax relating the agent's choice and the desired state.

Thus, for example, *John wanted to drown his sorrows, so he drank* is an instance of Motivation, because John's wanting is an explicitly stated state that exists before drinking. *John drank to drown his sorrows*, on the other hand, is an instance of Purpose, because drowning the sorrows has not yet happened; it is merely the hoped-for result.

A good test for whether a connective is a Purpose connective (i.e., another way to express the bottom left test in the decision tree) is whether the connective can be rephrased as "in the hopes of."

B.2.2.4 Distinguishing Consequence from Motivation

Some instances of Consequence are straightforward, concerning only non-agentive entities. In many cases, however, Consequence and Motivation are semantically close: like Consequence, the Effect in an instance of Motivation can involve an agent performing an action. The distinction in such a case is how the process connecting the Cause and Effect is framed. If an instance is framed as a natural process, with the Effect following from the Cause as a simple outcome of the laws of nature, it is a Consequence. If, on the other hand, the framing of the relationship highlights the decision-making capability of an agent or requires a conscious agent to make sense, it is an instance of Motivation.

For example, *John drank because his balance of neurotransmitters left him in a depressed mental state* is an instance of Consequence, whereas *John drank because he loves beer* is an instance of Motivation.

Thus, the first question in the decision tree should be answered with respect to how the causal relationship is framed—whether the framing emphasizes agentivity—not purely in terms of whether an agent is present.

B.2.2.5 Distinguishing causal Purpose from non-causal purposes

Similar language is sometimes used to express both causal purposes and non-causal purposes. For instance, *I want this chair* ***for*** *sitting in* is causal language of the Purpose variety, but *this chair is* ***for*** *me to sit in* is not. Likewise, *We wrote this paper* ***to*** *make our views known* is causal, but *The purpose of this paper is* ***to*** *make our views known* is not.

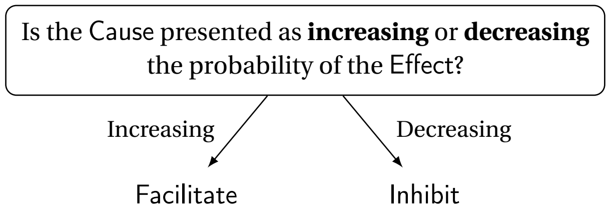
The key distinction lies in the first question in the decision tree above. If the language emphasizes that the Effect is an **agent's action or state of mind**, and they are taking that action or having that state of mind in the hopes of achieving a certain outcome, the language is causal. If, on the other hand, the language simply expresses that an item exists to serve a particular role (purpose of existence), that is not the sort of purpose we are interested in: it is not an agent being motivated by a desire to achieve a goal.

A corollary of this is that attachment can determine whether a potential instance of Purpose is in fact causal. For example, consider the sentence *I catch fish to eat*. If the intention is that the author catches fish which are for eating—i.e., *I [catch [fish to eat]]*—it would not be causal. However, if the author means that the reason she catches fish is so she can eat—*[I catch fish] [to eat]*—it is causal. (See §B.4.1.§4 below for what to do if the sentence is truly ambiguous in context.)

Note, however, that if the Effect is an action, particularly if it is a nominalized verb, purpose of existence does count as Purpose—for example, *The consumption of protein to build muscle is not as helpful as commonly believed*. The test for this case is whether the Effect can be paraphrased using *the act of*.

B.2.3 Classifying degrees of causation

Causal relationships exist on a spectrum, with the Cause making the Effect inevitable on one end and the Cause making the Effect impossible on the other. For the purposes of this corpus, we distinguish two different degrees of causation:


Is the Cause presented as **increasing** or **decreasing** the probability of the Effect?
Increasing
Decreasing
Facilitate
Inhibit

B.2.4 Annotating overlapping phenomena

The constructions used to express causation overlap with many other semantic domains. For example, the *if/then* language of hypotheticals and the *so* ⟨*adjective*⟩ construction of extremity have become conventionalized ways of expressing causation, usually in addition to their other meanings. In this corpus, we annotate the presence of these overlapping relations, as well.

An instance of a causal construction should be annotated as something—an instance of either causal language or a non-causal overlapping relation—any time the word sense and linguistic structure allow it to be coerced into a causal interpretation, and the meaning is either causal or one of the relation types below. Consider, for example, the connective *without*. It should be annotated in *Without your support, the campaign will fail*. However, annotators should ignore uses like *He left without saying goodbye*, because in this linguistic context, *without* cannot be coerced into a causal meaning.

All relation types present in the instance should be marked. For example, *so offensive that I left* would be annotated as both causation (Motivation) and Extremity/Sufficiency. When causality is not present in a use of a causal construction, the instance should be annotated as Non-causal, and the overlapping relations present should be marked.

All overlapping relations are understood to be between an Arg0 and an Arg1. When annotating a causal instance, Arg0 and Arg1 refer to the Cause and Effect, respectively. When annotating a non-causal instance, Arg0 and Arg1 refer to the arguments that would be Cause and Effect, respectively, if the instance were causal.

The following overlapping relation types should be annotated:

- **Temporal:** mark as present if the causal construction explicitly specifies a temporal order between the arguments (e.g., *once*, *after*) or simultaneity (e.g., *as*, *during*). Should **not** be marked when a temporal preposition sets up a phrase that functions like an NP, rather than relating two events (e.g., *My usual bedtime is after 2 AM*).
- **Correlation:** mark as present if the core meaning of the causal construction is that Arg0 and Arg1 vary together (e.g., *the more... the more...*; *as*).
- **Hypothetical:** mark as present if the causal construction explicitly introduces a hypothetical scenario, imagining that a questionable premise is true and then establishing what would hold in the world where it is (e.g., *if... then...*).
- **Obligation/permission:** mark as present if Arg1 (Effect) is an agent's action, and Arg0 (Cause) is presented as some norm, rule, or entity with power that is requiring, permitting, or forbidding Arg1 to be performed (e.g., *require* in the legal sense, *permit*). Words that can only ever mean legal obligation or permission but do not also imply causality, such as *bar* or *authorize*, should **not** be annotated.
- **Creation/termination:** mark as present if the causal construction frames the relationship as an entity or circumstance being brought into existence or terminated (e.g., *create*, *eliminate* [rarely causal]).
- **Extremity/sufficiency:** mark as present if the causal construction also expresses an extreme or sufficient/insufficient position of some value on a scale (e.g., *so* __ *that* __; *sufficiently* __ *that* __).
- **Context:** mark as present if the construction indicates important background information that clarifies the context of the Effect (e.g.,. *with*, *without*, *when*). The construction should not indicate the nature of the relationship beyond that recurrence/co-occurrence—e.g., it should not indicate temporal order or a hypothetical. (It may, however, metaphorically borrow the language of location or simultaneity, as long as it does not actually indicate a temporal or spatial relationship: *Where CEOs cozy up with regulators, corruption abounds.*) Should **not** be annotated for one person or thing accompanying another (*With a sigh, she departed*; *She left without her son*).

B.2.5 Determining when an overlapping relation is also causal

When annotating connectives such as *after* or *if/then*, it can be difficult to determine if a causal meaning is present in addition to the temporal, hypothetical, or other core meaning. The key question to answer is **whether the speaker intended to state a causal relationship**. When this is unclear from the text, annotators should consider the following questions to clarify their intuition about the instance:

- **The "why" test:** After reading the sentence, could a reader reasonably be expected to answer a "why" question about the potential Effect argument? If not, it is not causal. For example, after reading *I felt better after I had a drink*, one could answer the question "Why did the author feel better?" with "because she had a drink." (For enablement relationships, it may be necessary to instead consider a "why" question like "Why was John able to go to the ball?")
- **The linguistic tests:** Can the sentence be rephrased as "It is because (of) X that Y" or "X causes Y?" If so, it is likely to be causal. (This test is particularly important for Creation/termination verbs.)
- **The temporal order test:** Is the Cause asserted to come before the Effect? If not, it is not causal.

- **The counterfactuality test:** Would the Effect have been just as probable to occur or not occur had the Cause not happened? If so, it is not causal.
- **The ontological asymmetry test:** Could you just as easily claim the reverse direction of causation? I.e., is it hard to tell which is the Cause and which is the Effect? If so, it is not causal.

B.2.5.1 Temporal enablement

Sometimes, temporal language is used simply to anchor one event to another, as in *Her family came to the U.S. after the war.* Often, however, it is used to describe a series of related events where the later ones do not make sense without the earlier ones. For example, consider *When I get to class, I'll give a lecture on case marking.* It would not make sense to give a lecture before getting to class, but the intent of the speaker is to describe a sequence, not to emphasize the enablement relationship. Such instances should not be annotated as causal.

In contrast, some temporal language is intended to emphasize an enablement relationship, albeit implicitly. For example, *After the guests left, I could finally go to sleep* suggests that there was an enablement relationship between guests leaving and sleeping. Cases like this **should** be annotated as causal.

There are no hard and fast guidelines for this decision, but it often depends on the tenses of the two events and whether the initial event is suggested to resolve an obstacle to the second. For instance, *After the guests left, I cleaned up and went to sleep* does not qualify as causal, despite its similarity to the previous example, because it does not suggest that the guests' presence was an obstacle to sleeping.

B.3 Corpus lexicography to define the scope of causal language

To limit the scope of the annotations to a reasonable slice of the semantic space, we need to define what types of language we are considering causal. We use a corpus lexicography approach: we maintain a catalog of connectives that have been found to indicate causal relationships. The full catalog can be found in Appendix C.

The remainder of this section details the approach used to determine whether linguistic patterns are in fact causal connectives, and what should be included in the entry for the connective in the catalog. Annotators should use these guidelines to determine when to suggest a new connective pattern.

B.3.1 Defining causal language

We use the term "causal language" to refer to clauses or phrases in which one event, state, action, or entity is explicitly presented as promoting or hindering another. We call the event/state/action/entity that promotes/hinders the other the Cause, and the thing that is promoted/hindered the Effect. Both a Cause and an Effect should be explicitly mentioned in the text (except in cases where one or the other is syntactically eliminated, as in passives and some nominalizations and infinitives). They must also be deliberately related by an explicit connective. As discussed in §B.2.1, a Means of causation—the action taken by the Cause by which it causes the action—may also be present.

B.3.2 Types of language excluded by our definition

For the purposes of this corpus, the following types of language should **not** be annotated as causal language:

- Instances where the connective lexicalizes some other component beyond the causal relationship. For example, *These measures would lead to a strong economy* should be annotated, but *These measures would boost the economy* should not, because the connective encodes not just causation but also something about the result (i.e., raising).
- Instances where it is stated that some relationship exists, but it is not stated what that relationship is. *Smoking is linked to cancer* specifies nothing about what sort of causal link is present, so it should not be annotated.
- Instances where language of causation is borrowed to express a *pragmatic cause*, as in âĂIJ*Let's discuss that later, because I know you were hoping to get to it soon*, or an evidentiary inference, as in *The car was driven recently, because the hood is hot.* (See, however, §B.4.1.5.)
- Aspectual verbs—verbs that indicate the temporal stage of an event (*begin*, *maintain*, *preserve*, *resume*, etc.). These should not be annotated even though they occasionally also suggest causing a change in the stage of the event in question.

B.3.2.1 What counts as added information?

It may not be immediately obvious what counts as lexicalizing additional information—i.e., what would disqualify a connective on these grounds. For the purposes of this annotation scheme, we consider any of the following types of added information to disqualify a connective:

- Information about the means by which the causation is accomplished. For instance, resultative constructions such as *kicked the door shut* are excluded because the connective encodes the fact that the means of causing the door to shut was kicking. Likewise, even though *hire* participates in constructions like those for *cause* (e.g., *hired John to teach me*), it lexicalizes the means of causation, so it is not annotated.
- Information about the result. For instance, *kill* means *cause to die*, but it should not be annotated as a connective, because it includes information about the result.

B.3.3 Words to include in a connective pattern

Only the words that form the fixed part of the construction should be included as part of a connective. By "fixed," we do not mean that the lexical forms of the words must be fixed, but rather that their stems must be fixed. A word whose tense or number varies between instances, for example, should still be annotated as a fixed part of the construction. Words that modify the tense, aspect, or modality of the connective, such as *should* or *to*, should not be included in the annotation.

In some cases, the fixed construction will be a single conjunction or adverb, such as *because* or *therefore*. In many cases, however, the connective will consist of several words, possibly non-consecutive. Examples include *if/then* constructions and *prevent* __ *from* __.

B.4 Special cases

Several special cases require additional care with annotation.

B.4.1 Special cases of connectives

1. **Negations:** Connectives may be negated to say that the indicated causal relationship does not hold—for example, *This will not lead to the same disastrous consequences.* These negations should be ignored for the purposes of annotation: the negation is simply another modifier.

2. **Conjunctions:** If there are two different connectives related by a conjunction, two different annotations should be created, one for each connective. The same holds if only one part of the connective is conjoined (e.g., *This caused me to trip and to fall*).

   This does not apply to arguments—if there is a single connective but one or both of the arguments consist of two phrases or clauses connected by a conjunction, the entire conjoined argument phrase should be annotated as the argument of a single causation instance.

3. **Complementizers:** Arguments may be introduced by complementizers such as that, sometimes immediately after the connective—for example, *We must ensure that this does not happen.* When they occur after verbs, these complementizers should be annotated as part of the argument, not the connective. (This is because the constructions that can be followed by complementizers can all also be followed by another noun phrase, so it is more parsimonious not to consider the complementizer as a fixed part of the construction.)

   Complementizers not following verbs may be part of the connective if they are always present (e.g., *on the grounds that*). When the complementizer is optional after a non-verbal connective (e.g., *so that*), the *that* should be omitted from both spans.

4. **Ambiguity in the number of connectives:** If a connective could in principle either be split into multiple connectives or combined into a single one, it should be annotated as multiple connectives. For example, in *This is necessary to prevent war, necessary to prevent* could be considered a single larger connective (Inhibit). Nonetheless, it should split into the connectives *necessary to* (Facilitate) and *prevent* (Inhibit).

5. **Pragmatic discourse markers:** Occasionally, causal words will be used to express a *pragmatic cause*—e.g., not *X is true because*, but *X, and I'm saying this because....* These should not be annotated. However, annotators should leave a note on the connective with the comment "Discourse" in case this decision needs to be revisited.

6. **Inference markers:** Occasionally, causal words will be used to indicate evidence—for example, *The car was driven recently, because the hood is still hot.* Like pragmatic discourse markers, these should not be annotated. However, annotators should leave a note on the connective with the comment "Inference" in case this decision needs to be revisited.

   Causal language used to talk about whether something meets a particular definition (e.g., *This is not a square, because its sides are uneven*) does not fall into this category. Such language describes the reason for something being true, not merely the reason for believing something to be true.

7. **Nominal and adjectival connectives:** Many connectives are adjectives or nouns. These can appear embedded within many different linguistic constructions. For example, they can appear as arguments of a copula (*The cause of the fire was a cigarette butt*), complements (*Their support seems essential to the organization's continuity*), appositives (*He researches E. coli, the cause of many an infection*), prepositional arguments (*He pointed to his predecessor's mistakes as the cause of the current crisis*), and more. Only the noun or adjective and the function words consistently used to introduce their arguments should be annotated as part of the connective. In the above examples, *cause* and *essential to* should be annotated as the connectives.

Note that, for nouns, the possessive *of* or *'s* should not be annotated as part of the connective; it is considered an interacting construction, not part of the nominal construction itself. To test whether an *of* is possessive, try replacing it with a *'s*.

Nominal and adjectival connectives should only be annotated when an explicit Cause or Effect is given (for Purpose connectives, the Cause specifically must be present). For example, the connective and Effect should be annotated in *Authorities have yet to assign responsibility for [the mayhem]*$_{\text{Effect}}$, and both arguments should be annotated in *She bought a small truck for the purpose of carrying her kayak*. However, neither *Her purpose was to make kayaking trips easier* nor *Local sources placed responsibility on al Qaeda* should be annotated at all.

8. **Nominalized verbs:** A nominalization of any verb in the constructicon should be annotated as a connective (e.g., *prevention of*). When the object of the verb appears with an *of* phrase, as in *prevention of*, the *of* should be annotated as part of the connective, in keeping with the practice of annotating verb argument words.

9. ***To* indicating a verb argument:** Occasionally, *to* is used in a way that can be rephrased as *in order to*, but is arguably setting off a verb argument—e.g., *I used caulk to fix the leak*. This distinction is difficult to make reliably, and has few implications for downstream processing. Therefore, such cases should be annotated in the same fashion as other *to*s of Purpose. (These cases should not be annotated if they cannot be rephrased as *in order to*.)

10. **The word *for*:** One particularly difficult case is the preposition *for*. It is included in the catalog, but below is a more complete list that includes possible meanings that are not considered causal:

Sense	Examples	Decision
Exchange of goods	*Buy **for** $5, swap X **for** Y*	Not causal
Topic	*My ideas **for** a better world*	Not causal
Purpose of existence (i.e., existence of the putative Effect)	*a vase **for** the flowers,* *a forward-facing camera **for** video chat*	Not causal
Precipitating action	*He thanked the crowd **for** listening,* *I attacked him **for** slandering me,* *I'm reporting them to the BBB **for** horrible customer relations*	Motivation
Purpose of benefit	*I'm running **for** prostate research*	Purpose
Precipitating situation or need	*I go to the mall **for** the crowds,* *I went to the store **for** a bag of carrots*	If the purpose would happen anyway: Motivation If the agent's action causes the purpose to happen: Purpose

B.4.2 Special cases of arguments

1. **Coreferent nouns/pronouns:** If a Cause or Effect argument consists **entirely** of a pronoun that is coreferent with another noun, the pronoun should still be annotated as the primary Cause or Effect. However, a coreference link should be added to the span to which the pronoun refers if the

referent is in the same sentence as the pronoun. If the referent is not in the same sentence, no coreference link should be added.

2. **Missing arguments:** The Cause or Effect may missing entirely, particularly in passive sentences. For example, no Cause is given in the sentence *The hedging of business risks could well be discouraged.* In these cases, the missing argument should simply be omitted from the annotation. (Note that this is relatively rare.)

3. **Conjunctions with shared constituents:** Coordinate structures may lead to a piece of an argument being shared between two parts of the sentence. Only the relevant subspan of the coordination should be annotated as part of the argument. For example, the bolded portion of the following would be annotated as the Cause: *The **product** was widely praised but **came with no assembly instructions**, leading to many complaints.*

4. **Attachment ambiguities:** When there is an attachment ambiguity that cannot be resolved even semantically, low attachment should be preferred. For example, consider the sentence *If you go, you'll regret it, because John will be there.* It is completely ambiguous (and irrelevant semantically) whether the *because* is modifying *you'll regret it* or *if you go, you'll regret it.* Preferring low attachment means it should be read as modifying *you'll regret it.*

5. **Coreference ambiguities:** Similarly, if there are two possible chunks that could be annotated as the antecedent of a coreferent pronoun, and the ambiguity cannot be resolved even semantically, the smaller chunk should be annotated as the antecedent.

6. **Cause spans of Purposes:** In Purpose instances, the controlling subject of the Effect clause will often be a subspan of the action whose purpose is being stated. For example, in *Maggie went to the store to buy eggs*, *Maggie* is the controlling subject of *to buy eggs*, but it is Maggie's trip to the store whose purpose is to buy eggs. The entire action—in this example, *Maggie went to the store* —should be annotated as the Cause.

7. **Arguments of non-finite verbal connectives:** Verbal connectives can appear without an explicit subject (e.g., *It is important to prevent abuse*) In some cases, however, the subject is known (e.g., *It is important for us to prevent abuse*). If the subject can be definitively established, and no action is specified, the implied subject should be annotated as the Cause (*us*, in the above example), or the subject's action if it is explicitly given (see §B.2.2.3 above). If the subject cannot be definitively established, no Cause should be annotated.

   The test for a known subject is whether there is exactly one reflexive pronoun which can be added to the end of the clause. In the above example, *ourselves* can be added to the end of the *prevent* clause, but no other reflexive pronoun can, because *us* is implicitly the subject of *prevent.*

   This case also covers *how to* or *why to* constructions (e.g., *We must determine how to prevent abuse.*).

8. **Hedges:** Arguments can contain hedges that suggest varying amounts of certainty about the argument's truth (e.g., *It probably won't work*, *I don't think it'll work*). The assumption of this project is that causal language analysis will be composed with other NLP tools, such as hedge detectors. To keep this annotation scheme simpler, then, hedges should always be included in the argument annotation, and later analysis (a "hedge trimmer," perhaps) can strip out the hedges from arguments if desired.

9. **Periphrastic causatives:** In periphrastic constructions, such as *cause the basin to overflow*, there

are sometimes two distinct components of the Effect argument: the patient (e.g., the basin) and the result (e.g., overflowing). These can become syntactically separated, as in passives, and often do not form a single clause (e.g., *prevented the basin from overflowing* or *the basin was prevented from overflowing*). For consistency, these should always be annotated as two fragments of the same argument, even if they are adjacent. For a periphrastic connective that includes a preposition such as *from*, the preposition should not be included in either fragment.

10. **Speech acts:** The Effect argument may be a speech act—for example, *Jeremy can't make it, so can you bring some wine?* The instance should be classified considering the Effect to be the speaker performing the speech act, which will usually make it a Motivation.

11. **Participials:** There may be a participial phrase that seems to be an extension of the Cause or Effect but is separated from it in the sentence. For example, in *He walked warily because of the mud, skirting the lake at a distance*, the final participial is understood to apply to *he walked warily*. Such participials should not be annotated as part of either argument.

    If the connective itself is a participial, as in *The fire swept through the county, **causing** extensive damage*, the entire clause modified by the participle (here, *the fire swept through the county*) should be annotated as the argument, rather than just the subject of that clause (*the fire*).

12. **Parentheticals, prepositional phrases, and other modifiers and interruptions:** It is not always clear whether phrases like prepositional phrases and parentheticals should be included in an argument span. In general, the rule is that any language modifying or describing the argument (but not modifying the other argument or the entire relationship) should be included. Subspans that **should** be included in arguments include (with the relevant argument span in SMALL CAPS in the examples):

    - **Prepositions that modify only the argument span.** For example, ACTIVISTS' EFFORTS SINCE 1985 *have led to few changes.*
    - **Relative clauses.** For example, THE FIRST LADY, WHO TRADITIONALLY CHOOSES THE DECORATIONS, *caused a stir with her selection.*
    - **Appositives.** For example, THE FIRST LADY, TRADITIONALLY THE DECORATOR OF RECORD, *caused a stir with her selection.*

    Subspans that should not be included in arguments include:

    - **Parenthetical matrix clauses.** For example, THE PARK, *he said,* WAS FLOODED *because of the rain*; *Due to the rain,* THE PARK, *as she expected,* WAS FLOODED; *Due to the rain,* THE PARK WAS FLOODED, *as she expected.* For consistency, the same standard applies even in cases where the parenthetical is arguably the matrix clause only for the argument which it interrupts or is juxtaposed to.
    - **Prepositions that modify the entire relationship.** For example, *Since 1985,* ACTIVISTS' EFFORTS *have led to few changes.*

13. **Effects with modality of obligation:** When an Effect has a *should* in it—e.g., *I didn't get cake, so you should give me the next piece*—we interpret the statement as indicating that, in the speaker's opinion, an obligation exists in the world as a result of the Cause. These instances should therefore be annotated as Consequence.

B.5 Suggestions for annotation process

B.5.1 Annotation order

Some approximation of the following is the suggested procedure for annotating each instance of causal language:

1. Identify a connective as causal, or not causal but containing an overlapping relation, making sure it is used in a sense consistent with the notes on the pattern in the catalog. (Most cases are unambiguous, but many do require checking the comments and WordNet senses.)
2. Identify argument spans. For each argument, check whether a pronoun constitutes the entire syntactic argument, and if so, add the appropriate coreference link.
3. Using information from both the connective and the argument spans, classify the type and degree of causation according to the decision trees in §B.2.2 and §B.2.3. While editing the connective, also mark any overlapping relations present in the instance.
4. Add the Cause/Effect/Means links from the connective to the arguments. If the instance is a Purpose, make sure the motivation is listed as the Cause.
5. It may be easier to skim through the entire document once identifying all potential connectives, and only then going on to steps 2–4 for each connective.

B.5.2 Final error-catching passes

After annotating everything, two final passes should be made. First, annotators should look for missed or incorrectly annotated instances of all sorts, including mislabeled or missing arguments and Non-Causal instances with no relation type. Then, they should use the annotation tool's built-in search functionality to look for instances of each the following easy-to-miss and frequently non-causal connectives:

- *about*
- *after*
- *as*
- *at*
- *before*
- *during*
- *for*
- *once*
- *since*
- *to*
- *until*
- *when*
- *where*
- *while*
- *with*

B.5.3 Handling passives or infinitives

Passive and infinitival connectives can be confusing: with one argument missing, it can be tempting to look for the argument elsewhere in the sentence. To annotate these connectives appropriately, try restating the sentence with an active verb and seeing what the arguments would be then.

For example, consider the sentence *The action was not made public*. It may be tempting to mark *the action* as one argument and *public* as another, or something similar. However, we can see that this is incorrect by recasting the sentence in the active voice: *The Army did not make the action public*. In this case, it is clear that *the Army* and *the action public* are the two arguments, with *make* as the connective. In the passive sentence, then, *make* should still be the connective, and *the action* and *public* will be two fragments of the Effect argument.

In passives where the cause or agent is introduced with *by*, the *by* should not be included in the connective span.

B.6 Examples

The examples shown in the table below demonstrate how to apply these criteria to specific sentences (not including overlapping relations).

Sentence (with **connective**, CAUSE, *Effect*, Means, and antecedent)	**Type**	**Degree**
Some borrowers opted for nontraditional mortgages **because** THAT WAS THEIR ONLY WAY TO GET A FOOTHOLD IN THE CALIFORNIA HOUSING MARKET.	Motivation	Facilitate
If THEY ARE REGULATED ENTITIES, *yes, we can see their code and they need to freeze their code if asked*.	Consequence	Facilitate
MY BROTHER **caused** *a fire* by `dropping a lit match`.	Consequence	Facilitate
With `that one signature`, THE PRESIDENT sparked *hundreds of protests*.	Consequence	Facilitate
They come up with a common standard **so** that THEY ARE ALL BUSTING TRADES AT THE SAME LEVEL.	Purpose	Facilitate
LOW INTEREST RATES AND WIDELY AVAILABLE CAPITAL were **prerequisites for** *the creation of a credit bubble*.	Consequence	Facilitate
For *the United States* **to** *continue to lead the world's capital markets*, WE **must** CONTINUE TO ENCOURAGE INNOVATION.	Consequence	Facilitate
A JUDGMENT IN FAVOR OF THE UNITED STATES shall **stop** *the defendant* **from** *denying the allegations of the offense in any subsequent civil proceeding brought by the United States*.	Consequence	Inhibit
What are your recommendations for creating A SYSTEM THAT would prevent or **discourage** *banks* **from** *becoming "too-big-to-fail"*?	Consequence	Inhibit
Without BETTER REGULATION, *the economy will not recover and we can expect further crises*.	Consequence	Inhibit
WINE WITHOUT FOOD **makes** *my head hurt*, and with it makes my stomach hurt.	Consequence	Facilitate
WINE without food makes my head hurt, and WITH IT **makes** *my stomach hurt*.	Consequence	Facilitate

APPENDIX C

Causal Language Constructicon

Below is the full list of constructions consulted by annotators during annotation.

Due to space constraints, this table does not include several columns that were available to annotators:

- A list of other variants of the construction. For example, ⟨*cause*⟩ *forces* ⟨*effect*⟩ was listed with the variants ⟨*cause*⟩ *forces* ⟨*effect*⟩ *to* ⟨*effect*⟩ and ⟨*cause*⟩ *forces* ⟨*effect*⟩ *into* ⟨*effect*⟩. Similarly, *the* ⟨*more cause*⟩, *the* ⟨*more effect*⟩ has the variant *the* ⟨*comparative cause*⟩, *the* ⟨*comparative effect*⟩.
- For verbs, a list of WordNet sense keys indicating which senses of the verb were deemed causal.
- The degree of causality (Facilitate or Inhibit). Degrees are intuitive and predictable from the connective.

A table containing these additional items can be found at `https://bit.ly/CausalConstructicon`.

The following abbreviations are used in the "POS type" column:

- CC = conjunction (coordinating)
- CS = conjunction (subordinating)
- Cplx = complex—i.e., not easily describable in terms of a single POS

The "connective words" column indicates which words appear consistently enough as part of the construction that they should be annotated as part of the connective. Words in brackets indicate words that are only present in some variants of the construction.

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
⟨effect⟩ is conditioned on ⟨cause⟩	Adj	[conditioned], [conditional], on	Not Purpose			*Our participation is conditional on your good behavior.*
⟨effect⟩ is contingent on ⟨cause⟩	Adj	contingent, on	Not Purpose			*Resolution of the conflict was contingent on the signing of a cease-fire agreement.*
⟨cause⟩ is critical to ⟨effect⟩	Adj	critical, to			Should only be annotated if the effect is given; *critical* on its own does not imply that there is a particular effect that the erstwhile cause is critical **for**.	*Transparency is critical to a fair judicial system.*
⟨cause⟩ is essential to ⟨effect⟩	Adj	essential to			Should only be annotated if the effect is given; *essential* on its own does not imply that there is a particular effect that the erstwhile cause is essential **for**.	*Transparency is essential to a fair judicial system.*
⟨cause⟩ is responsible for ⟨effect⟩	Adj	responsible, for	Consequence		Only in the sense of being the primary cause of something and so able to be blamed or credited for it. Should not be annotated when it is attributing an action to an actor (e.g., *My dad was responsible for breaking the vase*).	*Residential use is responsible for a surprisingly small amount of the water shortage.*
⟨cause⟩ is vital to ⟨effect⟩	Adj	vital, [to], [for]	Consequence			*Good financial accounts are vital to the success of any enterprise.*
⟨cause⟩, and consequently, ⟨effect⟩	Adv	consequently	Consequence or Motivation			*They often avoid reading and consequently fail to improve.*
⟨cause⟩; hence, ⟨effect⟩	Adv	hence	Not Purpose		Cause may be conjoined to hence + effect by any conjunction or juxtaposition. When *hence* appears at the start of a sentence, only the effect should be annotated.	*Today there will be a stiff breeze and hence a high windchill.*
Once ⟨cause⟩, ⟨effect⟩	Adv	once	Not Purpose	Temporal		*Once we started seeing results, we began to relax.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
⟨cause⟩; therefore, ⟨effect⟩	Adv	therefore			If this connective appears at the start of a sentence, no cause should be annotated: we are currently only concerned with intra-sentential arguments, and in any case it can be difficult to determine how much previous material is the cause.	*Therefore, we should consider how to most appropriately give the Fed the necessary authority.*
⟨cause⟩ is why ⟨effect⟩	Adv	why	Not Purpose			*That is why I believe in regulation properly done.*
⟨adequate cause⟩ to ⟨effect⟩	Cplx	adequate, [for], [to]		Sufficiency/ extremity	*Adequate* should be included in the cause. Should only be annotated if the effect is given; *adequate* on its own generally means just that there is some **expected** level that has been met.	*We have adequate time to listen to this.*
⟨effect⟩ as long as ⟨cause⟩	Cplx	as, long, as	Not Purpose	Temporal, Hypothetical	Only in the sense of “provided that.” Should not be annotated as causal when used to mean either “for the full duration of” (e.g., *That’s been there as long as I’ve been alive*) or “once it’s already the case that…” (e.g., *As long as you’re doing it, you might as well do it well.*).	*As long as you feed him, he’ll be cooperative.*
As a result, ⟨effect⟩	Cplx	as, result, [of]			If this connective appears at the start of a sentence, no cause should be annotated. *A* should be omitted, because other determiners (*the*, *one*) can take its place.	*As a result, vital legislation involving the budget, elections, and amendments remains trapped.*
⟨effect⟩ by virtue of ⟨cause⟩	Cplx	by, [virtue], [of]				*They hold the posts by virtue of family connections.*
⟨cause⟩ clears the way for ⟨effect⟩	Cplx	clear, the, way, for, [to]	Consequence			*This FCC vote clears the way for lower-cost mobile data.*

Pattern	**POS type**	**Connective words**	**Causation type**	**Possible overlaps**	**Comments**	**Example(s)**
⟨enough cause⟩ to ⟨effect⟩	Cplx	enough, [that], [for], [to]		Sufficiency/ extremity	*Enough* should be included in the cause. Should only be annotated if the effect is given; *enough* on its own generally means just that there is some **expected** level that has been met. In the unusual event that the cause is a full clause, rather than an adjective, the full clause should be annotated.	*We have enough time to listen to this.*
for ⟨DET purpose=effect⟩, ⟨cause⟩	Cplx	for, purpose, [of]	Purpose		Determiner should not be included, as it can vary between instances.	*For that reason, I look forward to hearing from today's witnesses. I need to know for tax purposes.*
for ⟨DET reason=cause⟩, ⟨effect⟩	Cplx	for, reason, [of]	Motivation			*For that reason, I look forward to hearing from today's witnesses.*
⟨cause⟩ if ⟨effect⟩ is to ⟨effect⟩	Cplx	if, be, to	Not Purpose	Hypothetical	The cause must be a clause whose modality includes some form of obligation (*must*, *is required*, etc.).	*We'll need a collaborative effort if we are to have a successful outcome.*
⟨effect⟩ in advance of ⟨cause⟩	Cplx	in, advance, of	Not Purpose	Temporal	Causal when used to suggest "in preparation for."	*What these reflections mean for Catholics worldwide is under discussion in advance of the upcoming synod of bishops.*
⟨effect⟩ in case ⟨cause⟩	Cplx	in, case, [of]	Motivation		Usually describes one action being taken out of a perception for the possibility of another.	*He placed a suitcase in the corner in case of an emergency trip to the hospital.*
⟨effect⟩ in an effort to ⟨cause⟩	Cplx	in, effort, to	Purpose			*I climbed the ladder in an effort to see into the attic.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
for ⟨effect⟩ to ⟨effect⟩, ⟨cause⟩	Cplx	[in], [order], [for], [to]	Consequence		The cause must be a clause whose modality includes some form of obligation (e.g., *must*). Note that this may be abbreviated to just *to*. Even in this case, it is Consequence, not Purpose. This construction includes all *need to* / *need for* constructions, *necessary* constructions, and many *require* constructions, as well (e.g., *Greece will need debt relief to get back on its feet*; *more data is required to make a decision*). When *in order to* is used with modality of obligation to mean enablement, as in *We must get lots of input in order to make a good product*, it falls into this category, rather than Purpose.	*For market discipline to be effective, market participants must not expect that lending from the Fed is readily available.* *We'll need a lot of input to create the best product.* *To get good ice cream, you have to go to Mercurio's.*
⟨effect⟩ in response to ⟨cause⟩	Cplx	in, response, [to]	Motivation		Only in the sense of "as a reaction to," not in the sense of answering verbally or in writing.	*In response to heavy criticism, the CDC rescinded its recommendations.*
in the aftermath of ⟨cause⟩, ⟨effect⟩	Cplx	in, the, aftermath, of	Not Purpose	Temporal		*In the aftermath of the vote, the sense that the Union is fracturing cannot be ignored.*
In the face of ⟨cause⟩, ⟨effect⟩	Cplx	in, the, face, of	Not Purpose			*The debate has sharpened in the face of concerns about the safety of formaldehyde-treated flooring.*
⟨effect⟩ in the wake of ⟨cause⟩	Cplx	in, the, wake, of		Temporal		*In the wake of the storm, there were many broken tree limbs.*
in view of ⟨cause⟩, ⟨effect⟩	Cplx	in, view, of	Motivation			*In view of this new evidence, we would like to reconsider our decision.*

Pattern	**POS type**	**Connective words**	**Causation type**	**Possible overlaps**	**Comments**	**Example(s)**
⟨insufficient cause⟩ to ⟨effect⟩	Cplx	insufficient[ly], [for], [to]		Sufficiency/ extremity	*Insufficient[ly]* should be included in the cause. Should only be annotated if the effect is given; *insufficient* on its own generally means just that there is some **expected** level that has been met.	*It must contain sufficient criteria to prevent cost to the taxpayer to the greatest extent possible.*
⟨effect⟩ on condition of ⟨cause⟩	Cplx	on, condition, of	Consequence or Motivation		Usually seems to be *on condition of anonymity.*	*Only on condition of a radical widening of definitions will it be possible to consider art the only social power.*
⟨effect⟩ on grounds of ⟨cause⟩	Cplx	on, grounds, [of], [that]	Motivation			*Participants were chosen on the grounds of their past work.*
⟨cause⟩ opens the door for ⟨effect⟩	Cplx	open, door, [for], [to]	Consequence			*This opens doors to opportunity.*
⟨cause⟩ opens the way for ⟨effect⟩	Cplx	open, the, way, [for], [to]	Consequence			*This decision opens the way for much broader application of the law.*
⟨cause⟩ paves the way for ⟨effect⟩	Cplx	pave, the, way, for, [to]	Not Purpose			*Her success paves the way for the county's next wave of female wrestlers.*
⟨effect⟩ so as to ⟨cause⟩	Cplx	so, as, to	Purpose			*He nudged a stick aside so as to let the flames breathe.*
⟨so cause⟩ that ⟨effect⟩	Cplx	so, [much], [many]	Not Purpose	Sufficiency/ extremity	*So* and *much* should be included in the cause. In the unusual event that the cause is a full clause, rather than an adjective, the full clause should be annotated, including the part before the *so*.	*She got so lonely that she decided to watch TV.*
⟨sufficient cause⟩ to ⟨effect⟩	Cplx	sufficient[ly], [that], [for], [to]		Sufficiency/ extremity	*Sufficient[ly]* should be included in the cause. Should only be annotated if the effect is given; *sufficient* on its own generally means just that there is some **expected** level that has been met.	*It must contain sufficient criteria to prevent cost to the taxpayer to the greatest extent possible.*

Pattern	**POS type**	**Connective words**	**Causation type**	**Possible overlaps**	**Comments**	**Example(s)**
the ⟨comparative cause⟩ the ⟨comparative effect⟩	Cplx	the, the	Not Purpose	Correlation	Annotate even though we don't capture the need for a requirement.	*The more the wave rose, the faster we ran.*
⟨too cause⟩ to ⟨effect⟩	Cplx	too, [for], to		Sufficiency/ extremity	*Too* should be included in the cause. Should only be annotated if the effect is given; *too* __ on its own generally means just that there is some **expected** level that has been surpassed.	*Virtually every primary dealer is considered too big or too interconnected to fail.*
⟨effect⟩ with DET goal of ⟨cause⟩	Cplx	with, goal, of	Purpose		Note that the cause and effect may seem to be backwards. The motivating argument is always the cause. The determiner should not be included in the annotation.	*The speaker of Parliament read a roll call of the 275 elected members with a goal of shaming the no-shows.*
⟨effect⟩ with DET objective of ⟨cause⟩	Cplx	with, objective, of	Purpose		Note that the cause and effect may seem to be backwards. The motivating argument is always the cause. The determiner should not be included in the annotation.	*The speaker of Parliament read a roll call of the 275 elected members with the objective of shaming the no-shows.*
⟨cause⟩, so ⟨effect⟩	CC	so			Should not be annotated when used as a discourse marker to set off a new statement (e.g., *So where are we going?*).	*We will probably have some votes, so we will maximize our time.*
⟨cause⟩, and thus ⟨effect⟩	CC	thus				*This detergent is highly concentrated and thus you will need to dilute it.*
⟨effect⟩, as ⟨cause⟩	CS	as		Correlation, Temporal	Should not be annotated at all when used to mean "like."	*As science has yet to prove or disprove the existence of a divine power, and probably never will, I will use my gift of reason.* *As the power increased, the rotor spun faster.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
⟨effect⟩ because ⟨cause⟩	CS	because	Not Purpose		Should not be annotated when used in a discourse sense. A good test for this is whether the *because* can be rephrased as *I'm saying this because*—e.g., *What time is it? Because I'm hungry*.	*Nearly every session since November has been adjourned because as few as 65 members made it to work.*
⟨cause⟩ else ⟨effect⟩	CS	else			Unusual	*We must govern effectively, else chaos will reign.*
⟨effect⟩, for ⟨cause⟩	CS	for			Only in the sense of "because."	*That would be of tremendous help, for it is true that speculators bear blame.*
given ⟨cause⟩, ⟨effect⟩	CS	given			*Given* is a reasoning word, so we do not mark it as Context.	*Given that you can't make it, we'll have to postpone.*
If ⟨cause⟩, ⟨effect⟩	CS	if, [then]		Hypothetical	*If/then* constructions should only be marked as causal if the intention is that the "if" leads directly to the "then." They should not be marked when used to assert only what would be true in a hypothetical world (e.g., *if I had been asked, I would have said no*). See the annotation guide for some recommendations for making this judgment. *Even if* generally falls into the hypothetical category and should not be annotated.	*If I don't get an A on the final, I can't pass the course.*
in an attempt to ⟨cause⟩, ⟨effect⟩	CS	in, attempt, to	Purpose			*The fox leapt in an attempt to catch the mouse.*
⟨effect⟩ to ⟨cause⟩	CS	[in], [order], to	Purpose		Should only be marked when it can be paraphrased as "in the hopes of" or "with the goal of." Note that the cause and effect may seem to be backwards. The motivating argument is always the cause.	*What should be done to provide a better statutory framework?*
⟨effect⟩ lest ⟨cause⟩	CS	lest	Purpose		The fear of the cause motivates the effect.	*He kept his headphones on lest he disturb anyone.*

Pattern	**POS type**	**Connective words**	**Causation type**	**Possible overlaps**	**Comments**	**Example(s)**
now that ⟨cause⟩, ⟨effect⟩	CS	now, that	Motivation or Consequence			*Now that she is rich and famous, she is constantly being besieged by appeals for aid.*
should ⟨cause⟩, ⟨effect⟩	CS	should		Hypothetical	Only if the sense of *if ⟨cause⟩ should ⟨cause⟩, ⟨effect⟩*.	*Should you wish to cancel your order, please contact our customer service department.*
⟨effect⟩, since ⟨cause⟩	CS	since		Temporal		*Religion should not be substituted for science, nor the reverse, since they deal with different domains.*
⟨effect⟩ so ⟨cause⟩	CS	so	Purpose		Note that the cause and effect may seem to be backwards. The motivating argument is always the cause.	*Masking tape is used to cover up some beams so that only one is triggered.*
⟨effect⟩ thanks to ⟨cause⟩	CS	thanks, to				*Thanks to their fast action, we dodged a bullet.*
⟨cause⟩, and then ⟨effect⟩	CS	then		Temporal	Generally only causal when the effect has a modality of possibility. Another indicator of causality is *only* (i.e., *only then…*).	*I'll buy some peanut butter, and only then can I make a sandwich.*
⟨effect⟩ unless ⟨cause⟩	CS	unless			*Then* form is unusual.	*We have not replaced the discipline of a lender not lending to a borrower unless the lender is sure that the borrower can repay.*
⟨cause⟩ where ⟨effect⟩	CS	where		Context	Should not be annotated when it refers to physical location. It should only be annotated when it is a metaphorical *where* referring to circumstance—i.e., where it could just as easily have been *when.* Don't annotate when it can be replaced with *in which* (e.g., *a meeting where 10 people showed up*).	*Where regulators cozy up with CEOs, corruption abounds.*
the aftermath of ⟨cause⟩ is ⟨effect⟩	Noun	aftermath	Not Purpose	Temporal	Usually appears with an argument missing. Should not be annotated for *in the aftermath of,* which is its own collocation.	*Residents are still dealing with the aftermath of the storm.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
⟨cause⟩ is DET cause of ⟨effect⟩	Noun	cause				*Confusion regarding the R.O.E. was the proximate cause of the death of at least four unarmed individuals.*
⟨cause⟩ (is) condition of ⟨effect⟩	Noun	condition, [of], [for]				
DET consequence of ⟨cause⟩ is ⟨effect⟩	Noun	consequence	Consequence			
DET effect of ⟨cause⟩ is ⟨effect⟩	Noun	effect				*One effect of the Thirty Years' War was the severe depopulation of Europe.*
⟨cause⟩ is grounds for ⟨effect⟩	Noun	grounds, for				*This behavior is grounds for divorce!*
the implications of ⟨cause⟩ are ⟨effect⟩	Noun	implication	Consequence		Only in the sense of "consequences." As with other nouns, possessive language should not be included in the connective. Frequently appears without an effect.	*Lilly had not fully understood the implications of his first complaint.*
⟨cause⟩ is the key to ⟨effect⟩	Noun	key, to			Only when interchangeable with "essential for."	*Happiness is the key to success.*
⟨cause⟩ is DET necessary condition of ⟨effect⟩	Noun	necessary, condition, [of], [for]			The *of* is included in the connective because this usage cannot be replaced with a âĂIJâĂŹs,âĂİ indicating that it is not a conventional possessive.	*The regulatory restraints that many experts regard as a necessary condition of technological progress are largely unnecessary.*
DET reason for ⟨effect⟩ is ⟨cause⟩	Noun	reason, for	Motivation			*My reason for leaving now is that I want to get home on time.*
DET reason [that] ⟨effect⟩ is ⟨cause⟩	Noun	reason, [that], [be because]			The *be* should only be annotated in the *is because* version.	*One of the reasons that JPMorgan Chase did better is they did not have embedded in that institution a trading house.*
⟨cause⟩ is reason to ⟨effect⟩	Noun	reason, to	Motivation		Often appears missing cause argument (e.g., *I have reason to complain*).	*I have reason to think you're a liar.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
⟨cause⟩ (is) reason why ⟨effect⟩	Noun	reason, why	Consequence or Motivation			*This is the reason why I am opposed to the bill.*
⟨effect⟩ is DET result of ⟨cause⟩	Noun	result	Consequence			*Success is the result of perfection, hard work, learning from failure, loyalty, and persistence.* *The result: a neat kitchen hack!*
after ⟨cause⟩, ⟨effect⟩	Prep	after		Temporal	Don't annotate *even after*, which is its own construction.	*After I had a drink, I felt much better.*
⟨effect⟩ amid ⟨cause⟩	Prep	amid		Context	In the more abstract sense of "in the context of," not "surrounded by" in a physical sense.	*Banks and stores closed yesterday amid growing fears of violence.*
anytime ⟨cause⟩, ⟨effect⟩	Prep	[any], [time], [anytime]	Consequence or Motivation	Correlation		
⟨effect⟩ at ⟨cause⟩	Prep	at	Consequence or Motivation	Temporal		*I jumped at the sound of the bell.*
⟨effect⟩ because of ⟨cause⟩	Prep	because, of				*The Missouri lawsuit seeks to end what it calls a state voting ban for people under full guardianship because of mental illness.*
before ⟨effect⟩, ⟨cause⟩	Prep	before		Temporal	Usually only causal in the sense of "as a precondition for."	*Before he can come to the US, he'll have to get a visa.*
before ⟨cause⟩, ⟨effect⟩	Prep	before		Temporal	In usages where later event/state represents the end of the earlier event/state (i.e., where *before* is used like *until*).	*Before I came to the US, I never bothered to smile for photos.*
⟨effect⟩ by ⟨cause⟩	Prep	by	Consequence or Motivation		Rarely causal. Only annotate in the sense of "by virtue of." Another way to think of it: only annotate when in answer to a "why" question, not when in answer to a "how" question (e.g., *By complaining, he forfeited his chances of promotion*).	*We won the game by forfeit.*
⟨effect⟩ by reason of ⟨cause⟩	Prep	by, reason, of			Often used with adjectival effects.	*"Not master of one's own mind" is different from "not guilty by reason of insanity."*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
during ⟨cause⟩, ⟨effect⟩	Prep	during		Temporal		*During the thunderstorm, we all hunkered down inside.*
⟨effect⟩ every time ⟨cause⟩	Prep	every, time		Correlation		*Every time I see that image, I cringe.*
following ⟨cause⟩, ⟨effect⟩	Prep	following		Temporal		*Following the storm, utility companies swept through looking for precarious tree limbs.*
⟨effect⟩ for ⟨cause⟩	Prep	for	Motivation, Purpose, or Consequence		Only in the senses that indicate either precipitating action/event/state (=Motivation or Consequence) or purpose of benefit (=Purpose). Should **not** be annotated in the senses that indicate exchange of goods (*buy it for $5*), topic (*my ideas for a better world*), or purpose of existence/presence (*a vase for flowers*).	*He thanked the crowd for listening.* *You're a good man for helping out.* *I'm running for prostate research.*
⟨effect⟩ for the sake of ⟨cause⟩	Prep	for, the, sake, of	Purpose		Note that the cause and effect may seem to be backwards. The motivating argument is always the cause.	*It is discouraging to see citizens cooperating with the government for the sake of public safety denigrated as rats and tattletales.*
⟨effect⟩ from ⟨cause⟩	Prep	from			Only in the sense that indicates a precipitating event or state.	*He died from a blocked artery.*
⟨effect⟩ from ⟨cause⟩	Prep	from			In the sense of "since" (e.g., *from the day he arrived, …*). Rarely causal.	*From the time of my third-grade class trip to the Brooklyn Botanic Garden, I understood flowers were my ticket to a world of beauty.*
given ⟨cause⟩, ⟨effect⟩	Prep	given				*Given the importance of this, we are going to hold pretty firmly to the 5-minute rule.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
⟨effect⟩ in ⟨cause⟩	Prep	in	Motivation		Causal in uses like *in the hopes of*, *in honor of*, or *in deference to*, in which it means "motivated by." This is true even though these are arguably multi-word expressions; they still retain the original situational meaning of *in*. For multi-word expressions like *in an attempt to* where the object of *in* is the same thing as the action preceding the *in*, rather than a motivation for it, the full phrase should be annotated as a connective, rather than just *in*.	*I left early in the hopes of beating the traffic.*
in light of ⟨cause⟩, ⟨effect⟩	Prep	in, light, of	Motivation, Consequence			*Formal charges were not warranted "in light of his honest belief of the correctness of the mission R.O.E."*
⟨cause⟩ into ⟨effect⟩	Prep	into	Motivation, Consequence		Only when it indicates a result of an action.	*He tricked me into coming with.*
⟨effect⟩ of ⟨cause⟩	Prep	of			Only in the sense that indicates a precipitating event or state.	*He died of a blocked artery.*
⟨effect⟩ out of ⟨cause⟩	Prep	out, of	Motivation			*He fled out of fear for his life.*
over ⟨time-unit of cause⟩, ⟨effect⟩	Prep	over	Not Purpose	Temporal	Very rarely causal.	*Over 3000 years of domestication, silkworms have lost their flight skills.*
⟨cause⟩ to ⟨effect⟩	Prep	to	Consequence		This use of *to* can be rephrased as *which is needed to* (i.e., it introduces an effect that a nominal cause enables).	*We have the votes to pass it.*
⟨effect⟩ until ⟨cause⟩	Prep	until		Temporal	Causal almost exclusively when it appears with a negated effect, and used to indicate that the cause is a precondition for the effect not to happen.	*We can't let you go until we see improvement.*
upon ⟨cause⟩, ⟨effect⟩	Prep	upon	Consequence or Motivation	Temporal	Only in the sense of "immediately following."	*Upon hearing the news, she burst into tears.*

Pattern	**POS type**	**Connective words**	**Causation type**	**Possible overlaps**	**Comments**	**Example(s)**
when ⟨cause⟩, ⟨effect⟩	Prep	when		Temporal, Context	Context when it could be replaced with *where*. Temporal otherwise.	*When they told me the price, I nearly fainted.*
whenever ⟨cause⟩, ⟨effect⟩	Prep	whenever		Correlation		*Whenever I smile at my son, he always smiles back.*
while ⟨cause⟩, ⟨effect⟩	Prep	while		Temporal	Only in the sense of "during."	*While the family was on vacation, the burgler was able to come and go as he pleased.* *While recovering, his batting average was significantly lower.*
with ⟨cause⟩, ⟨effect⟩	Prep	with	Motivation, Consequence	Context	Should not be annotated when used in a possessive or accompaniment sense (e.g., *With thousands of unique species, the island is an evolutionary biologist's dream.*).	*With supplies running low, we didn't even make a fire that night.*
⟨effect⟩ with ⟨cause⟩	Prep	with	Motivation, Consequence		Only in the sense of "from" or "out of" (e.g., *trembling with fear*, *yellow with age*), not in the sense of "about" (e.g., *frustrated with your stubbornness*).	*The paper was yellow with age.*
within ⟨time-unit⟩ of ⟨cause⟩, ⟨effect⟩	Prep	within, [of], [after]	Consequence or Motivation	Temporal		*Within two minutes of drinking the cocktail, Tom was dead.*
without ⟨cause⟩, ⟨effect⟩	Prep	without	Not Purpose	Context	With the meaning "in the absence of."	*Without better regulation, the same problem will recur.*
⟨cause⟩ allows ⟨effect⟩	Verb	allow, [to]		Obligation/ permission		*Plastics have allowed an enormous range of new inventions.*
⟨effect⟩ arises from ⟨cause⟩	Verb	arise, [from]	Consquence	Creation/ termination		*A slight unpleasantness arose from this discussion.*
⟨cause⟩ assures ⟨effect⟩	Verb	assure	Consequence		*That*, if present, should not be annotated. Only annotated in the sense of "ensure," not "give assurance to someone."	*She wants to assure a favorable vote.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
NP attributes ⟨effect⟩ to ⟨cause⟩	Verb	attribute, to	Consequence or Motivation		Only when it can be paraphrased as __ *believes ⟨effect⟩ was caused by ⟨cause⟩*. Should not be annotated when used to mean "credits as a source" (e.g., *attributed the quote to Shakespeare*) or "consider to be a characteristic" (e.g., *I attributed great strength to him*). Wordnet does not capture this distinction. Often appears in the passive (⟨effect⟩ is attributed to ⟨cause⟩).	*She attributed his bad temper to ill health.*
⟨cause⟩ averts ⟨effect⟩	Verb	avert				*Cuts could avert the worst consequences of climate change.*
⟨cause⟩ avoids ⟨effect⟩	Verb	avoid	Consequence			*They wished to avoid a cascading effect.*
⟨effect⟩, barring ⟨cause⟩	Verb	bar	Consequence or Motivation		Only in the sense of "unless," not in the sense of banning.	*Barring clouds, it will be viewed by millions of people.*
NP blames ⟨cause⟩ for ⟨effect⟩	Verb	blame, [for], [on]	Consequence		Only in the sense of being the primary cause of something and so able to be blamed or credited for it. Should not be annotated when used to mean "associate guilt with"—e.g., *Can you blame her for leaving him?* May be missing the effect argument if it is clear from context.	*We blamed the accident on her.*
⟨cause⟩ blocks ⟨effect⟩	Verb	block			Only when it can be used synonymously with *prevent*.	*His formal reprimand effectively blocks any chance for promotion.*
⟨cause⟩ brings on ⟨effect⟩	Verb	bring, on	Consequence	Creation/ termination		*Eating ice cream too fast can bring on a headache.*
⟨cause⟩ brings ⟨effect⟩ to ⟨effect⟩	Verb	bring, to	Motivation		Only in the sense of inducing/persuading.	*The confession of one of the accused brought the others to admit to the crime as well.*
⟨cause⟩ causes ⟨effect⟩	Verb	cause, [to]				*Something that simple causes problems in subprime.*

Pattern	**POS type**	**Connective words**	**Causation type**	**Possible overlaps**	**Comments**	**Example(s)**
⟨effect⟩ comes after ⟨cause⟩	Verb	come, after	Not Purpose	Temporal	May have a time unit before the *after*, which we do not annotate.	*The investigation comes after government watchdogs raised red flags about classified material possibly being shared.*
⟨effect⟩ comes from ⟨cause⟩	Verb	come, from			Only in the sense of being a product or a result. Should not be annotated as causal when used to indicate a place or object of origin.	*These threats come from unconstrained risk-taking.*
⟨cause⟩ compels ⟨effect⟩ to ⟨effect⟩	Verb	compel, [to]	Motivation	Obligation/ permission		*The news compelled him to act.*
⟨cause⟩ contributes to ⟨effect⟩	Verb	contribute, to				*The colonel's "miscommunication" of the rules contributed to the deaths of four unarmed Iraqis.*
⟨cause⟩ creates ⟨effect⟩	Verb	create	Consequence	Creation/ termination	Only causal when used in the sense of inducing, not in the sense of manufacturing or yielding something (*John created/produced a snowman out of cotton balls*). When annotating creation/termination "create"s, a general guideline is that bringing things into existence should be annotated but manufacturing/churning out a thing should not. Difficult or ambiguous cases should be discussed.	*The immediate crisis created by the run on Bear Stearns has passed.*
⟨cause⟩ produces ⟨effect⟩	Verb	create	Consequence	Creation/ termination	Only causal when used in the sense of causing, not in the sense of manufacturing or yielding something (*John created/produced a snowman out of cotton balls*).	*No conventional drugs had produced any significant change.*
⟨effect⟩ depends on ⟨cause⟩	Verb	depend, on	Not Purpose		Only in the sense of "hinges on as a necessity," not "relies on" or "hinges on the question of."	*His success here depends upon effort and ability.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
⟨cause⟩ deters ⟨effect⟩	Verb	deter	Consequence or Motivation			*Only a health problem would deter him from seeking re-election.*
⟨cause⟩ discourages ⟨effect⟩	Verb	discourage	Consequence or Motivation		Should not be annotated in the sense of "dissuade by argumentation" (e.g., *I discouraged him from applying.*).	*This feeder discourages squirrels from stealing seeds.*
⟨cause⟩ drives ⟨effect⟩	Verb	drive, [to]	Motivation			*His behavior drove me to drink heavily.*
⟨cause⟩ eases ⟨effect⟩	Verb	ease	Consequence		Only in the sense of "makes easier."	*Tokyo's dominance of government eased efficient contact-making.*
⟨cause⟩ eliminates ⟨effect⟩	Verb	eliminate	Consequence	Creation/ termination	Only causal when forward-looking (eliminating some future eventuality).	*We must eliminate future bank runs.*
⟨cause⟩ enables ⟨effect⟩	Verb	enable, [to]				*This will in turn enable our economy to continue to grow.*
⟨cause⟩ encourages ⟨effect⟩	Verb	encourage	Consequence or Motivation		Only in the sense of promoting or fostering. Should not be annotated when used to mean "to stimulate someone by approval or urging."	*His financial success encouraged him to look for a wife.*
⟨cause⟩ engenders ⟨effect⟩	Verb	engender		Creation/ termination		*Regulatory competation can engender a race to the bottom.*
⟨cause⟩ ensures ⟨effect⟩	Verb	[ensure], [make], [sure]	Consequence		*That*, if present, should not be annotated.	*We must ensure that taxpayers are not left holding the bag.*
⟨cause⟩ guarantees ⟨effect⟩	Verb	[ensure], [make], [sure]	Consequence		*That*, if present, should not be annotated. Should not be annotated when used to mean "gives surety" or "assumes responsibility."	*Preparation will guarantee success!*
⟨cause⟩ facilitates ⟨effect⟩	Verb	facilitate				*The FDIC may set up a bridge bank to facilitate an orderly liquidation of an insolvent firm.*
⟨cause⟩ feeds ⟨effect⟩	Verb	feed			Only in the sense of fanning the flames of something.	*Racial profiling has fed a deep sense of persecution.*
⟨cause⟩ foils ⟨effect⟩	Verb	foils	Consequence			*A brave policewoman foiled the armed robbery.*

Pattern	**POS type**	**Connective words**	**Causation type**	**Possible overlaps**	**Comments**	**Example(s)**
⟨effect⟩ follows ⟨cause⟩	Verb	follow	Not Purpose	Temporal	If used intransitively, no cause should be annotated.	*The plunge in stock prices followed a disappointing earnings report.*
⟨cause⟩ forbids ⟨effect⟩	Verb	forbid, [to]	Not Purpose	Obligation/ permission	Very rarely causal. Usually just obligation/permission.	*Her sense of modesty forbids her to speak up.*
⟨cause⟩ forces ⟨effect⟩	Verb	force, [to], [into]		Obligation/ permission	Often appears in the passive. (*Force into* should be included here, not as an instance of the *into* construction.)	*Senator Brownback forces there to be a contradiction between faith and reason.*
⟨cause⟩ fosters ⟨effect⟩	Verb	foster	Consequence			*These texts foster an interest in learning new information.*
⟨cause⟩ generates ⟨effect⟩	Verb	generate		Creation/ termination	Only causal in the sense of causing, not in the sense of manufacturing or yielding something (*The new manager generated a lot of problems*, not *The computer generated this image*). WordNet does not appear to capture this distinction.	*These harsh enforcement provisions will undoubtedly generate abuses and mistreatment.*
⟨cause⟩ gets ⟨effect⟩ to ⟨effect⟩	Verb	get, to	Consequence			*His wife got him to eat more vegetables.*
⟨cause⟩ gives ⟨effect⟩	Verb	give	Consequence or Motivation		Most uses, even with the specified WordNet sense, are not causal.	*The presence of my father gave some peace of mind.*
⟨cause⟩ gives rise to ⟨effect⟩	Verb	give, rise, to	Consequence	Creation/ termination		*Their decisions gave rise to subsequent arguments.*
⟨Had cause⟩, ⟨effect⟩	Verb	had	Not Purpose	Hypothetical	Only in the sense of "if X had done Y."	*Had I known, I'd have come with you.*
⟨cause⟩ hampers ⟨effect⟩	Verb	hamper, [from]	Consequence			*Their work is hampered by lack of funds.*
⟨cause⟩ has ⟨effect⟩	Verb	have	Consequence		Only in the sense of "makes." Should be distinguished from the sometimes similar sense of "undergoes" (e.g., *I had a random guy compliment my shirt yesterday!*).	*My wife had me go to the doctor.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
having ⟨cause⟩, ⟨effect⟩	Verb	having	Consequence or Motivation	Context		*Having looked at the older document, the faculty believed that a middle ground might be appropriate.*
⟨cause⟩ helps ⟨effect⟩ ⟨effect⟩	Verb	help, [to]			Only in the sense of "makes easier," not "assists." These can usually be distinguished by whether there are clear agents helping and being helped, in which case it is assistance, or whether at least one of them is an event or state of affairs, in which case it is making easier. Should be annotated only when there is a specific effect that is being helped to happen (though that effect may be unstated and implicit). Should be annotated even when it immediately precedes another connective (e.g., *This helps ensure that. . .*).	*Your steady leadership is helping us weather the storm.*
⟨cause⟩ hinders ⟨effect⟩	Verb	hinder	Consequence			*The storm hindered our progress.*
⟨cause⟩ impedes ⟨effect⟩	Verb	impede	Consequence or Motivation			*The sap causes swelling that can impede breathing.*
⟨cause⟩ incites ⟨effect⟩	Verb	incite	Motivation	Creation/ termination		*The news incited widespread fear and paranoia.*
⟨cause⟩ induces ⟨effect⟩ to ⟨effect⟩	Verb	induce, [to]	Motivation			*We must not induce a migration of risk-taking activities to less-regulated or offshore institutions.*
⟨cause⟩ inhibits ⟨effect⟩	Verb	inhibit			WordNet senses don't capture this word well. Any sense meaning "hinder, restrain, or prevent" should be annotated.	*Our regulatory structure severly inhibits our competitiveness.*
⟨cause⟩ inspires ⟨effect⟩	Verb	inspire	Motivation			*Her story inspired me to call my own relatives.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
⟨effect⟩ takes ⟨cause⟩	Verb	[it], take, [for], [to]		Temporal		*It will take ten men to construct this building.*
⟨cause⟩ keeps ⟨effect⟩ from ⟨effect⟩	Verb	keep, from	Consequence or Motivation			*The cold kept me from going outside.*
⟨cause⟩ launches ⟨effect⟩	Verb	launch		Creation/ termination	In the sense of setting an event in motion.	*The scandal launched a massive rebellion.*
⟨cause⟩ leads ⟨effect⟩ to ⟨effect⟩	Verb	lead, to	Motivation		The two fragments of the effect should be annotated as separate fragments even if they appear consecutively.	*Recent events have led the Basel Committee on Banking Supervision to consider higher capital charges for such items.*
⟨cause⟩ leads to ⟨effect⟩	Verb	lead, to	Consequence (can be Motivation if the effect is clearly agentive)			*Sectarian divisions have often led to deadlock.*
⟨cause⟩ lets ⟨effect⟩ ⟨effect⟩	Verb	let		Obligation/ permission	Should not be annotated when used as a formality or mark of respect (*Let me just say. . .*) or an invitation (*let's get out of here*).	*I let the creature climb up my shirt.*
⟨cause⟩ makes ⟨effect⟩ ⟨effect⟩	Verb	make		Obligation/ permission	Should only be annotated if *make* is not part of a more specific causal construction. If there are two fragments of the effect, they should be annotated as separate fragments even if they appear consecutively. The *for* or *to* should also be included in the argument. Should not be annotated when used as in *make X out of Y*.	*This bill seeks to make regulation more efficient.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
⟨cause⟩ makes certain ⟨effect⟩	Verb	[make], [certain]	Consequence or Motivation		Only in the sense of “ensures,” not “ascertains.” *That*, if present, should not be annotated.	*We must make certain that nobody enters.*
⟨cause⟩ makes for ⟨effect⟩	Verb	make, for	Not Purpose		Not in the sense of “constitutes.”	*Their background made for a rocky relationship.*
⟨cause⟩ means ⟨effect⟩	Verb	mean	Consequence or Inference			*Having more chefs means that food gets out to customers faster.*
⟨cause⟩ necessitates ⟨effect⟩	Verb	necessitate				*An appreciation of biological evolution does not necessitate a completely materialistic and deterministic worldview.*
⟨cause⟩ obliges ⟨effect⟩ to ⟨effect⟩	Verb	oblige	Not Purpose	Obligation/ permission		*Her financial problems obliged her to get another job.*
⟨cause⟩ permits ⟨effect⟩	Verb	permit		Obligation/ permission		*The unusually calm winds permitted a much more gentle sailing style.*
⟨cause⟩ precipitates ⟨effect⟩	Verb	precipitate		Creation/ termination		*Every shudder had precipitated a dusting of granules onto the floor.*
⟨effect⟩ is predicated on ⟨cause⟩	Verb	predicate, on	Consequence		Only occurs in its causal meaning in the passive. Should not be annotated when used to mean “assumes as a premise” or “is a function of.”	*Peace is predicated on compromise, not firepower or belligerence.*
⟨cause⟩ prevents ⟨effect⟩	Verb	prevent, [from]		Obligation/ permission		*A market sensitive regulatory authority not only prevents some of the problems, but is pro-market.*
⟨cause⟩ prohibits ⟨effect⟩	Verb	prohibit, [from]	Not Purpose	Obligation/ permission	Very rarely causal. Usually just obligation/permission.	*The cost of safety glass often prohibits its use in private buildings.*
⟨cause⟩ promotes ⟨effect⟩	Verb	promote				*This bill promotes consolidation and cooperation among regulatory agencies.*
⟨cause⟩ prompts ⟨effect⟩	Verb	prompt, [to]	Motivation		Often appears in passive.	*His death has prompted an industry-wide investigation of safety violations.*

Pattern	**POS type**	**Connective words**	**Causation type**	**Possible overlaps**	**Comments**	**Example(s)**
⟨cause⟩ provokes ⟨effect⟩	Verb	provoke, [to]	Motivation			*Their belligerence provoked a war.*
⟨cause⟩ requires ⟨effect⟩	Verb	require, [to]	Consequence	Obligation/ permission	In the sense of "mandates." Note that in this use of *require*, it is not that the object enables the subject, but rather that the subject mandates the object.	*Politeness usually requires that the speaker shall mention the addressed person first, and himself last.*
⟨effect⟩ requires ⟨cause⟩	Verb	require, [to]	Consequence		When used to indicate that the thing required would enable the effect. Arguments are reversed from the usage meaning "mandates" (e.g., *this routine requires students to listen*).	*Combating the growing culture of fear will require a greater emphasis on the physical protection of witnesses.*
⟨cause⟩ restrains ⟨effect⟩ from ⟨effect⟩	Verb	restrain, from	Consequence			
⟨effect⟩ results from ⟨cause⟩	Verb	result, [from]				*Her victory resulted from absentee ballots cast by the nursing home residents.*
⟨cause⟩ results in ⟨effect⟩	Verb	result, in				*This move resulted in disaster.*
⟨cause⟩ sets off ⟨effect⟩	Verb	set, off	Consequence or Motivation	Creation/ termination	Should be carefully examined to make sure it's not one of the many other senses of "set off."	*The attack set off great unrest among the people.*
⟨cause⟩ sparks ⟨effect⟩	Verb	spark	Consequence or Motivation	Creation/ termination		*Unrest in the Middle East sparked a global oil shock.*
⟨cause⟩ spurs ⟨effect⟩	Verb	spur				*That should not obscure the real good that can be done in spurring a wider global fight against H.I.V.*
⟨effect⟩ stems from ⟨cause⟩	Verb	stem, from	Consequence	Creation/ termination		*Many of the universities' problems stem from rapid expansion.*
⟨cause⟩ thwarts ⟨effect⟩	Verb	thwart			Only when the object is an action.	*Private citizens have helped to thwart many of these attacks.*
⟨cause⟩ triggers ⟨effect⟩	Verb	trigger	Consequence or Motivation	Creation/ termination	Only in the sense of initiating or precipitating.	*Their small protest triggered a mass demonstration.*

Pattern	POS type	Connective words	Causation type	Possible overlaps	Comments	Example(s)
⟨cause⟩ wards off ⟨effect⟩	Verb	ward, off	Not Purpose		Only in the metaphorical sense of preventing, not in the literal sense of keeping an attacker at bay.	*This cream will ward off premature sagging and wrinkles.*

Table C.1: The full constructicon of causal constructions.

APPENDIX D

Counts of Connective Patterns in the BECAUSE Corpus

The table below shows the (approximate) number of times each connective appears in the BECAUSE 2.1 corpus, where connectives are distinguished by unique sequences of lemmas. Note that this conflates some constructions incorrectly (e.g., *to* as in *We have the votes to pass it*, where it means "needed to," and *to* as in *I left early to pick up the kids*, where it means "with the goal of." Distinguishing connectives by lemma sequence also separates related constructions (e.g., *too* __ *for* __ and *too* __ *for* __ *to* __).

The entries in the table are sorted by count (descending), then entropy (ascending; see §8.7), then type, then lexicographically by connective. The abbreviations used are the same as those in Appendix C.

Connective	**Count**	**Entropy**	**Type**
to	204	1.000	Prep.
because	118	0.837	CS
for	113	0.484	Prep.
if	108	0.988	CS
when	79	0.979	Adv.
so	70	0.976	CC
after	69	0.972	Prep.
make	65	0.871	Verb
allow to	48	0.242	Verb
help	34	0.000	Verb
because of	34	0.986	Adv.
cause	33	0.750	Verb
as	31	0.300	CS
why	30	0.439	Adv.
since	24	0.881	CS
prevent	24	0.935	Verb
until	24	0.949	Prep.
too to	22	0.750	Cplx
lead to	20	0.894	Verb
enough to	18	0.684	Cplx
need to	17	0.206	Cplx
ensure	15	0.000	Verb
create	15	0.926	Verb
given	14	0.000	Prep.
in	13	0.051	Cplx
with	13	0.135	Prep.
prevent from	13	0.371	Verb
make sure	12	0.000	Cplx
mean	12	0.928	Verb
let	12	0.963	Verb
avoid	11	0.896	Verb
allow	10	0.663	Verb
force to	10	0.792	Verb
block	10	0.998	Verb
unless	9	0.469	CS
result	9	0.857	Noun
require to	9	0.954	Verb
once	9	0.998	Prep.
from	8	0.115	Prep.
during	8	0.598	Prep.

Connective	Count	Entropy	Type
result in	8	0.722	Verb
effect	8	0.997	Noun
before	7	0.380	Prep.
for reason	7	0.544	Cplx
without	7	0.650	Prep.
hamper	6	0.000	Verb
blame for	6	0.592	Verb
reason	6	0.650	Noun
thus	6	0.811	CS
thanks to	6	0.863	CS
cause to	6	0.985	Verb
reason to	6	1.000	Noun
attribute to	5	0.000	Verb
compel to	5	0.000	Verb
stem from	5	0.000	Verb
have	5	0.033	Verb
while	5	0.331	CS
require	5	0.581	Verb
help to	5	0.706	Verb
consequence	5	0.863	Noun
prompt	5	0.863	Verb
in effort to	5	0.961	Cplx
as result	5	0.991	Cplx
if then	5	0.991	CS
in order to	5	0.994	Cplx
spark	4	0.000	Verb
have to to	4	0.164	Cplx
into	4	0.190	Prep.
avert	4	0.722	Verb
in light of	4	0.722	Cplx
spur	4	0.722	Verb
come from	4	0.742	Verb
contribute to	4	0.918	Verb
necessary to	4	0.946	Cplx
discourage	4	0.985	Verb
guarantee	4	1.000	Verb
consequently	3	0.000	Adv.
facilitate	3	0.000	Verb
impede	3	0.000	Verb
on ground that	3	0.000	Cplx
open door to	3	0.000	Cplx
therefore	3	0.000	Adv.
of	3	0.011	Prep.
the the	3	0.017	Cplx
now that	3	0.422	Prep.
follow	3	0.480	Verb
as long as	3	0.811	Cplx
encourage	3	0.811	Verb
too for to	3	0.811	Cplx
come after	3	0.881	Verb
in attempt to	3	0.881	Cplx
responsible for	3	0.918	Adj.
essential to	3	0.971	Adj.
amid	3	0.985	Prep.
in response to	3	0.985	Cplx
promote	3	0.985	Verb
by reason of	2	0.000	Cplx
contingent on	2	0.000	Adj.
deter	2	0.000	Verb
foster	2	0.000	Verb
hinder	2	0.000	Verb
inspire	2	0.000	Verb
provoke	2	0.000	Verb
with goal of	2	0.000	Cplx
by	2	0.031	Prep.
at	2	0.038	Prep.
take	2	0.089	Verb
where	2	0.137	CS
then	2	0.183	CC
get to	2	0.222	Verb
need to to	2	0.286	Cplx
for to have to	2	0.414	Cplx
it take to	2	0.426	Cplx
produce	2	0.523	Verb
give rise to	2	0.559	Cplx
depend on	2	0.722	Verb
require for	2	0.863	Verb
as result of	2	0.918	Cplx
avoidance	2	0.918	Noun
enable to	2	0.918	Verb
for purpose	2	0.918	Cplx
generate	2	0.918	Verb
hence	2	0.918	Adv.
in the wake of	2	0.918	Cplx
on condition of	2	0.918	Cplx
prompt to	2	0.918	Verb
result from	2	0.918	Verb
whenever	2	0.918	CS
critical to	2	0.971	Adj.
reason for	2	0.971	Noun
condition on	2	1.000	Verb
enough for to	2	1.000	Cplx
for to must	2	1.000	Cplx
sufficient to	2	1.000	Cplx
thwart	2	1.000	Verb
trigger	2	1.000	Verb
attributable to	1	0.000	Adj.
blame on	1	0.000	Verb

Connective	Count	Entropy	Type
clear the way for	1	0.000	Cplx
engender	1	0.000	Verb
foil	1	0.000	Verb
for the sake of	1	0.000	Cplx
ground for	1	0.000	Cplx
in order to have to	1	0.000	Cplx
incite	1	0.000	Verb
insufficient to	1	0.000	Cplx
lest	1	0.000	CS
must in order to	1	0.000	Cplx
necessary condition of	1	0.000	Cplx
necessary if be to	1	0.000	Cplx
necessitate	1	0.000	Verb
open the way for to	1	0.000	Cplx
pave the way for to	1	0.000	Cplx
precipitate	1	0.000	Verb
predicate on	1	0.000	Verb
reason be because	1	0.000	Cplx
reason why	1	0.000	Cplx
unless then	1	0.000	Cplx
ward off	1	0.000	Verb
with objective of	1	0.000	Cplx
within after	1	0.000	Cplx
should	1	0.067	CS
force	1	0.078	Verb
take to	1	0.106	Cplx
out of	1	0.107	Prep.
in case	1	0.149	Cplx
make for	1	0.196	Verb
help in	1	0.229	Verb
must to	1	0.242	Cplx
need for	1	0.242	Cplx
bring to	1	0.310	Verb
stem	1	0.310	Verb
bar	1	0.337	Verb
in case of	1	0.353	Cplx
eliminate	1	0.414	Verb
else	1	0.439	Adv.
blame	1	0.469	Verb
too for	1	0.469	Cplx
force into	1	0.503	Verb
ease	1	0.544	Verb
in response	1	0.544	Cplx
give	1	0.559	Verb
arise	1	0.592	Verb
bring on	1	0.592	Verb
feed	1	0.650	Verb
key to	1	0.650	Adj.
so as to	1	0.650	Cplx
encourage to	1	0.722	Verb
in view of	1	0.722	Cplx
on ground	1	0.722	Cplx
set off	1	0.722	Verb
assure	1	0.811	Verb
block from	1	0.811	Verb
following	1	0.811	Prep.
vital to	1	0.811	Adj.
enable	1	0.918	Verb
every time	1	0.918	Cplx
implication	1	0.918	Noun
in the face of	1	0.918	Cplx
keep from	1	0.918	Verb
sufficient for	1	0.918	Cplx
vital for	1	0.918	Adj.
adequate to	1	1.000	Adj.
arise from	1	1.000	Verb
for purpose of	1	1.000	Cplx
in advance of	1	1.000	Cplx
in the aftermath	1	1.000	Cplx
in the aftermath of	1	1.000	Cplx
induce	1	1.000	Verb
inhibit	1	1.000	Verb
it take for to	1	1.000	Cplx
make certain	1	1.000	Cplx
need if be to	1	1.000	Cplx
on ground of	1	1.000	Cplx

Table D.1: Connective pattern statistics. .

Bibliography

Peter Adolphs, Feiyu Xu, Hong Li, and Hans Uszkoreit. 2011. Dependency graphs as a generic interface between parsers and relation extraction rule learning. In Joscha Bach and Stefan Edelkamp, editors, *KI 2011: Advances in Artificial Intelligence: 34th Annual German Conference on AI, Berlin, Germany, October 2011. Proceedings*, pages 50–62. Springer Berlin Heidelberg, Berlin, Germany.

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers, Principles, Techniques*. Addison-Wesley, Reading, MA, USA.

Enrique Alfonseca, Katja Filippova, Jean-Yves Delort, and Guillermo Garrido. 2012. Pattern learning for relation extraction with a hierarchical topic model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 54–59. Association for Computational Linguistics.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2442–2452. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*. URL http://arxiv.org/abs/1409.0473.

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval'07 task 19: frame semantic structure extraction. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007)* , pages 99–104. Association for Computational Linguistics.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING 1998)*, pages 86–90. Association for Computational Linguistics.

Omid Bakhshandeh, Alexis Cornelia Wellwood, and James Allen. 2016. Learning to jointly predict ellipsis and comparison structures. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 62–74. Association for Computational Linguistics.

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*, volume 2, pages 267–292. CRC Press, Boca Raton, USA.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 349–359. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse (LAW VII & ID)*, pages 178–186. Association for Computational Linguistics.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2670–2676.

Dagmar Barth. 2000. "that's true, although not really, but still": Expressing concession in spoken English. *Topics in English Linguistics*, 33:411–438.

Cosmin Adrian Bejan and Chris Hathaway. 2007. UTD-SRL: A pipeline architecture for extracting frame semantic structures. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 460–463. Association for Computational Linguistics.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1499–1510. Association for Computational Linguistics, Doha, Qatar.

Benjamin Bergen and Nancy Chang. 2005. Embodied Construction Grammar in simulation-based language understanding. *Construction grammars: Cognitive grounding and theoretical extensions*, 3:147–190.

Steven Bethard, William J Corvey, Sara Klingenstein, and James H. Martin. 2008. Building a corpus of temporal-causal structure. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, pages 908–915. European Language Resources Association.

Hans C. Boas. 2013. Cognitive Construction Grammar. In Hoffmann and Trousdale (2013), pages 233–254.

Claire Bonial, Bianca Badarau, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Tim O'Gorman, Martha Palmer, and Nathan Schneider. 2018 (in press). Abstract Meaning Representation of constructions: The more we include, the better the representation. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association.

Claire Bonial, Julia Bonn, Kathryn Conger, Jena D. Hwang, and Martha Palmer. 2014. PropBank: Semantics of new predicate types. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 3013–3019. European Languages Resources Association.

Claire Bonial, Susan Windisch Brown, Jena D Hwang, Christopher Parisien, Martha Palmer, and Suzanne Stevenson. 2011. Incorporating coercive constructions into a verb lexicon. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 72–80. Association for Computational Linguistics.

Ted Briscoe and John Carroll. 1993. Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, pages 1–10. Association for Computational Linguistics.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In Hwee Tou Ng and Ellen Riloff, editors, *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL 2004)*, pages 89–97. Association for Computational Linguistics.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL 2005)*, pages 152–164. Association for Computational Linguistics.

Jill Carrier and Janet H Randall. 1992. The argument structure and syntactic structure of resultatives. *Linguistic inquiry*, pages 173–234.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 740–750. Association for Computational Linguistics.

Wenliang Chen, Yue Zhang, and Min Zhang. 2014. Feature embedding for dependency parsing. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers (COLING 2014)*, pages 816–826. Association for Computational Linguistics.

Jinho D. Choi and Andrew Mccallum. 2013. Transition-based dependency parsing with selectional branching. In *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1052–1062. Association for Computational Linguistics.

Jinho D. Choi and Martha Palmer. 2011a. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT '11)*, volume 2, pages 687–692. Association for Computational Linguistics.

Jinho D Choi and Martha Palmer. 2011b. Transition-based semantic role labeling using predicate argument clustering. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 37–45. Association for Computational Linguistics.

Noam Chomsky and Howard Lasnik. 1993. The theory of principles and parameters. In *Syntax: An international handbook of contemporary research*, volume 1, pages 506–569. Walter de Gruyter.

Massimiliano Ciaramita, Giuseppe Attardi, Felice Dell'Orletta, and Mihai Surdeanu. 2008. DeSRL: A linear-time semantic role labeling system. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 258–262. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Juliette Conrath, Stergos Afantenos, Nicholas Asher, and Philippe Muller. 2014. Unsupervised extraction of semantic relations using discourse cues. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 2184–2194. Dublin City University and Association for Computational Linguistics.

William Croft. 2001. *Radical Construction Grammar: Syntactic theory in typological perspective*. Oxford University Press.

William Croft, Pavlina Pešková, and Michael Regan. 2016. Annotation of causal and aspectual structure of events in RED: a preliminary report. In *Proceedings of the Fourth Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 8–17. Association for Computational Linguistics.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Mark Davies. 2008. The corpus of contemporary American English (COCA): 520 million words, 1990-present. URL https://corpus.byu.edu/coca/.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D. Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 449–454. European Language Resources Association.

Dun Deng and Nianwen Xue. 2014. Aligning Chinese-English parallel parse trees: Is it feasible? In *Proceedings of the 8th Linguistic Annotation Workshop (LAW VIII)*, pages 29–37. Association for Computational Linguistics.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics.

Phil Dowe. 2008. Causal processes. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition. http://plato.stanford.edu/archives/fall2008/entries/causation-process/.

Stuart Dreyfus and Robert Wagner. 1971. The Steiner problem in graphs. *Networks*, 1(3):195–207.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2015. Annotating causal language using corpus lexicography of constructions. In *Proceedings of The 9th Linguistic Annotation Workshop (LAW IX)*, pages 188–196. Association for Computational Linguistics, Denver, CO.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2017a. Automatically tagging constructions of causation and their slot-fillers. In *Transactions of the Association for Computational Linguistics*, volume 5, pages 117–133. Association for Computational Linguistics.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2017b. The BECauSE corpus 2.0: Annotating causality and overlapping relations. In *Proceedings of the 11th Linguistic Annotation Workshop (LAW XI)*, pages 95–104. Association for Computational Linguistics.

Jesse Dunietz, Lori Levin, and Miriam Petruck. 2017c. Construction detection in a conventional NLP pipeline. In *Proceedings of the 2017 AAAI Spring Symposium on Computational Construction Grammar and Natural Language Understanding*. AAAI Press.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, pages 334–343. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545. Association for Computational Linguistics.

Charles J. Fillmore. 2006. Toward the linking of text annotations, the FrameNet lexicon, and an intended future constructicon. Presentation at the Fourth International Conference on Construction Grammar, Tokyo, Japan.

Charles J. Fillmore. 2008. Border conflicts: FrameNet meets Construction Grammar. In Elisenda Bernal and Janet DeCesaris, editors, *Proceedings of the XIII EURALEX International Congress*, pages 49–68. Universitat Pompeu Fabra, Universitat Pompeu Fabra.

Charles J. Fillmore, Paul Kay, and Mary Catherine O'Connor. 1988. Regularity and idiomaticity in grammatical constructions: The case of *let alone*. *Language*, 64(3):501–538.

Charles J. Fillmore, Russell Lee-Goldman, and Russell Rhodes. 2012. The FrameNet constructicon. *Sign-Based Construction Grammar*, pages 309–372.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370. Association for Computational Linguistics.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 17–21. Association for Computational Linguistics.

Dan Flickinger. 2011. Accuracy vs. robustness in grammar engineering. In Emily M. Bender and Jennifer E. Arnold, editors, *Language from a cognitive perspective: Grammar, usage, and processing*, pages 31–50. CSLI Publications.

William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 279–288. Association for Computational Linguistics.

King Sun Fu. 1968. *Sequential Methods in Pattern Recognition and Machine Learning*. Mathematics in Science and Engineering. Elsevier Science.

Felix A Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *Neural Networks, 2000. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000)*, volume 3, pages 189–194. IEEE.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics* , 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 239–246. Association for Computational Linguistics.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering*, pages 76–83. Association for Computational Linguistics.

Roxana Girju, Stan Szpakowicz, Preslav Nakov, Peter Turney, Vivi Nastase, and Deniz Yuret. 2007. SemEval-2007 task 04: Classification of semantic relations between nominals. In *In Fourth International Workshop on Semantic Evaluations (SemEval-2007)*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. Proceedings of Machine Learning Research.

Adele Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. Chicago University Press.

Adele E. Goldberg. 2006. *Constructions at Work: The Nature of Generalization in Language*. Oxford linguistics. Oxford University Press.

Adele E. Goldberg. 2013. Constructionist approaches. In Hoffmann and Trousdale (2013), pages 15–31.

Yoav Goldberg and Graeme Hirst. 2017. *Neural Network Methods in Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Cécile Grivaz. 2010. Human judgements on causation in French texts. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC 2010)*. European Languages Resources Association (ELRA).

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth SIGNLL Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18. Association for Computational Linguistics.

Patrick Hanks. 2013. *Lexical analysis: Norms and exploitations*. MIT Press.

Yoko Hasegawa, Russell Lee-Goldman, Kyoko Hirose Ohara, Seiko Fujii, and Charles J. Fillmore. 2010. On expressing measurement and comparison in English and Japanese. *Contrastive construction grammar*, pages 169–200.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational Linguistics (COLING '92)*, volume 2, pages 539–545. Association for Computational Linguistics.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 178–182. Association for Computational Linguistics.

Christopher Hidey and Kathleen McKeown. 2016. Identifying causal relations using parallel Wikipedia articles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1424–1433. Association for Computational Linguistics.

G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Jerry R Hobbs. 2005. Toward a useful concept of causality for lexical semantics. *Journal of Semantics*, 22(2):181–209.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. *A Field Guide to Dynamical Recurrent Neural Networks*, chapter Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, pages 237–243. IEEE Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Thomas Hoffmann and Graeme Trousdale. 2013. *The Oxford Handbook of Construction Grammar*. Oxford Handbooks in Linguistics. Oxford University Press USA.

Jena D. Hwang, Rodney D. Nielsen, and Martha Palmer. 2010. Towards a domain independent semantics: Enhancing semantic representation with construction grammar. In *Proceedings of the NAACL HLT Workshop on Extracting and Using Constructions in Computational Linguistics*, pages 1–8. Association for Computational Linguistics.

Jena D. Hwang and Martha Palmer. 2015. Identification of caused motion constructions. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (* SEM 2015)*, pages 51–60.

Nancy Ide, Christiane Fellbaum, Collin Baker, and Rebecca Passonneau. 2010. The manually annotated sub-corpus: A community resource for and by the people. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 68–73. Association for Computational Linguistics.

Rei Ikuta, Will Styler, Mariah Hamang, Tim O'Gorman, and Martha Palmer. 2014. Challenges of adding causation to Richer Event Descriptions. In *Proceedings of the Second Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 12–20. Association for Computational Linguistics.

Ashwin Ittoo and Gosse Bouma. 2011. Extracting explicit and implicit causal relations from sparse, domain-specific texts. In *Natural Language Processing and Information Systems*, pages 52–63. Springer.

Paul Jaccard. 1912. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50.

Richard Johansson and Pierre Nugues. 2007a. Incremental dependency parsing using online learning. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 1134–1138. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2007b. LTH: Semantic structure extraction using nonprojective dependency trees. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 227–230. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2017. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (draft)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, third edition.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL 2003)*, pages 423–430. Association for Computational Linguistics.

Zornitsa Kozareva and Eduard Hovy. 2010. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1482–1491. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Lori Levin, Teruko Mitamura, Davida Fromm, Brian MacWhinney, Jaime Carbonell, Weston Feely, Robert Frederking, Anatole Gershman, and Carlos Ramirez. 2014. Resources for the detection of conventionalized metaphors in four languages. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 498–501. European Language Resources Association.

Roger Levy and Galen Andrew. 2006. TRegex and TSurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 2231–2234. European Language Resources Association.

Barbara Lewandowska-Tomaszczyk. 2007. Polysemy, prototypes, and radial categories. *The Oxford handbook of cognitive linguistics*, pages 139–169.

Hong Li, Sebastian Krause, Feiyu Xu, Andrea Moro, Hans Uszkoreit, and Roberto Navigli. 2015. Improvement of n-ary relation extraction by adding lexical semantics to distant-supervision rule learning. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, volume 2, pages 317–324. SCITEPRESS, Science and Technology Publications, Lda.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A PDTB-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530. Association for Computational Linguistics.

Ken Litkowski and Orin Hargraves. 2005. The Preposition Project. In *Proceedings of the Second ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, pages 171–179. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Gus Hahn-Powell Dane Bell Marco and A Valenzuela-Escárcega Mihai Surdeanu. 2016. This before that: Causal precedence in the biomedical domain. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 146–155. Association for Computational Linguistics.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology (HLT '94)*, pages 114–119. Association for Computational Linguistics.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL 2012)*, pages 523–534. Association for Computational Linguistics.

Jonathan May. 2016. SemEval-2016 task 8: Meaning representation parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, pages 1063–1073. Association for Computational Linguistics.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval 2017)*, pages 536–545. Association for Computational Linguistics.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. Annotating noun argument structure for NomBank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*. Association for Computational Linguistics.

Laura A Michaelis. 2013. Construction grammar and the syntax-semantics interface. In *The Bloomsbury Companion to Syntax*, pages 421–435. Bloomsbury Publishing.

Claudiu Mihăilă. 2014. *Discourse Causality Recognition in the Biomedical Domain*. Ph.D. thesis, The University of Manchester, Manchester, UK.

Claudiu Mihăilă, Tomoko Ohta, Sampo Pyysalo, and Sophia Ananiadou. 2013. BioCause: Annotating and analysing causality in the biomedical domain. *BMC Bioinformatics*, 14(1):2.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. URL https://arxiv.org/abs/1301.3781, arXiv:1301.3781.

Paramita Mirza, Rachele Sprugnoli, Sara Tonelli, and Manuela Speranza. 2014. Annotating causality in the TempEval-3 corpus. In *Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language (CAtoCL)*, pages 10–19.

Paramita Mirza and Sara Tonelli. 2014. An analysis of causality between events and its relation to temporal information. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 2097–2106.

Erwan Moreau and Isabelle Tellier. 2009. The Crotal SRL system : a generic tool based on tree-structured CRF. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 91–96. Association for Computational Linguistics.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the 4th Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 51–61. Association for Computational Linguistics.

Ion Muslea. 1999. Extraction patterns for information extraction tasks: A survey. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 1–6. AAAI Press.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814. Omnipress, Madison, Wisconsin, USA.

Ad Neeleman and Hans Van de Koot. 2012. *The Theta System: Argument Structure at the Interface*, chapter The Linguistic Expression of Causation, pages 20–51. Oxford University Press.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. URL https://arxiv.org/abs/1701.03980, arXiv:1701.03980.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160. Association for Computational Linguistics.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Stephan Oepen, Jonathon Read, Tatjana Scheffler, Uladzimir Sidarenka, Manfred Stede, Erik Velldal, and Lilja Øvrelid. 2016. OPT: Oslo-Potsdam-Teesside. pipelining rules, rankers, and classifier ensembles for shallow discourse parsing. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016): Shared Task*, pages 20–26. Association for Computational Linguistics.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods Treebank: Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, volume 2, pages 1–5. Association for Computational Linguistics.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer Event Description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 47–56. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*. European Language Resources Association.

James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. 2003. ISO-TimeML: An international standard for semantic annotation. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*.

Peng Qi and Christopher D Manning. 2017. Arc-swift: A novel transition system for dependency parsing. URL https://arxiv.org/abs/1705.04434, arXiv:1705.04434.

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–10. Association for Computational Linguistics.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 41–47. Association for Computational Linguistics.

Stefan Riezler. 2014. On the problem of theoretical terms in empirical computational linguistics. *Computational Linguistics*, 40(1):235–245.

Ellen Riloff. 1996. *Automatically generating extraction patterns from untagged text*, volume 2, pages 1044–1049. Association for the Advancement of Artificial Intelligence.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence & 11th Innovative Applications of Artificial Intelligence Conference*, pages 474–479. AAAI Press & MIT Press.

Michael Ringgaard, Rahul Gupta, and Fernando C. N. Pereira. 2017. SLING: a framework for frame semantic parsing. URL http://arxiv.org/abs/1710.07032, arXiv:1710.07032.

F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386.

Michael Roth. 2016. Improving frame semantic parsing via dependency path embeddings. In *Book of Abstracts of the 9th International Conference on Construction Grammar*, pages 165–167.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1192–1202. Association for Computational Linguistics.

Grzegorz Rozenberg, editor. 1997. *Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R Johnson, Charles Baker, and Jan Scheffczyk. 2016. FrameNet II: Extended theory and practice.

Ivan A Sag. 1997. English relative clause constructions. *Journal of linguistics*, 33(2):431–483.

Ivan A. Sag and Hans C. Boas, editors. 2012. *Sign-based construction grammar*. CSLI Publications.

Mark Sammons, VG Vinod Vydiswaran, and Dan Roth. 2011. Recognizing textual entailment. In *Multilingual Natural Language Applications: From Theory to Practice*. Prentice Hall.

Yvonne Samuelsson, Oscar Täckström, Sumithra Velupillai, Johan Eklund, Mark Fishel, and Markus Saers. 2008. Mixing and blending syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 248–252. Association for Computational Linguistics.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia* .

Jonathan Schaffer. 2014. The metaphysics of causation. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2014 edition. http://plato.stanford.edu/archives/sum2014/entries/causation-metaphysics/.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Meredith Green, Abhijit Suresh, Kathryn Conger, Tim O'Gorman, and Martha Palmer. 2016. A corpus of preposition supersenses. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 99–109. Association for Computational Linguistics.

Nathan Schneider and Noah A. Smith. 2015. A corpus and model integrating multiword expressions and supersenses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015)*, pages 1537–1547. Association for Computational Linguistics.

Karin K. Schuler. 2005. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA. AAI3179808.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 304–311. Association for Computational Linguistics.

Noah A. Smith, Claire Cardie, Anne L. Washington, and John Wilkerson. 2014. Overview of the 2014 NLP unshared task in poliinformatics. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. Association for Computational Linguistics.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*.

Luc Steels. 2011. *Design Patterns in Fluid Construction Grammar*. Constructional Approaches to Language. John Benjamins Publishing Company.

Luc Steels. 2012. Computational issues in Fluid Construction Grammar. *Lecture Notes in Computer Science*, 7249.

Luc Steels. 2017. Basics of Fluid Construction Grammar. *Constructions and Frames*, 9(2).

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2001. Automatic pattern acquisition for Japanese information extraction. In *Proceedings of the First International Conference on Human Language Technology Research (HLT '01)*, pages 1–7. Association for Computational Linguistics.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL 2003)*, pages 224–231. Association for Computational Linguistics.

Weiwei Sun, Hongzhan Li, and Zhifang Sui. 2008. The integration of dependency relation classification and semantic role labeling using bilayer maximum entropy Markov models. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 243–247. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 159–177. Association for Computational Linguistics.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016)*, pages 187–197. Association for Computational Linguistics.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental RNNs and a syntactic scaffold. URL http://arxiv.org/abs/1706.09528, arXiv:1706.09528.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Leonard Talmy. 1988. Force dynamics in language and cognition. *Cognitive Science*, 12(1):49–100.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence (IJCAI '09)*, pages 1562–1567. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL)*, pages 173–180. Association for Computational Linguistics.

Stephen Tratz. 2011. *Semantically-Enriched Parsing for Natural Language Understanding*. Ph.D. thesis, University of Southern California, Los Angeles, CA, USA.

William M. Trochim. 2006. Reliability & validity. In *The Research Methods Knowledge Base, 2nd Edition*. Atomic Dog. http://www.socialresearchmethods.net/kb/relandval.php.

Marco A. Valenzuela-Escárcega, Gus Hahn-Powell, Dane Bell, and Mihai Surdeanu. 2016. SnapToGrid: From statistical to interpretable models for biomedical information extraction. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 56–65. Association for Computational Linguistics.

Marco A. Valenzuela-Escárcega, Gus Hahn-Powell, and Mihai Surdeanu. 2016. Odin's runes: A rule language for information extraction. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association.

Remi van Trijp. 2017. A computational construction grammar for English. In *Proceedings of the 2017 AAAI Spring Symposium on Computational Construction Grammar and Natural Language Understanding*. AAAI Press.

Carla Vergaro. 2008. Concessive constructions in english business letter discourse. *Text & Talk*, 28(1):97–118.

Laure Vieu, Philippe Muller, Marie Candito, and Marianne Djemaa. 2016. A general framework for the annotation of causality based on FrameNet. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association.

Jianxiang Wang and Man Lan. 2015. A refined end-to-end discourse parser. In *Proceedings of the Nineteenth SIGNLL Conference on Computational Natural Language Learning (CoNLL 2015): Shared Task*, pages 17–24. Association for Computational Linguistics.

Jianxiang Wang and Man Lan. 2016. Two end-to-end shallow discourse parsers for English and Chinese in CoNLL-2016 shared task. In *Proceedings of the Twentieth SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016): Shared Task*, pages 33–40. Association for Computational Linguistics.

Phillip Wolff, Bianca Klettke, Tatyana Ventura, and Grace Song. 2005. Expressing causation in English and other languages. In Woo-kyoung Ahn, Robert L. Goldstone, Bradley C. Love, Arthur B. Markman, and Phillip Wolff, editors, *Categorization inside and outside the laboratory: Essays in honor of Douglas L. Medin*, pages 29–48. American Psychological Association, Washington, DC, USA.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Christopher Bryant, Rashmi Prasad, and Attapol T. Rutherford. 2015. The CoNLL-2015 shared task on shallow discourse parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning (CoNLL 2015): Shared Task*, pages 1–16. Association for Computational Linguistics.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Attapol Rutherford, Bonnie Webber, Chuan Wang, and Hongmin Wang. 2016. CoNLL 2016 shared task on multilingual shallow discourse parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (CoNLL 2016):Shared Task*, pages 1–19. Association for Computational Linguistics.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT '11)*, volume 2, pages 188–193. Association for Computational Linguistics.

Hai Zhao, Wenliang Chen, Chunyu Kity, and Guodong Zhou. 2009. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 55–60. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1127–1137. Association for Computational Linguistics.