Grounded Knowledge Bases for Scientific Domains

Dana Movshovitz-Attias

CMU-CS-15-120

August 2015

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

Thesis Committee:
William W. Cohen, Chair
Tom Mitchell
Roni Rosenfeld
Alon Halevy, Google Research

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

This research was sponsored by SRI International under grant number 27001371, the National Institute of General Medicines under grant number 1R01GM081293, Google, and the National Science Foundation under grant numbers IIS-0811562, CCF-1247088, IIS-1250956 and CCF-1414030. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: grounded language learning, natural language processing, knowledge base construction, knowledge representation, statistical language modeling, unsupervised learning, semi supervised learning, bootstrapping, topic modeling, machine learning, probabilistic graphical models, ontology, grounding, information extraction.

For Yair, with my love.

Abstract

This thesis is focused on building knowledge bases (KBs) for scientific domains. Specifically, we create structured representations of technical-domain information using unsupervised or semi-supervised learning methods. This work is inspired by recent advances in knowledge base construction based on Web text. However, in the technical domains we consider here, in addition to text corpora we have access to the objects named by text entities, as well as data associated with those objects. For example, in the software domain, we consider the implementation of classes in code repositories, and observe the way they are being used in programs. In the biomedical realm, biological ontologies define interactions and relations between domain entities, and there is experimental information on entities such as proteins and genes. We consider the process of *grounding*, namely, linking entity mentions from text to external domain resources, including code repositories and biomedical ontologies, where objects can be uniquely identified. Grounding presents an opportunity for learning, not only how entities are discussed in text, but also what are their real-world properties.

The main contribution of this thesis is in addressing challenges from the following research areas, in the context of learning about technical domains: (1) *Knowledge representation:* How should knowledge about technical domains be represented and used? (2) *Grounding:* How can existing resources of technical domains be used in learning? (3) *Applications:* What applications can benefit from structured knowledge bases dedicated to scientific data?

We explore grounded learning and knowledge base construction for the biomedical and software domains. We first discuss approaches for improving applications based on well-studied statistical language models. Next, we construct a deeper semantic representation of domain-entities by building a grounded ontology, where entities are linked to a code repository, and through an adaption of an ontology-driven KB learner to scientific input. Finally, we present a topic model framework for knowledge base construction, which jointly optimizes the KB schema and learned facts, and show that this framework produces high precision KBs in our two domains of interest. We discuss extensions to our model that allow: first, incorporating human input, leading to a semi-supervised learning process, and second, grounding the modeled entities with domain data.

Acknowledgments

The past 5 years at CMU have been a crazy and fun experience. I survived this journey thanks to the help and support of my friends and colleagues.

I am grateful to my advisor, William Cohen, for transforming me from a Computational Biologist to a Natural Language Processing researcher. His experience and knowledge, his patience, humor, and easy-going nature have all made our meetings both enjoyable and insightful. I especially appreciate his approach to research: starting with a few interesting examples and simple solutions, while keeping in mind the broad context of the problem and relation to other areas.

I am fortunate to have in my thesis committee Tom Mitchell, Roni Rosenfeld and Alon Halevy. Tom has a vast knowledge of Artificial Intelligence and I appreciated getting his perspective on my work, always delivered with extraordinary kindness. It was a pleasure TAing for Roni, and learning from his interactive and engaging teaching style. I also had a fruitful and fun internship working in Alon's group at Google, where I learned a lot about ontologies, attributes and coffee. My conversations with everyone in the committee, your advice, and support, have been valuable to me.

My work was influenced by thoughts shared with the great individuals I had the opportunity to meet, work with, and TA with, including: Bhavana Dalvi Mishra, Katie Rivard Mazaitis, William Yang Wang, Frank Lin, Ramnath Balasubramanyan, Tae Yano, Ni Lao, Nan Li, Mahesh Joshi, Einat Minkov, Malcolm Greaves, Freddy Chua, Premkumar Devanbu, Lin Tan, Song Wang, Partha Pratim Talukdar, Ndapa Nakashole, Estevam Hruschka, my internship hosts and co-workers: Steven Whang, Natasha Noy, Alon Halevy, Eric Sun, and my TA collaborators: Ariel Procaccia, Emma Brunskill, Danai Koutra, Yair Movshovitz-Attias, Roni Rosenfeld and Ming Sun.

The surest way to make it through grad-school is with great friends. Thanks for the lunches, coffee breaks, parties, vacations, and for making it all so much fun! John Wright, Sarah Loos and Jeremy Karnowski, Gabe and Cat Weisz, Danai Koutra and Walter Lasecki, Aaditya Ramdas, Kate Taralova, Sam Gottlieb, João Martins, David Henriques,

Akshay Krishnamurthy, David Naylor, Galit Regev and Tzur Frenkel, Or Sheffet, Yuzi Nakamura, Mary Wootters, Jesse Dunietz, Nika Haghtalab, Deby Katz, Yu Zhao. Many thanks also to the administrative staff at CMU, who are always looking out for us. Thank you Deb Cavlovich, Catherine Copetas, Sharon Cavlovich and Sandy Winkler.

I thank my parents Meir and Eti Atias, and my brothers Nir and Ben Atias, for their long-distance support, their fun visits to Pittsburgh, and for their encouragement throughout my studies.

More than all, I thank my husband Yair Movshovitz-Attias, who has been here through it all. Together we have been through army service, undergrad and graduate studies. With your love, support, and mainly your endless stream of jokes, I know I can get anywhere. I am looking forward to more joint adventures.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Building Grounded Knowledge Bases

Algorithmic advances in the fields of Natural Language Processing and Machine Learning have enabled a surge in the construction of large-scale knowledge bases from Web resources. Knowledge bases such as YAGO [Suchanek et al., 2007, 2008], DBpedia [Auer et al., 2007], and Freebase [Google, 2011] are derived mainly from Wikipedia, while others are learned from large corpora of Web pages, including NELL [Carlson et al., 2010], Knowledge Vault [Dong et al., 2014], and TextRunner [Yates et al., 2007]. These systems transform unstructured input into some structured representation, which includes large collections of entities, a mapping of entities to semantic classes, and relations among them. In the Open Information Extraction paradigm, used for example in TextRunner, the entire system is extracted using a single pass over the corpora. Conversely, systems such as NELL use an iterative bootstrapping approach, where each iteration improves the existing KB. The existence of this large variety of KBs has promoted development of applications that are based on semantics and can take advantage of this type of structured knowledge, which did not previously exist in large scale. This in turn is contributing to the creation of improved KBs, including YAGO2 which integrates a spatio-temporal dimension [Hoffart et al., 2013], and YAGO3 which combines multilingual information [Mahdisoltani et al., 2014].

The KBs described above draw their strength from the plentitude of available Web data. They describe interesting entities and topics that are discussed and written about by people. Still, some information, in particular technical and scientific knowledge, is harder to reason over with existing NLP technology. Specialized domain terminology, which often includes

domain specific language constructs, affects standard techniques used for KB construction, including part-of-speech tagging, parsing, entity extraction, and relation understanding. For example, the phrase "Class not found exception" is recognized by standard English parsers as a verb relation between a class and an exception, which can be summarized as: NOT_FOUND(CLASS, EXCEPTION). However, as known to Java programmers, this is a type of an exception, meaning that it is a noun phrase.

Notably, technical information is not always conveyed through natural discourse. In this thesis, we consider challenges in building KBs for technical domains, specifically, the biomedical and software domains, where in addition to text corpora we have access to the objects named by text entities, as well as data associated with those objects. For example, in the software domain, we can consider the implementation of classes in a code repository, and we can observe the way they are being used statically or dynamically in programs. In the biomedical realm, biological ontologies define interactions and relations between entities in this domain, and there is experimental information on domain entities such as proteins and genes. Here, we consider the process of *grounding*, namely, linking entity mentions from text to external domain resources, including code repositories and biomedical ontologies, where domain objects can be uniquely identified. Grounding entities to the additional resources available in technical domains, present an opportunity for learning, not only how entities are discussed, but also how they are used and what are their real-world properties. Some properties can be learned from text, such as research literature from the domain, while others are only available through other sources. Hence, in this thesis, we strive for a combined approach.

The main contribution of this thesis is in answering questions from the following research areas, in the context of learning about a technical domain:

Knowledge Representation. *How should knowledge about a technical domain be represented and used?* The answer is often task-specific, and so in this thesis we consider tasks that range from comment prediction, which can make use of a shallow statistical language modeling, to tasks that require more structured knowledge, like information extraction.

Grounding. *How can existing resources of technical domains be used in learning?* Available resources, which can be linked to textual data, are domain-specific, and here we consider two cases: In the biomedical domain, many hand-built narrow ontologies are available, which describe subfields within biology; In the software realm, no ontologies exists, however, some of the objects and discourse (such as software modules) exist in digital form. We use biomedical ontologies as seed information to a semi-supervised bootstrapping learning system, and we ground software entities

by linking them with their implementation in a code repository.

Applications. *What applications can benefit from the use of structured knowledge bases dedicated to scientific data?* While this thesis is not focused on applications, we consider them as an instrument for better understanding the role of grounding in technical domains.

Table 1.1 summarizes our contributions in the context of the three research areas and the two technical domains we studied. Next, Section 1.2 formalizes the thesis statement. Section 1.3 provides a high-level roadmap, describing the development of the work along the following chapters. Finally, Section 1.4 gives a short synopsis of each chapter.

Software	Biomedical
Knowledge Representation	
• Statistical language modeling of software code [Chapter 2, 2013] • Grounded ontology construction using coordinate term relationships [Chapter 4, 2015b] • Topic model based approach for learning a complete knowledge base [Chapter 5, 2015a]	• Bootstrap knowledge base learning for the biomedical domain [Chapter 3, 2012b, 2012a] • Aligning relations from a grounded corpus with a topic-model guided knowledge base [Chapter 6]
Grounding	
• Grounded ontology construction using coordinate term relationships [Chapter 4, 2015b] • Grounding in statistical language models [Chapter 2, 2013]	• Relation extraction from a grounded corpus of annotated biomedical entities [Chapter 6]
Applications	
• Programming comments prediction [Chapter 2, 2013] • Extracting domain-specific facts from an open IE resource [Chapter 5, 2015a]	• Named entity recognition based on a learned gene lexicon [Chapter 3, 2012b, 2012a]

Table 1.1: Thesis overview. This table lists the contributions of each chapter in the thesis within the areas of Knowledge Representation, Grounding, and Application, as applied to the scientific domains of Software development or Biomedical research.

1.2 Thesis Statement

In this thesis I consider strings not only as text but as potential technical elements from a scientific domain. I then revisit core NLP problems associated with knowledge base construction and KB-driven information retrieval in the context of scientific domains. I show that the grounding process, linking entities in a target domain to specialized data, suggests modifications to the unsupervised and semi-supervised algorithms used to resolve NLP tasks in that domain. More formally:

> "Grounding entities to specialized data from a scientific domain facilitates improved unsupervised and semi-supervised algorithms for Knowledge Base construction for that domain"

1.3 Thesis Roadmap

We begin our study of information extraction for scientific applications by manipulating well-studied statistical language models for a software domain application (Chapter 2). Here, we are interested in the task of predicting code comments. Code and comment tokens from a code repository are modeled using statistical models with increasing levels of language and domain understanding: (1) the n-gram model, a shallow statistical model that has been found to be a strong language predictor; (2) the LDA topic model, which has a greater understanding of language semantics, and (3) an extension of LDA, called Link-LDA, which allows us to encode some basic domain understanding (namely, a distinction between code and text tokens). We show that a surprisingly large fraction of comment text is predictable, as we were able to complete up to 47% of comment characters. This is especially true for comments within a single project that describe similar classes, methods, and algorithms. Notably, performance on this task does not seem to correlate with our intuitive notion of what it means to understand text, as we found that the n-gram model performed best on this task. However, different models lead to different predictions. Moreover, intuitively, encoding domain understanding into the topic models lead to improved performance using the Link-LDA model. We hypothesized that gaining a better understanding of the entities in the domain, could further improve performance. Consider the following comment fragment:

> "In this method we read the next `double` from the input"

The type `double` could reasonably be replaced by `float`, and even `long` or `int`, as they all belong in the category of numerical data types. In this example, understanding

the semantics and the underlying categorical structure of our domain can be helpful in the prediction task.

Knowledge bases provide a standard NLP framework for gaining categorical and semantic understanding about the information they model. Many existing KB population systems rely on an input corpus and a predefined ontology in the form of a hierarchy of categories describing a domain of interest. The system then populates the KB with facts from the corpus that match the predefined categories. However, to the best of our knowledge, no extensive software ontology exists, and so, in Chapter 3 we turn to the biomedical domain, which has many existing ontologies, describing categories from Life Science and Chemistry. In this work we aim at constructing a deeper representation of domain-specific entities and knowledge found in the text, by constructing a KB for this domain. In particular, we adapt a semi-supervised bootstrapping algorithm for knowledge base population, using biomedical text and the existing domain ontologies. We show here that KB population is possible in technical domains, even using existing ontologies that were not defined for the purpose of KB population, with modest extensions to existing bootstrapping methods.

The main disadvantage of a KB population approach is that it relies on an input ontology definition, and therefore cannot be used to explore domains where no extensive ontology exists, such as the software domain. This motivates our work in Chapters 4 and 5, where we explore different approaches to developing ontologies from corpora alone. In Chapter 4 we construct a grounded software ontology for entities that refer to Java classes. We use distant supervision to predict coordinate relations, which indicate similarity, among pairs of class entities that were discovered in a text corpus. We describe a method for linking mentions the classes in text to a specific implementation in a target repository, and we then describe a distributional similarity measure over pairs of Java classes, given their implementation. We combine this information with distributional similarity measures based on the text corpus, and finally we predict pairwise similarity based on all signals. By aggregating the predicted similarity pairs, we create an ontology that can be used by ontology-driven KB population approaches such as the one described in Chapter 3. The advantage of this ontology is that it reflects statistics in language and code; this is in contrast to human-created ontologies such as the ones from the biomedical domain, or those typically used in open-domain KB population methods. The main limitation of this method is that grounding software entities directly to code limits the scope of the learned ontology, as it can only reason over entities that appear directly in the code. The resulting ontology is, therefore, missing higher level software concepts, including ones that discuss the users of the applications, the computer resources it consumes, and the design patterns used in its implementation. This lead us to a more general approach for KB construction

which exploits more information found in the text.

In Chapter 5 we describe our main contribution: a novel model for corpus-driven KB construction, named KB-LDA, in which the schema and the facts are both learned from the corpus. The model extends topic models and block stochastic block models, and it is unsupervised. This means that it learns the optimal latent structure of the input corpus together with the best-matching facts for the structure. It learns from relations extracted using basic text patterns, combined with corpus-wide statistics. The result is a KB with a hierarchical topic structure and typed relation definitions among the topics. Using this approach we have built a software KB which goes beyond the code and includes concepts for programming paradigms, design patterns, databases, and a variety of software platforms and tools.

The KB construction model described above does not take into account pre-existing domain knowledge. This fact makes it useful for exploration of new domains, as we have demonstrated on the software domain. However, in domains where high-quality knowledge exists, such as human-curated biomedical ontologies, can this data be used to improve the KB learning process? By how much? To answer these questions, we return to the study of the biomedical domain. In Chapter 6, we analyze an alignment of grounded relations emerging from biomedical ontologies with those learned from a corpus using KB-LDA. We show that: (1) our model finds many known entities and relations, which validates our results; (2) KB-LDA discovers new entities and relations, showing the added value of learning from language statistics, even in this well-researched domain; and (3) some entities and relations are only found in the known relations, which indicates the potential of grounding the KB learning process.

Finally, in Chapter 7 we discuss potential extensions to our KB construction model that allow: first, incorporating human input, leading to a semi-supervised learning process, and second, grounding the modeled entities with domain data. We further discuss how scientific knowledge bases can be used to improve domain-specific language modeling.

To summarize, we refer the reader back to the key idea presented in the thesis statement: In this thesis we address tasks from technical domains through a semantic understanding of those domains, which we achieve by constructing knowledge bases. We have found that grounding the entities found in domain corpora to specialized resources that uniquely identify objects in the domain, can improve KB building for that domain. Throughout the different projects described in this thesis, we have demonstrated a variety of methods for improving KB construction that are supported by the specialized resources available in the software and biomedical domain.

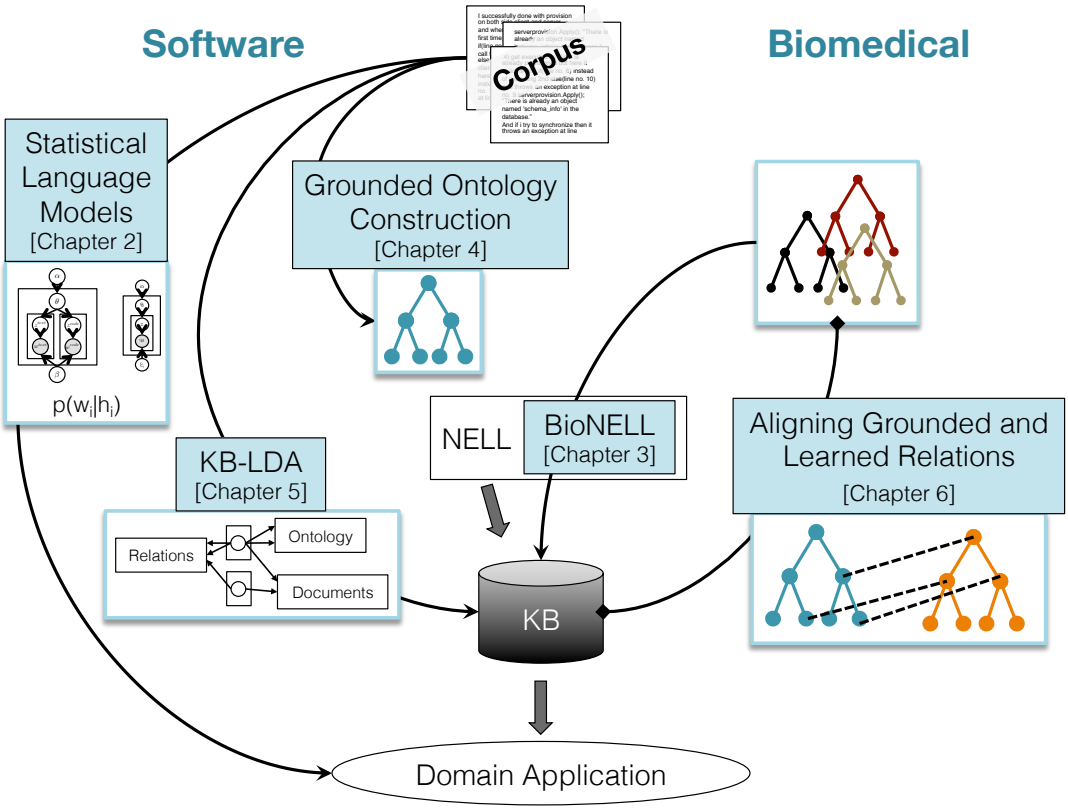


Figure 1.1: Roadmap diagram. Each chapter in the thesis makes a contribution in transforming input resources (e.g., a corpus, biomedical ontologies, or a code repository) for the task of building a knowledge base or addressing a domain specific application. In Chapter 2 we build models directly based on a code repository corpus in order to address a software domain task. Then, in Chapter 3 we use a higher level reasoning of language, by building a KB for the biomedical domain, based on biomedical ontologies. In Chapter 4 we build a software ontology from corpus, and in Chapter 5 we start from a corpus and build a complete KB. Finally, in Chapter 6 we compare relations from a learned KB, with relations originating in biomedical ontologies. For more details see Section 1.3.

Figure 1.1 summarizes the progression between thesis chapters and the way they support the thesis statement. In Chapter 4 we make a direct leap from corpus to a task in the software domain, which leads us to the conclusion that greater domain understanding can contribute to an improved solution for the domain application. Then, in Chapter 3

we proceed to building a knowledge base for the biomedical domain using existing domain ontologies. However, similar resources do not exist in the software domain, and so in Chapter 4 we construct a grounded software ontology, describing Java classes, from a combination of a corpus and code repository. Then, in Chapter 5 we take a more general approach and build a corpus-driven KB, describing a large variety of software concepts. In Chapter 6 we use the same framework to build a biomedical KB, and we close the loop, by comparing our the learned KB relations with ones that emerge directly from biomedical ontologies.

1.4 Chapter Synopsis

Statistical language modeling of software code [Chapter 2]

We consider an application of statistical language modeling to the software domain, illustrating the potential of structured knowledge in assisting downstream software understanding tasks. Given an implementation of a Java class, we are interested in predicting a natural language description matching the implementation. Beyond a summarization of the conceptual idea behind the code, this type of description can be viewed as a form of document expansion, providing significant terms relevant to the implementation. We experiment with LDA, link-LDA and n-gram models to model the correlations between code segments and the comments that describe them. With these we model local syntactic dependencies, or term relevancy based on the topic of the code, which are used to predict the main class comment of Java classes.

We evaluate our models based on their comment-completion capability in a setting similar to code completion tools that are built into standard code editors, and show that they can save a significant amount of typing. We also implemented a plugin for the Eclipse IDE based on one of the models, which assists in comment completion in real time.

Bootstrap knowledge base learning for the biomedical domain [Chapter 3]

Motivated by recent advances in knowledge base systems extracted from the Web, in this work we describe an open information extraction system for biomedical text. Leveraging the large available collections of biomedical data, in our system, an initial ontology and set of example seeds are automatically derived from existing resources. This knowledge

base is then populated using a coupled semi-supervised bootstrapping approach, based on NELL, which uses multiple set expansion techniques which are combined by a joint learning objective. As part of this work, we show that NELL's bootstrapping method is susceptible to ambiguous starting seeds, and can quickly lead to an accumulation of erroneous terms and contexts when learning a semantic class. We address this problem by introducing a method for assessing seed quality at each bootstrapping iteration, using point-wise mutual information.

We analyzed open biomedical categories learned with our system, based on dictionaries taken from Freebase, and we show that the proposed algorithm produces significantly more precise classes. Additionally, we used learned gene lexicons to improve annotations in a named-entity recognition task.

Grounded Software Ontology Construction using Coordinate Term Relationships [Chapter 4]

Discovering semantic relations between text entities is a key task in natural language understanding, which is a critical component that enables the success of knowledge representation systems. We examine coordinate relations between text entities that potentially refer to Java classes. Usually, relations are found by examining corpus statistics associated with text entities. Here we explore the idea of grounding the relation discovery with information about the class implementation, and coupling this with corpus statistics. To this end, we develop a similarity measure for Java classes using distributional information about how they are used in software, which we combine with corpus statistics on the distribution of contexts in which the classes appear in text that discusses code.Our results are verified by human labeling and are also compared with coordinate relations extracted using high-precision Hearst patterns.

We see this work as a first step towards building a knowledge representation system for the software domain, in which text entities refer to elements from a software code base, including classes, methods, applications, and programming languages. As an initial step, we combine the predicted coordinate pairs from this study, by aggregating them into a graph where entities are nodes and edges are determined by a coordinate term relation. Using methods for community detection on graphs, we discover that highly connected component in the resulting graph correspond to functional software groups, such as UI elements, utility classes, exceptions, and graphic objects. This hierarchy highlights class interactions that cannot be directly recovered through traditional software taxonomies, such as the type hierarchy or class namespace.

Topic Model Based Approach for Learning a Complete Knowledge Base [Chapter 5]

Many existing knowledge bases, including Freebase, Yago, and NELL, rely on a fixed ontology, given as an input to the system, which defines the data to be cataloged in the KB, i.e., a hierarchy of categories and relations between them. The system then extracts facts that match the predefined ontology, where in some cases, the input ontology is later extended with automatically discovered relations. We propose an unsupervised topic model, named KB-LDA, that jointly learns a latent ontological structure, including typed relations, of an input corpus, and identifies facts from the corpus that match the learned structure. Our approach combines mixed membership stochastic block models and topic models to infer a structure by jointly modeling text, a latent concept hierarchy, and latent semantic relationships among the entities mentioned in the text. The model learns the optimal latent structure of the corpus together with the best-matching facts. As a case study, we apply the model to a corpus of Web documents from the software domain, and learn a software KB. We evaluate the accuracy of the various components of the learned KB, and compare our results to human provided labels, and to relations extracted from an open IE resource.

Aligning Grounded and Learned Relations: A Comparison of Relations From a Grounded Corpus with a Topic-Model Guided Knowledge Base [Chapter 6]

KB-LDA is an unsupervised topic model for knowledge base construction. As a case study, the model was used to create a knowledge base for the software domain, for which the were no existing structured knowledge resources. In contrast, in the biomedical domain, many ontologies exist which describe sub-areas in the domain, including proteins, small chemical molecules and organism species. Grounding the KB learning process, by augmenting it with available domain resources such as biomedical ontologies, can potentially improve the learned structure. In order to estimate this potential, in this work, we investigate an alignment of relations emerging from biomedical ontologies with those learned from a corpus using KB-LDA.

We propose a method for extracting entity-to-entity relations from a corpus in which entities were annotated using state-of-the-art named-entity recognition systems, which integrate information from multiple known biomedical ontologies. Annotated entities in the corpus are grounded to specific entries in the source ontologies. We use KB-LDA to learn

a KB over the same corpus, regardless of the given annotations, meaning that the learning process is *not* grounded. We then align relations found using the two methods, and we consider the overlap of the two systems as a hint at the potential of grounding the KB learning process, by combining corpus-based information with information from known ontologies.

Future Directions [Chapter 7]

The key idea behind the work presented in this thesis is that grounding, the process of linking an individual word token to the real-world entity it represents, has the potential to advance statistical modeling approaches for knowledge base construction, especially in the realm of scientific domains, where high-quality grounding data is available. This idea extends vector-based document modeling, a standard in NLP today, in that vector statistics may be drawn not only directly from text corpora but also extended by statistics from more material resources. In chapter 7 we discuss lessons that were learned from this work, including the parallels between generation text and code, the challenges of knowledge base evaluation and the potential of automating this process, and different opportunities for grounding. Finally we explore possible extensions of the presented work, including an improvement to language modeling for scientific domains using induced ontologies, and extensions to the KB-LDA topic model framework that enable grounding KB construction learning, or augmenting it with small amounts of labeled data, leading to a semi-supervised framework.

Chapter 2

Statistical Language Modeling for a Software Domain Application

Statistical language models have successfully been used to describe and analyze natural language documents. Recent work applying language models to programming languages is focused on the task of predicting code, while mainly ignoring the prediction of programmer comments. In this work, we predict comments from JAVA source files of open source projects, using topic models and n-grams, and we analyze the performance of the models given varying amounts of background data on the project being predicted. We evaluate models on their comment-completion capability in a setting similar to code-completion tools built into standard code editors, and show that using a comment completion tool can save up to 47% of the comment typing.

2.1 Introduction and Related Work

Statistical language models have traditionally been used to describe and analyze natural language documents. Recently, software engineering researchers have adopted the use of language models for modeling software code. Hindle et al. [2012] observe that, as code is created by humans it is likely to be repetitive and predictable, similar to natural language. NLP models have thus been used for a variety of software development tasks such as code token completion [Han et al., 2009, Jacob and Tairas, 2010], analysis of names in code [Lawrie et al., 2006, Binkley et al., 2011] and mining software repositories [Gabel and Su, 2008].

An important part of software programming and maintenance lies in documentation,

which may come in the form of tutorials describing the code, or inline comments provided by the programmer. The documentation provides a high level description of the task performed by the code, and may include examples of use-cases for specific code segments or identifiers such as classes, methods and variables. Well documented code is easier to read and maintain in the long-run but writing comments is a laborious task that is often overlooked or at least postponed by many programmers.

Code commenting not only provides a summarization of the conceptual idea behind the code [Sridhara et al., 2010], but can also be viewed as a form of document expansion where the comment contains significant terms relevant to the described code. Accurately predicted comment words can therefore be used for a variety of linguistic uses including improved search over code bases using natural language queries, code categorization, and locating parts of the code that are relevant to a specific topic or idea [Tseng and Juang, 2003, Wan et al., 2007, Kumar and Carterette, 2013, Shepherd et al., 2007, Rastkar et al., 2011]. A related and well studied NLP task is that of predicting natural language caption and commentary for images and videos [Blei and Jordan, 2003, Feng and Lapata, 2010, 2013, Wu and Li, 2011].

In this work, our goal is to apply statistical language models for predicting class comments. We show that n-gram models are extremely successful in this task, and can lead to a saving of up to 47% in comment typing. This is expected as n-grams have been shown as a strong model for language and speech prediction that is hard to improve upon [Rosenfeld, 2000]. In some cases however, for example in a document expansion task, we wish to extract important terms relevant to the code regardless of local syntactic dependencies. We hence also evaluate the use of LDA [Blei et al., 2003] and link-LDA [Erosheva et al., 2004] topic models, which are more relevant for the term extraction scenario. We find that the topic model performance can be improved by distinguishing *code* and *text* tokens in the code.

2.2 Method

2.2.1 Models

We model a training code repository with three models that have increasing levels of language and domain understanding. First, we train n-gram models, which have a shallow statistical understanding of language, and yet have been found to be strong language predictors [Rosenfeld, 2000], making them suitable for our task. We train n-gram models ($n = 1, 2, 3$) over source code documents containing sequences of combined code and text

tokens from multiple training datasets (described below). We use the Berkeley Language Model package [Pauls and Klein, 2011] with absolute discounting (Kneser-Ney smoothing; Kneser and Ney [1995]) which includes a backoff strategy to lower-order n-grams.

Next, we model documents from the repository with LDA topic models [Blei et al., 2003]. LDA models documents as bags-of-words, taking into account word co-occurrence in documents. This model has a greater amount of understanding of language semantics than the n-gram model. We use LDA topic models trained on the same data, with 1, 5, 10 and 20 topics. The joint distribution of a topic mixture θ, and a set of K topics z, for a single source code document with N observed word tokens, $d = \{w_i\}_{i=1}^{N}$, given the Dirichlet parameters α and β, is therefore

$$p(\theta, z, d|\alpha, \beta) = \text{Dir}(\theta|\alpha) \prod_{w_i \in d} p(z|\theta)p(w_i|z, \beta) \tag{2.1}$$

where $\text{Dir}(\theta|\alpha)$ is the topic distribution of document d (sampled from a Dirichlet distribution with parameter α), $p(z|\theta)$ is the topic of word w_i (sampled from θ), and $p(w_i|z, \beta)$ is the probability of w_i in the sampled topic z. For further reading on the LDA model we refer the reader to [Blei et al., 2003]. Under the models described so far, there is no distinction between text and code tokens.

Finally, we consider an extension of the LDA model, named link-LDA [Erosheva et al., 2004], which allows us to encode a basic amount of domain understanding into the model. Here, we consider documents as having a mixed membership of two entity types, *code* and *text* tokens, $d = (\{w_i^{code}\}_{i=1}^{C_n}, \{w_i^{text}\}_{i=1}^{T_n})$, where the *text* words are tokens from comment and string literals, and the *code* words include the programming language syntax tokens (*e.g.*, `public`, `private`, `for`, etc') and all identifiers. Documents with multiple entity types can be modeled with link-LDA, and so in this case, we train link-LDA models with 1, 5, 10 and 20 topics. Under the link-LDA model, the mixed-membership joint distribution of a topic mixture, words and topics is then

$$\begin{aligned} p(\theta, z, d|\alpha, \beta) = p(\theta|\alpha) \cdot & \prod_{w_i^{text} \in d} p(z^{text}|\theta)p(w_i^{text}|z^{text}, \beta) \cdot \\ & \prod_{w_i^{code} \in d} p(z^{code}|\theta)p(w_i^{code}|z^{code}, \beta) \end{aligned} \tag{2.2}$$

where θ is the joint topic distribution, d is the set of observed document words, z^{text} is a topic associated with a text word, and z^{code} a topic associated with a code word. The link-LDA model allows us to encode a clear distinction between tokens originating from the code or text by learning distinct topics for each entity type. So while in the LDA

model, topic statistics are learned from all tokens, when using link-LDA, text topics are learned only based on tokens that appear in text, and only the joint document distribution is affected by the co-occurrence of text and code tokens in documents. Of the three categories of models we consider here, the link-LDA model has the greatest amount of language and domain understanding.

The LDA and link-LDA models use Gibbs sampling [Griffiths and Steyvers, 2004] for topic inference, based on the implementation of Balasubramanyan and Cohen [2011] with single or multiple entities per document, respectively.

2.2.2 Testing Methodology

Our goal is to predict the tokens of the JAVA class comment (the one preceding the class definition) in each of the test files. Each of the models described above assigns a probability to the next comment token. In the case of n-grams, the probability of a token word w_i is given by considering previous words $p(w_i|w_{i-1}, \ldots, w_0)$. This probability is estimated given the previous $n-1$ tokens as $p(w_i|w_{i-1}, \ldots, w_{i-(n-1)})$.

For the topic models, we separate the document tokens into the class definition and the comment we wish to predict. The set of tokens of the class comment w^c, are all considered as text tokens. The rest of the tokens in the document w^r, are considered to be the class definition, and they may contain both code and text tokens (from string literals and other comments in the source file). We then compute the posterior probability of document topics by solving the following inference problem conditioned on the w^r tokens

$$p(\theta, z^r|w^r, \alpha, \beta) = \frac{p(\theta, z^r, w^r|\alpha, \beta)}{p(w^r|\alpha, \beta)} \tag{2.3}$$

This gives us an estimate of the document distribution, θ, with which we infer the probability of the comment tokens as

$$p(w^c|\theta, \beta) = \sum_z p(w^c|z, \beta)p(z|\theta) \tag{2.4}$$

Following Blei et al. [2003], for the case of a single entity LDA, the inference problem from equation (2.3) can be solved by considering $p(\theta, z, w|\alpha, \beta)$, as in equation (2.1), and by taking the marginal distribution of the document tokens as a continuous mixture distribution for the set $w = w^r$, by integrating over θ and summing over the set of topics z

$$p(w|\alpha, \beta) = \int p(\theta|\alpha) \cdot \left(\prod_w \sum_z p(z|\theta)p(w|z, \beta) \right) d\theta \tag{2.5}$$

For the case of link-LDA where the document is comprised of two entities, in our case *code* tokens and *text* tokens, we can consider the mixed-membership joint distribution θ, as in equation (2.2), and similarly the marginal distribution $p(w|\alpha, \beta)$ over both code and text tokens from w^r. Since comment words in w^c are all considered as text tokens they are sampled using *text* topics, namely z^{text}, in equation (2.4).

2.3 Experimental Settings

2.3.1 Data and Training Methodology

We use source code from nine open source JAVA projects: Ant, Cassandra, Log4j, Maven, MinorThird, Batik, Lucene, Xalan and Xerces. For each project, we divide the source files into a training and testing dataset. Then, for each project in turn, we consider the following three main training scenarios, leading to using three training datasets.

To emulate a scenario in which we are predicting comments in the middle of project development, we can use data (documented code) from the same project. In this case, we use the in-project training dataset (*IN*). Alternatively, if we train a comment prediction model at the beginning of the development, we need to use source files from other, possibly related projects. To analyze this scenario, for each of the projects above we train models using an out-of-project dataset (*OUT*) containing data from the other eight projects.

Typically, source code files contain a greater amount of code versus comment text. Since we are interested in predicting comments, we consider a third training data source which contains more English text as well as some code segments. We use data from the popular Q&A website StackOverflow (*SO*) where users ask and answer technical questions about software development, tools, algorithms, etc'. We downloaded a dataset of all actions performed on the site since it was launched in August 2008 until August 2012. The data includes 3,453,742 questions and 6,858,133 answers posted by 1,295,620 users. We used only posts that are tagged as JAVA related questions and answers.

All the models for each project are then tested on the testing set of that project. We report results averaged over all projects in Table 2.1. Figure 2.1 shows the full results for each tested project.

Source files were tokenized using the Eclipse JDT compiler tools, separating code tokens and identifiers. Identifier names (of classes, methods and variables), were further tokenized by camel case notation (e.g., 'minMargin' was converted to 'min margin'). Non alpha-numeric tokens (e.g., dot, semicolon) were discarded from the code, as well as nu-

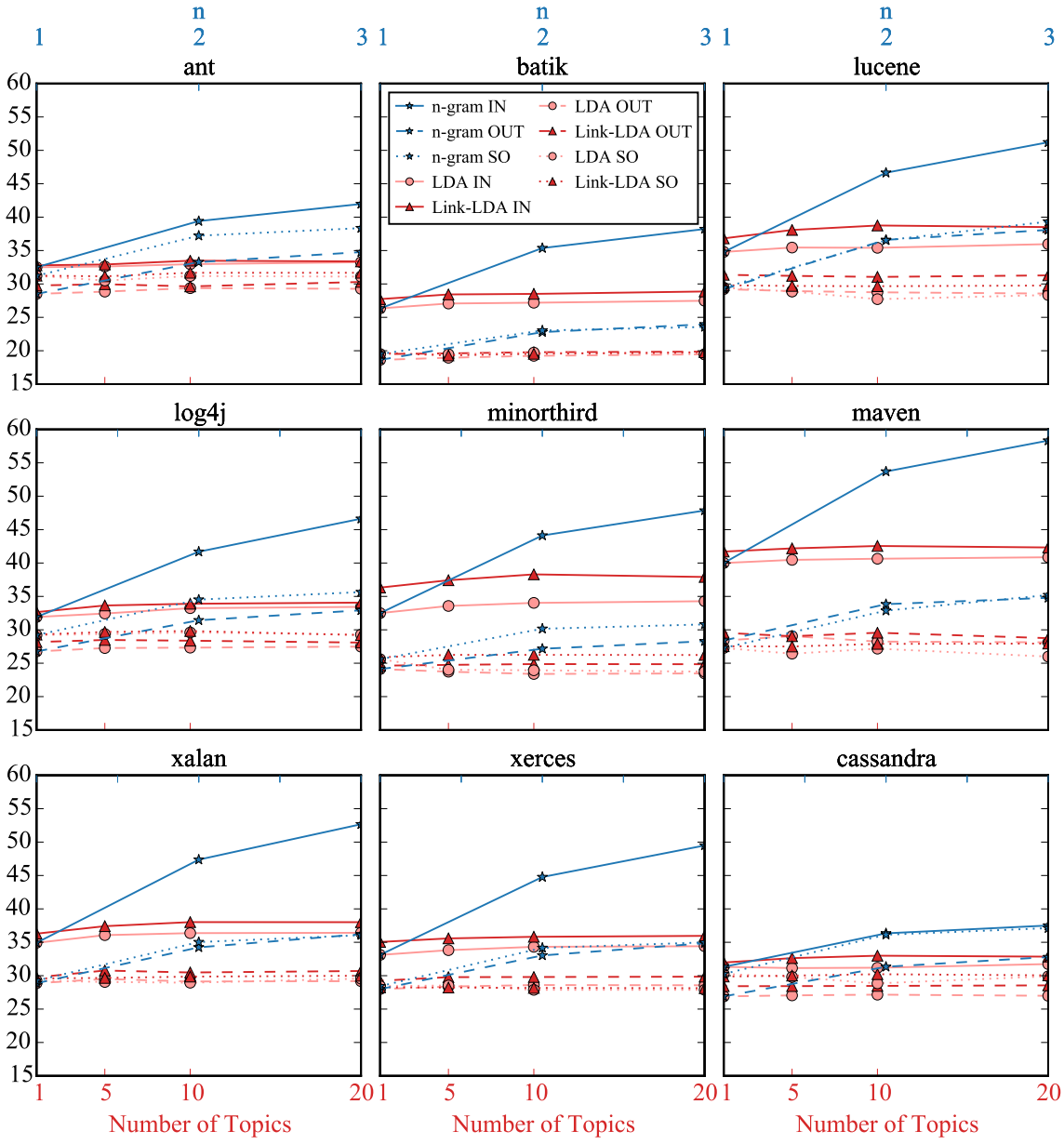


Figure 2.1: Results per project, on the average percentage of characters saved per comment, using n-gram (blue), LDA (light red) and Link-LDA (red) models trained on three training sets: IN (solid line), OUT (dashed line), and SO (dotted line). Top axis mark {1,2,3}-grams, and bottom axis mark the number of topics used for LDA and Link-LDA models. The results are also summarized in Table 2.1.

Model	n-gram			LDA				Link-LDA			
n / topics	1	2	3	20	10	5	1	20	10	5	1
IN	33.05 (3.62)	43.27 (5.79)	47.1 (6.87)	34.20 (3.63)	33.93 (3.67)	33.63 (3.67)	33.05 (3.62)	35.76 (3.95)	35.81 (4.12)	35.37 (3.98)	34.59 (3.92)
OUT	26.6 (3.37)	31.52 (4.17)	32.96 (4.33)	26.79 (3.26)	26.8 (3.36)	26.86 (3.44)	26.6 (3.37)	28.03 (3.60)	28 (3.56)	28 (3.67)	27.82 (3.62)
SO	27.8 (3.51)	33.29 (4.40)	34.56 (4.78)	27.25 (3.67)	27.22 (3.44)	27.34 (3.55)	27.8 (3.51)	28.08 (3.48)	28.12 (3.58)	27.94 (3.56)	27.9 (3.45)

Table 2.1: Average percentage of characters saved per comment using n-gram, LDA and Link-LDA models trained on three training sets: *IN*, *OUT*, and *SO*. The results are averaged over nine JAVA projects (with standard deviations in parenthesis).

Model	Predicted Comment
IN trigram	Train a named-entity extractor
IN link-LDA	Train a named-entity extractor
OUT trigram	Train a named-entity extractor
SO trigram	Train a named-entity extractor

Table 2.2: Sample comment from the *Minor-Third* project predicted using *IN*, *OUT* and *SO* based models. Saved characters are underlined and green. More prediction examples can be found in Table 2.4.

meric and single character literals. Text from comments or any string literals within the code were further tokenized with the Mallet statistical natural language processing package [McCallum, 2002]. Posts from SO were parsed using the Apache Tika toolkit[1] and then tokenized with the Mallet package. We considered as raw code tokens anything labeled using a <*code*> markup (as indicated by the SO users who wrote the post).

2.3.2 Evaluation

Since our models are trained using various data sources, the vocabularies used by each of them are different, making the comment likelihood given by each model incomparable due

[1] http://tika.apache.org/

to different sets of out-of-vocabulary tokens. We thus evaluate models using a character saving metric which aims at quantifying the percentage of characters that can be saved by using the model in a word-completion settings, similar to standard code completion tools built into code editors. For a comment word with n characters, $w = w_1, \ldots, w_n$, we predict the two most likely words given each model filtered by the first $0, \ldots, n$ characters of w. Let k be the minimal k_i for which w is in the top two predicted word tokens where tokens are filtered by the first k_i characters. Then, the number of saved characters for w is $n - k$.

The evaluation metric we use here is similar to metrics commonly used to evaluate code completion in the Software Engineering literature: Raychev et al. [2014] evaluate API method call completion and report the ratio of cases for which the desired completion appears in the top 3 results; Nguyen et al. [2012] measure the *usefulness* of a code completion task by evaluating the minimization of *developer effort* through the ratio of written versus predicted code; and similarly, Han et al. [2009] report the amount of reduction in keystrokes following a successful completion. Additional metrics used in the literature for evaluating code completion include, user studies [Bruch et al., 2009, Omar et al., 2012], and development of precision, recall, and F1 measures which evaluate code completion as a recommender system [Nguyen et al., 2012]. Robbes and Lanza [2010] discuss the biases introduced by user studies for completion tasks, in particular in the case of comparing multiple models, and motivate the use of an evaluation using a "gold-standard" benchmark, as we have used here.

In Table 2.1 we report the average percentage of saved characters per comment using each of the above models. These results are averaged over the nine input projects. Figure 2.1 shows the detailed results for each tested project. As an example, in the predicted comment shown in Table 2.2, taken from the project *Minor-Third*, the token *entity* is the most likely token according to the model *SO trigram*, out of tokens starting with the prefix 'en'. The saved characters in this case are 'tity'.

2.4 Results

Table 2.1 displays the average percentage of characters saved per class comment using each of the models. Models trained on in-project data (*IN*) perform significantly better than those trained on another data source, regardless of the model type, with an average saving of 47.1% characters using a trigram model. This is expected, as files from the same project are likely to contain similar comments, and identifier names that appear in the comment of one class may appear in the code of another class in the same project. Clearly,

Dataset	n-gram	link-LDA
IN	2778.35	574.34
OUT	1865.67	670.34
SO	1898.43	638.55

Table 2.3: Average words per project for which each tested model completes the word better than the other. This indicates that each of the models is better at predicting a different set of comment words.

in-project data should be used when available as it improves comment prediction leading to an average increase of between 6% for the worst model (26.6 for *OUT* unigram versus 33.05 for *IN*) and 14% for the best (32.96 for *OUT* trigram versus 47.1 for *IN*).

Of the out-of-project data sources, models using a greater amount of text (*SO*) mostly out-performed models based on more code (*OUT*). This increase in performance, however, comes at a cost of greater run-time due to the larger word dictionary associated with the *SO* data. Note that in the scope of this work we did not investigate the contribution of each of the background projects used in *OUT*, and how their relevance to the target prediction project effects their performance.

The trigram model shows the best performance across all training data sources (47% for *IN*, 32% for *OUT* and 34% for *SO*). Amongst the tested topic models, link-LDA models which distinguish *code* and *text* tokens perform consistently better than simple LDA models in which all tokens are considered as text. We did not however find a correlation between the number of latent topics learned by a topic model and its performance. In fact, for each of the data sources, a different number of topics gave the optimal character saving results.

Note that in this work, all topic models are based on unigram tokens, therefore their results are most comparable with that of the unigram in Table 2.1, which does not benefit from the backoff strategy used by the bigram and trigram models. By this comparison, the link-LDA topic model proves more successful in the comment prediction task than the simpler models which do not distinguish *code* and *text* tokens. Using n-grams without backoff leads to results significantly worse than any of the presented models (not shown).

Table 2.2 shows a sample comment segment for which words were predicted using trigram models from all training sources and an in-project link-LDA. The comment is taken from the *TrainExtractor* class in the *Minor-Third* project, a machine learning library for annotating and categorizing text. Both *IN* models show a clear advantage in completing

the project-specific word *Train*, compared to models based on out-of-project data (*OUT* and *SO*). Interestingly, in this example the trigram is better at completing the term *named-entity* given the prefix *named*. However, the topic model is better at completing the word *extractor* which refers to the target class. This example indicates that each model type may be more successful in predicting different comment words, and that combining multiple models may be advantageous. This can also be seen by the analysis in Table 2.3 where we compare the average number of words completed better by either the best n-gram or topic model given each training dataset. Again, while n-grams generally complete more words better, a considerable portion of the words is better completed using a topic model, further motivating a hybrid solution.

In Table 2.4 we include additional examples of comment prediction for classes in the Lucene project, taken from the *IN* dataset. We display prediction results using the most successful models in each of the modeling categories for this project: a trigram model, and LDA and link-LDA models trained with 10 topics. As indicated by the results in Table 2.1, the trigram model is the best predictor of the three and predicts the greatest amount of comment characters. However, additional observations can be drawn from a qualitative examination of the predictions.

The link-LDA model distinguishes comment and code tokens, and predicts comment text based only on text topics (z^{text}). This leads to two complementary behaviors that can be observed in the predictions: (1) link-LDA is more likely to predict "textual" tokens, and (2) LDA is more likely to predict tokens that are found in the code. As an example, LDA is better at predicting Java keywords (e.g., *exception*, *extends*, *implements*) and low-level data types (e.g., *int*, *boolean*), whereas link-LDA consistently performs better at predicting words that are more likely to appear only in text, such as *performance*, *stemming*, *characters*, *segmentation*, etc'. Similarly, we see that LDA is more likely predict the keyword *throws*, whereas link-LDA will better predict conjugations of the same verb that are not keywords in the language, including *thrown* or *throwing*. For the same reason, Link-LDA is also better at predicting javadoc description tags such as @*see*, @*link*, and @*since*.

Following the same logic, we expected class names to be consistently better predicted by the LDA model, which makes predictions based on code tokens as well as text; however, we find many contradicting examples to this. In the examples presented here, *ScoredDocIdCollector* is best predicted by LDA, *PhraseQuery* by link-LDA, and *StandardTokenizer* by the trigram model. This can be the result of class names that appear in the training comments more often than others, which can lead to a better prediction by the link-LDA or trigram models.

Both topic models predict the most probable token, of the most probable topic, at any position in the document. Since only one token can be the most probable at any position, it

Trigram	LDA	Link-LDA
Test ScoredDocIdCollector	Test ScoredDocIdCollector	Test ScoredDocIdCollector
@since lucene 1.4 @see Sort	@since lucene 1.4 @see Sort	@since lucene 1.4 @see Sort
This class converts alphabetic, numeric, and symbolic Unicode characters ... ASCII characters (the "Basic Latin" Unicode block)	This class converts alphabetic, numeric, and symbolic Unicode characters ... ASCII characters (the "Basic Latin" Unicode block)	This class converts alphabetic, numeric, and symbolic Unicode characters ... ASCII characters (the "Basic Latin" Unicode block)
As of 3.1, StandardTokenizer implements Unicode text segmentation, and StopFilter correctly handles Unicode 4.0 supplementary characters in stopwords	As of 3.1, StandardTokenizer implements Unicode text segmentation, and StopFilter correctly handles Unicode 4.0 supplementary characters in stopwords	As of 3.1, StandardTokenizer implements Unicode text segmentation, and StopFilter correctly handles Unicode 4.0 supplementary characters in stopwords
MergePolicy that makes random decisions for testing	MergePolicy that makes random decisions for testing	MergePolicy that makes random decisions for testing
MultiPhraseQuery is a generalized version of PhraseQuery, with an added method {@link #add(Term[])}.	MultiPhraseQuery is a generalized version of PhraseQuery, with an added method {@link #add(Term[])}.	MultiPhraseQuery is a generalized version of PhraseQuery, with an added method {@link #add(Term[])}.
... Stemming filters for instance can use this attribute to conditionally skip a term if {@link #isKeyword()} returns true	... Stemming filters for instance can use this attribute to conditionally skip a term if {@link #isKeyword()} returns true	... Stemming filters for instance can use this attribute to conditionally skip a term if {@link #isKeyword()} returns true
For performance reasons ... an exception will be thrown	For performance reasons ... an exception will be thrown	For performance reasons ... an exception will be thrown

Table 2.4: Examples of predicted characters (in green and underlined) of classes from the Lucene project, taken from the *IN* dataset. Each line contains a comment fragment, where the highlighted characters have been predicted using either a trigram, an LDA, or a link-LDA model.

is therefore highly unlikely for either topic model to predict a token in full (before seeing any typed characters). In contrast, the n-gram model predicts words based on the sentence history, which makes a full token prediction much more likely, and this can clearly be seen

in the results.

2.5 Implementation and Corpus

We implemented an Eclipse plugin for comment completion, based on the results of this work. The plugin enables word completion within comments, based on a 3-gram model that was trained on the full corpus of 9 open source JAVA projects used here. Word completion using this plugin works in a similar way to code completion tools built into standard code editors. While writing a comment, the user is prompted for suggestions based on the implementation of the class she/he are currently commenting. The plugin update site is: http://www.cs.cmu.edu/~dmovshov/software/commentCompletionPlugin/

We also release the training corpus, including source files for all projects and tokenizations of the code and comments, on GitHub: https://github.com/habeascorpus/habeascorpus-data-withComments

2.6 Conclusions

We analyze the use of language models for predicting class comments for source file documents containing a mixture of *code* and *text* tokens, originating from code comments and string literals. Our experiments demonstrate the effectiveness of using language models for comment completion, showing savings of up to 47% of comment characters. When available, using in-project training data proves significantly more successful than using out-of-project data. However, we find that when using out-of-project data, a dataset based on more words than code performs consistently better. Un-intuitively, we discovered that the n-gram model, which has the lowest level of domain and language understanding out of the models we tested, performed best on this task. However, the results indicate that different models are better at predicting different comment words, which motivates a hybrid solution combining the advantages of multiple models. Intuitively, we discovered that encoding some amount of domain understanding into a model, does improve the prediction performance, as can be seen by the fact that the link-LDA model outperforms the basic LDA predictor. We hypothesize that gaining a deeper semantic and categorical understanding of entities in the software domain can lead to an even greater improvement in this prediction task.

Chapter 3

Bootstrap Knowledge Base Learning for the Biomedical Domain

In Chapter 2 we investigated how well-studied statistical language models can be manipulated to improve a software domain application. Our results indicate that, intuitively, encoding our understanding of the domain into a model can improve results; indeed, making a distinction between code and non-code related tokens improved the performance of a topic model on the task of predicting code comments. However, the best performing model in this task had only a shallow statistical representation of text. We hypothesize that gaining a better understanding of the entities in a domain could improve prediction ability in domain-specific tasks. In this work, therefore, we aim at constructing a deeper representation of domain-specific entities and knowledge found in the text, by constructing a KB for a specific domain. We turn to the biomedical domain, where many ontologies are publicly available, which we utilize for the purpose of KB construction and population.

We describe an open information extraction system for biomedical text based on NELL (the Never-Ending Language Learner) [Carlson et al., 2010], a system designed for extraction from Web text. NELL uses a coupled semi-supervised bootstrapping approach to learn new facts from text, given an initial ontology and a small number of "seeds" for each ontology category. In contrast to previous applications of NELL, in our task the initial ontology and seeds are automatically derived from the existing biomedical resources. We show that NELL's bootstrapping algorithm is susceptible to ambiguous seeds, which are frequent in the biomedical domain. Using NELL to extract facts from biomedical text quickly leads to semantic drift. To address this problem, we introduce a method for assessing seed quality, based on a larger corpus of data derived from the Web. In our method, seed quality is assessed at each iteration of the bootstrapping process. Experimental results show signif-

icant improvements over NELL's original bootstrapping algorithm on two types of tasks: learning terms from biomedical categories, and named-entity recognition for biomedical entities using a learned lexicon.

3.1 Introduction

NELL (the Never-Ending Language Learner) is a semi-supervised learning system, designed for extraction of information from the Web. The system uses a coupled semi-supervised bootstrapping approach to learn new facts from text, given an initial ontology and a small number of "seeds", i.e., labeled examples for each ontology category. The new facts are stored in a growing structured knowledge base.

One of the concerns about gathering data from the Web is that it comes from various un-authoritative sources, and may not be reliable. This is especially true when gathering scientific information. In contrast to Web data, scientific text is potentially more reliable, as it is guided by the peer-review process. Open access scientific archives make this information available for all. In fact, the production rate of publicly available scientific data far exceeds the ability of researchers to "manually" process it, and there is a growing need for the automation of this process.

The biomedical field presents a great potential for text mining applications. An integral part of life science research involves production and publication of large collections of data by curators, and as part of collaborative community effort. Prominent examples include: publication of genomic sequence data, e.g., by the Human Genome Project; online collections of three-dimensional coordinates of protein structures; and databases holding data on genes. An important resource, initiated as a means of enforcing data standardization, are ontologies describing biological, chemical and medical terms. These are heavily used by the research community. With this wealth of available data the biomedical field holds many information extraction opportunities.

We describe an open information extraction system adapting NELL to the biomedical domain. We present an implementation of our approach, named *BioNELL*, which uses three main sources of information: (1) a public corpus of biomedical scientific text, (2) commonly used biomedical ontologies, and (3) a corpus of Web documents.

NELL's ontology, including categories and seeds, has been manually designed during the system development. Ontology design involves assembling a set of interesting categories, organized in a meaningful hierarchical structure, and providing representative seeds for each category. Redesigning a new ontology for a technical domain is difficult

High PMI Seeds			Random Seeds		
SoxN	achaete	cycA	cac	section 33	28
Pax-6	Drosomycin	Zfh-1	crybaby	hv	Bob
BX-C	Ultrabithorax	GATAe	ael	LRS	dip
D-Fos	sine oculis	FMRFa	chm	sht	3520
Abd-A	dCtBP	Antp	M-2	AGI	tou
PKAc	huckebein	abd-A	shanti	disp	zen
Hmgcr	Goosecoid	knirps	Buffy	Gap	Scm
fkh	decapentaplegic	Sxl	lac	Mercurio	REPO
abdA	naked cuticle	BR-C	subcosta	mef	Ferritin
zfh-1	Kruppel	hmgcr	Slam	dad	dTCF
tkv	gypsy insulator	Dichaete	Cbs	Helicase	mago
CrebA	alpha-Adaptin	Abd-B	Sufu	ora	Pten
D-raf	doublesex	gusA	pelo	vu	sb
MtnA	FasII	AbdA	sombre	domain II	TrpRS
Dcr-2	GAGA factor	dTCF	TAS	CCK	ripcord
fushi tarazu	kanamycin resistance	Ecdysone receptor	GABAA receptor	diazepam binding inhibitor	yolk protein
Tkv	dCBP		Debcl	arm	

Table 3.1: Two samples of fruit-fly genes, taken from the complete fly gene dictionary. *High PMI Seeds* are the top 50 terms selected using PMI ranking, and *Random Seeds* are a random draw of 50 terms from the dictionary. These are used as seeds for the *Fly Gene* category (Section 3.4.2). Notice that the random set contains many terms that are often not used as genes including *arm*, *28*, and *dad*. Using these as seeds can lead to semantic drift. In contrast, high PMI seeds exhibit much less ambiguity.

without non-trivial knowledge of the domain. We describe a process of merging source ontologies into one structure of categories with seed examples.

However, as we will show, using NELL's bootstrapping algorithm to extract facts from a biomedical corpus is susceptible to noisy and ambiguous terms. Such ambiguities are common in biomedical terminology (see examples in Table 3.1 and Figure 3.1), and some ambiguous terms are heavily used in the literature. For example, in the sentence

> "We have cloned an induced *white* mutation and characterized the insertion sequence responsible for the mutant phenotype"

A

Gene ID	Name 1	Name 2	Name 3
FBgn0000011	white	enhancer of garnet	e(g)
FBgn0002545	section 9	9	lf
FBgn0003204	raspberry	IMP dehydrogenase	ras-l
FBgn0004034	yellow	y	T6
FBgn0012326	Antp	Antennapedia	Dgua\Antp
FBgn0020493	dad	Daughters against dpp	Dad1

B

Abstract	Gene IDs
In Drosophila, MR (male recombination) second chromosomes are known to act as mutators and recombination inducers in males. The induction of visible mutations by MR is observed at only a limited number of genes, such as singed bristle (sn), raspberry eye colour (ras), yellow body colour (y) and a carmine eye colour (car) …	FBgn0003204 FBgn0004034

Figure 3.1: A sample from the BioCreative data set: (A) a list of gene identifiers (first column) as well as alternative common names and symbols used to describe each gene in the literature (second to last columns). The full data contains 7151 terms; and (B) sample abstract and two IDs of genes that have been annotated as being discussed in the text. In this example, the gene IDs *FBgn0003204* and *FBgn0004034* (can be found in the table) refer to the *raspberry* and *yellow* genes which are mentioned in the abstract. The full data contains 108 abstracts.

white is an ambiguous term referring to the name of a gene. In NELL, ambiguity is limited using coupled semi-supervised learning [Carlson et al., 2009]: if two categories in the ontology are declared mutually exclusive, instances of one category are used as negative examples for the other, and the two categories cannot share any instances. To resolve the ambiguity of *white* with mutual exclusion, we would have to include a *Color* category in the ontology, and declare it mutually exclusive with the *Gene* category. Then, instances of *Color* will not be able to refer to genes in the KB. It is hard to estimate what additional categories should be added, and building a "complete" ontology tree is practically infeasible.

NELL also includes a polysemy resolution component that acknowledges that one term, for example *white*, may refer to two distinct concepts, say a color and a gene, that map to different ontology categories, such as *Color* and *Fly Gene* [Krishnamurthy and Mitchell, 2011]. By including a *Color* category, this component can identify that *white* is both a color and a gene. The polysemy resolver performs word sense induction and synonym resolution based on relations defined between categories in the ontology, and labeled synonym examples. However, at present, BioNELL's ontology does not contain relation

definitions (it is based only on categories), so we cannot include this component in our experiments. Additionally, it is unclear how to avoid the use of polysemous terms as category seeds, and no method has been suggested for selecting seeds that are representative of a single specific category.

To address the problem of ambiguity, we introduce a method for assessing the desirability of noun phrases to be used as seeds for a specific target category. We propose ranking seeds using a Pointwise Mutual Information (PMI) -based collocation measure for a seed and a category name. Collocation is measured based on a large corpus of domain-independent data derived from the Web, accounting for uses of the seed in many different contexts.

NELL's bootstrapping algorithm uses the morphological and semantic features of seeds to propose new facts, which are added to the KB and used as seeds in the next bootstrapping iteration to learn more facts. This means that ambiguous terms may be added at any learning iteration. Since *white* really *is* a name of a gene, it is sometimes used in the same semantic context as other genes, and may be added to the KB despite not being used as an initial seed. To resolve this problem, we propose measuring seed quality in a *Rank-and-Learn* bootstrapping methodology: after every iteration, we rank all the instances that have been added to the KB by their quality as potential category seeds. Only high-ranking instances are used as seeds in the next iteration. Low-ranking instances are stored in the KB and "remembered" as true facts, but are not used for learning new information. This is in contrast to NELL's approach (and most other bootstrapping systems), in which there is no distinction between acquired facts, and facts that are used for learning.

3.2 Related Work

Biomedical Information Extraction systems have traditionally targeted recognition of few distinct biological entities, focusing mainly on genes [Morgan et al., 2004, Chang et al., 2004, Tanabe and Wilbur, 2002, Carpenter, 2004]. Few systems have been developed for fact-extraction of many biomedical predicates, and these are relatively small scale [Wattarujeekrit et al., 2004], or they account for limited sub-domains [Dolbey et al., 2006]. We suggest a more general approach, using bootstrapping to extend existing biomedical ontologies, including a wide range of sub-domains and many categories. The current implementation of BioNELL includes an ontology with over 100 categories. To the best of our knowledge, such large-scale biomedical bootstrapping has not been done before.

Sources of Ambiguity in Biomedical Terminology. It has been shown that biomedical terminology suffers from a higher level of ambiguity than what is found in ordinary

English words, with even greater ambiguity found in gene names [Chen et al., 2005, Krallinger et al., 2008] (see examples in Table 3.1 and Figure 3.1). This problem is manifested in two main forms. The first is the use of short-form names, lacking meaningful morphological structure, including abbreviations of three or less letters as well as isolated numbers. The second is ambiguous and polysemous terms used to describe names of genes, organisms, and biological systems and processes. For examples, *peanut* is used as both the name of a plant and a gene, and many gene names are often shared across species. What's more, with a limited possible number of three-English-letter abbreviations, and an estimate of around 35,000 human genes alone, newly introduced abbreviations are bound to overlap existing ones. Krallinger et al. [2008] provide an in-depth review discussing the ambiguous nature of this domain-specific terminology in greater detail.

Bootstrap Learning and Semantic Drift. Carlson et al. [2010] use coupled semi-supervised bootstrap learning in NELL to learn a large set of category classifiers with high precision. One drawback of using iterative bootstrapping is the sensitivity of this method to the set of initial seeds [Pantel et al., 2009]. An ambiguous set of seeds can lead to *semantic drift*, i.e., accumulation of erroneous terms and contexts when learning a semantic class. Strict bootstrapping environments reduce this problem by adding boundaries or limiting the learning process, including learning mutual terms and contexts [Riloff and Jones, 1999] and using mutual exclusion and negative class examples [Curran et al., 2007].

McIntosh and Curran [2009] propose a metric for measuring the semantic drift introduced by a learned term, favoring terms different than the recent m learned terms and similar to the first n, (shown for n=20 and n=100), following the assumption that semantic drift develops in late bootstrapping iterations. As we will show, for biomedical categories, semantic drift in NELL occurs within a handful of iterations (< 5), however according to the authors, using low values for n produces inadequate results. In fact, selecting effective n and m parameters may not only be a function of the data being used, but also of the specific category, and it is unclear how to automatically tune them.

Seed Set Refinement. Vyas et al. [2009] suggest a method for reducing ambiguity in seeds provided by human experts, by selecting the tightest seed clusters based on context similarity. The method is described for an order of 10 seeds, however, in an ontology containing hundreds of seeds per class, it is unclear how to estimate the correct number of clusters to choose from. Another approach, suggested by Kozareva and Hovy [2010], is using only constrained contexts where both seed and class are present in a sentence. Extending this idea, we consider a more general collocation metric, looking at entire documents including both the seed and its category.

3.3 Implementation

3.3.1 NELL's Bootstrapping System

We have implemented BioNELL based on the system design of NELL. NELL's bootstrapping algorithm is initiated with an input ontology structure of categories and seeds. Three sub-components operate to introduce new facts based on the semantic and morphological attributes of known facts. At every iteration, each component proposes candidate facts, specifying the supporting evidence for each candidate, and the candidates with the most strongly supported evidence are added to the KB. The process and sub-components are described in detail by Carlson et al. [2010] and Wang and Cohen [2009].

3.3.2 Text Corpora

PubMed Corpus: We used a corpus of 200K full-text biomedical articles taken from the PubMed Central Open Access Subset (extracted in October 2010)[1], which were processed using the OpenNLP package[2]. This is the main BioNELL corpus and it is used to extract category instances in all the experiments presented in this chapter.

Web Corpus: BioNELL's seed-quality collocation measure (Section 3.3.4) is based on a domain-independent Web corpus, the English portion of the ClueWeb09 data set [Callan et al., 2009], which includes 500 million web documents.

3.3.3 Ontology

BioNELL's ontology is composed of six base ontologies, covering a wide range of biomedical sub-domains: the Gene Ontology (GO) [Ashburner et al., 2000], describing gene attributes; the NCBI Taxonomy for model organisms [Sayers et al., 2011]; Chemical Entities of Biological Interest (ChEBI) [Degtyarenko et al., 2008], a dictionary focused on small chemical compounds; the Sequence Ontology [Eilbeck et al., 2005], describing biological sequences; the Cell Type Ontology [Bard et al., 2005]; and the Human Disease Ontology [Osborne et al., 2009]. Each ontology provides a hierarchy of terms but does not distinguish concepts from instances.

[1] http://www.ncbi.nlm.nih.gov/pmc/
[2] http://opennlp.sourceforge.net

We used an automatic process for merging base ontologies into one ontology tree. First, we group the ontologies under one hierarchical structure, producing a tree of over 1 million entities, including 856K terms and 154K synonyms. We then separate these into *potential categories* and *potential seeds*. *Categories* are nodes that are unambiguous (have a single parent in the ontology tree), with at least 100 descendants. These descendants are the category's *Potential seeds*. This results in 4188 category nodes. In the experiments of this chapter we selected only the top (most general) 20 categories in the tree of each base ontology. We are left with 109 final categories, as some base ontologies had less than 20 categories under these restrictions. Leaf categories are given seeds from their descendants in the full tree of all terms and synonyms, giving a total of around 1 million potential seeds. Seed set refinement is described below. The seeds of leaf categories are later extended by the bootstrapping process.

3.3.4 BioNELL's Bootstrapping System

PMI Collocation with the Category Name

We define a seed quality metric based on a large corpus of Web data. Let s and c be a seed and a target category, respectively. For example, we can take s = "white", the name of a gene of the fruit-fly, and c = "fly gene". Now, let D be a document corpus (Section 3.3.2 describes the Web corpus used for ranking), and let D_c be a subset of the documents containing a mention of the category name. We measure the collocation of the seed and the category by the number of times s appears in D_c, $|Occur(s, D_c)|$. The overall occurrence of s in the corpus is given by $|Occur(s, D)|$. Following the formulation of Church and Hanks [1990], we compute the PMI-rank of s and c as

$$\text{PMI}(s, c) = \frac{|Occur(s, D_c)|}{|Occur(s, D)|} \tag{3.1}$$

Since this measure is used to compare seeds of the same category, we omit the $\log$ from the original formulation. In our example, as *white* is a highly ambiguous gene name, we find that it appears in many documents that do not discuss the fruit fly, resulting in a PMI rank close to 0.

The proposed ranking is sensitive to the descriptive name given to categories. For a more robust ranking, we use a combination of rankings of the seed with several of its ancestors in the ontology hierarchy. In [Movshovitz-Attias and Cohen, 2012b] we describe this hierarchical ranking in more detail and additionally explore the use of the binomial log-likelihood ratio test (BLRT) as an alternative collocation measure for ranking.

We further note that some specialized biomedical terms follow strict nomenclature rules making them easily identifiable as category specific. These terms may not be frequent in general Web context, leading to a low PMI rank under the proposed method. Given such a set of high confidence seeds from a reliable source, one can enforce their inclusion in the learning process, and specialized seeds can additionally be identified by high-confidence patterns, if such exist. However, the scope of this work involves selecting seeds from an ambiguous source, biomedical ontologies, thus we do not include an analysis for these specialized cases.

Rank-and-Learn Bootstrapping

We incorporate PMI ranking into BioNELL using a *Rank-and-Learn* bootstrapping methodology. After every iteration, we rank all the instances that have been added to the KB. Only high-ranking instances are added to the collection of seeds that are used in the next learning iteration. Instances with low PMI rank are stored in the KB and are not used for learning new information. We consider a high-ranking instance to be one with PMI rank higher than 0.25.

3.4 Experimental Evaluation

3.4.1 Experimental Settings

Configurations of the Algorithm

In our experiments, we ran BioNELL and NELL with the following system configurations, all using the biomedical corpus and the ontology described in Sections 3.3.2 and 3.3.3, and all running 50 iterations, in order to evaluate the long term effects of ranking. Section 3.4.2 includes a discussion on the learning rate of the tested systems which motivates the reason for evaluating performance at the 50th iteration.

To expand a category we used the following systems, also summarized in Table 3.2: (1) the *BioNELL* system, using Rank-and-Learn bootstrapping (Section 3.3.4) initialized with the top 50 seeds using PMI ranking, (2) the *NELL* system, using NELL's original bootstrapping algorithm (Section 3.3.1) initialized with 50 random seeds from the category's potential seeds (NELL does not provide a seed selection method), and (3) in order to distinguish the contribution of Rank-and-Learn bootstrapping over ranking the initial seeds, we tested a third system, *BioNELL+Random*, using Rank-and-Learn bootstrapping

Learning System	BootstrappingAlgorithm	InitialSeeds	Corpus
BioNELL	Rank-and-Learn with PMI	PMI top 50	PubMed
NELL	NELL's algorithm	Random 50	PubMed
BioNELL+Random	Rank-and-Learn with PMI	Random 50	PubMed

Table 3.2: Learning systems used in our evaluation, all using the PubMed biomedical corpus and the biomedical ontology described in Sections 3.3.2 and 3.3.3.

initialized with 50 random seeds.

Evaluation Methodology

Using BioNELL we can learn *lexicons*, collections of category terms accumulated after running the system. One evaluation approach is to select a set of learned instances and assess their correctness [Carlson et al., 2010]. This is relatively easy for data extracted for general categories like City or Sports Team. For example, it is easy to evaluate the statement "London is a City". This task becomes more difficult when assessing domain-specific facts such as "Beryllium is an S-block molecular entity" (in fact, it is). We cannot, for example, use the help of Mechanical Turk for this task. A possible alternative evaluation approach is asking an expert. On top of being a costly and slow approach, the range of topics covered by BioNELL is large and a single expert is not likely be able to assess all of them.

We evaluated lexicons learned by BioNELL by comparing them to available resources. Lexicons of gene names for certain species are available, and Freebase [Google, 2011], an open repository holding data for millions of entities, includes some biomedical concepts. For most biomedical categories, however, complete lexicons are scarce.

Data Sets

We compared learned lexicons to category *dictionaries*, lists of concept terms taken from the following sources, which we consider as a Gold Standard.

We used three lexicons of biomedical categories taken from Freebase: Disease (9420 terms), Chemical Compound (9225 terms), and Drug (3896 terms).

To evaluate gene names we used data from the BioCreative Challenge [Hirschman et al., 2005], an evaluation competition focused on annotations of genes and gene products.

Learning System	Precision	Correct	Total
BioNELL	**.83**	109	132
NELL	.29	186	**651**
BioNELL+Random	.73	**248**	338
NELL $_{\text{by size 132}}$	.72	93	130

Table 3.3: Precision, total number of instances (*Total*), and correct instances (*Correct*) of gene lexicons learned with BioNELL and NELL. BioNELL significantly improves the precision of the learned lexicon compared with NELL. When examining only the first 132 learned items, BioNELL has both higher precision and more correct instances than NELL (last row, NELL $_{\text{by size 132}}$).

The data includes a dictionary of genes of the fruit-fly, *Drosophila Melanogaster*, which specifies a set of gene identifiers and possible alternative forms of the gene name, for a total of 7151 terms, which we consider to be the complete fly gene dictionary. Figure 3.1A contains a sample from the fruit-fly gene dictionary.

We used additional BioCreative data for a named-entity recognition task. This includes 108 scientific abstracts, manually annotated by BioCreative with gene IDs of fly genes discussed in the text. The abstracts contain either the gene ID or any gene name. Figure 3.1B contains an excerpt from one of the abstracts in the data and two IDs of genes that have been annotated as being mentioned in the text.

3.4.2 Extending Lexicons of Biomedical Categories

Recovering a Closed Category Lexicon

We used BioNELL to learn the lexicon of a closed category, representing genes of the fruit-fly, *D. Melanogaster*, a model organism used to study genetics and developmental biology. Two samples of genes from the full fly gene dictionary are shown in Table 3.1: *High PMI Seeds* are the top 50 dictionary terms selected using PMI ranking, and *Random Seeds* are a random draw of 50 terms. Notice that the random set contains many seeds that are not distinct gene names including *arm*, *28*, and *dad*. In contrast, high PMI seeds exhibit much less ambiguity. We learned gene lexicons using the test systems described in Section 3.4.1 with the high-PMI and random seed sets shown in Table 3.1. We measured the precision, total number of instances, and correct instances of the learned lexicons against the full dictionary of genes. Table 3.3 summarizes the results.

BioNELL, initialized with PMI-ranked seeds, significantly improved the precision of the learned lexicon over NELL (29% for *NELL* to 83% for *BioNELL*). In fact, the two learning systems using Rank-and-Learn bootstrapping resulted in higher precision lexicons (83%, 73%), suggesting that constrained bootstrapping using iterative seed ranking successfully eliminates noisy and ambiguous seeds.

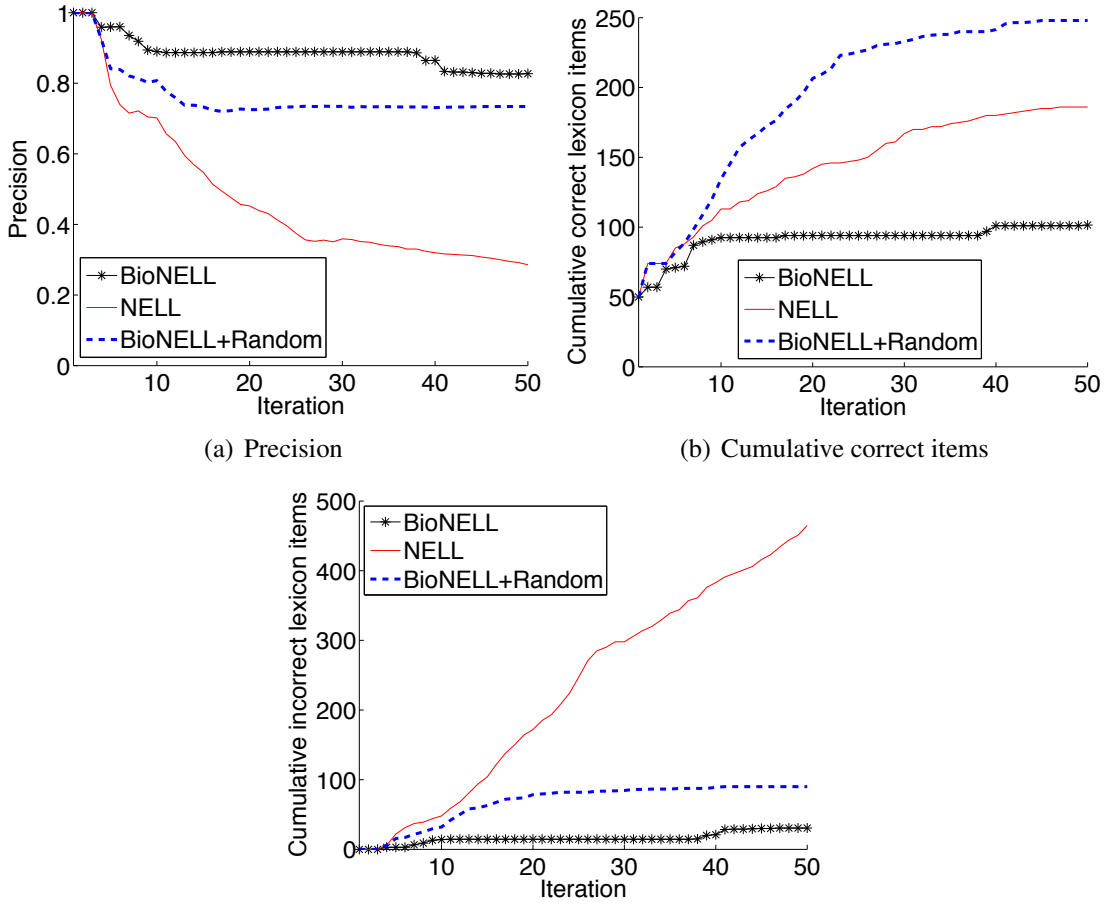
BioNELL's bootstrapping methodology is highly restrictive and it affects the size of the learned lexicon as well as its precision. Notice, however, that while *NELL*'s final lexicon is 5 times larger than *BioNELL*'s, the number of correctly learned items in it are less than twice that of *BioNELL*. Additionally, *BioNELL+Random* has learned a smaller dictionary than *NELL* (338 and 651 terms, respectively) with a greater number of correct instances (248 and 186).

We examined the performance of *NELL* after the 7th iteration, when it has learned a lexicon of 130 items, similar in size to *BioNELL*'s final lexicon (Table 3.3, last row). After learning 130 items, *BioNELL* achieved both higher precision (83% versus 72%) and higher recall (109 versus 93 correct lexicon instances) than *NELL*, indicating that *BioNELL*'s learning method is overall more accurate.

After running for 50 iterations, all systems recover only a small portion of the complete gene dictionary—between 109 and 248 instances out of 7151—suggesting either that (1) more learning iterations are required (though the rate of learning new correct instances has slowed down by the 50th iterations, it was still positive; see Figure 3.2), (2) the biomedical corpus we use is too small and does not contain (frequent) mentions of some gene names from the dictionary, or (3) some other limitations exist that prevent the learning algorithm from finding additional class examples. Some observations support the notion that certain gene names are particularly challenging to discover with NLP systems. Out of the 7151 names in the dictionary, 1363 are variations of other names with only a modification of letter case, or an addition of '.' or '-'. Of the rest, 222 are numeric-only names, and 401 are 2 letter abbreviations, both categories tending to be more ambiguous than general English words.

Lexicons learned using BioNELL show persistently high precision throughout the 50 iterations, even when initiated with random seeds (Figure 3.2A). By the final iteration, all systems stop accumulating further significant amounts of correct gene instances (Figure 3.2B). Systems that use PMI-based Rank-and-Learn bootstrapping also stop learning incorrect instances (*BioNELL* and *BioNELL+Random*; Figure 3.2C). This is in contrast to *NELL* which continues learning incorrect examples.

Interestingly, the highest number of correct gene instances was learned using Rank-and-Learn bootstrapping with random initial seeds (248 items; *BioNELL+Random*). This



(a) Precision

(b) Cumulative correct items

(c) Cumulative incorrect items

Figure 3.2: Performance per learning iteration for gene lexicons learned using BioNELL and NELL.

may happen when the random set includes genes that are infrequent in the general Web corpus, despite being otherwise category-specific in the biomedical context. As such, these would result in low PMI rank (see note in Section 3.3.4). However, random seed selection does not offer any guarantees on the quality of the seeds used, and therefore will result in unstable performance. Note that *BioNELL+Random* was initiated with the same random seeds as *NELL*, but due to the more constrained Rank-and-Learn bootstrapping it achieves both higher recall (248 versus 186 correct instances) and precision (73% versus 29%).

Learning System	Precision			Correct			Total		
	CC	Drug	Disease	CC	Drug	Disease	CC	Drug	Disease
BioNELL	**.66**	**.52**	**.43**	63	508	276	96	972	624
NELL	.15	.40	.37	**74**	**522**	**288**	**449**	**1300**	**782**
NELL $_{\text{by size}}$	.58	.47	.37	58	455	232	100	968	623

Table 3.4: Precision, total number of instances (*Total*), and correct instances (*Correct*) of learned lexicons of *Chemical Compound* (CC), *Drug*, and *Disease*. BioNELL's lexicons have higher precision on all categories compared with NELL, while learning a similar number of correct instances. When restricting NELL to a total lexicon size similar to BioNELL's, BioNELL has both higher precision and more correct instances (last row, NELL $_{\text{by size}}$).

Extending Lexicons of Open Categories

We evaluated learned lexicons for three open categories, *Chemical Compound* (*CC*), *Drug*, and *Disease*, using dictionaries from Freebase. Since these are open categories — new drugs are being developed every year, new diseases are discovered, and varied chemical compounds can be created — the Freebase dictionaries are not likely to contain complete information on these categories. For our evaluation, however, we considered them to be complete.

We used *BioNELL* and *NELL* to learn these categories, and for all of them *BioNELL*'s lexicons achieved higher precision than *NELL* (Table 3.4). The number of correct learned instances was similar in both systems (63 and 74 for *CC*, 508 and 522 for *Drug*, and 276 and 288 for *Disease*), however in *BioNELL*, the additional bootstrapping restrictions assist in rejecting incorrect instances, resulting in a smaller, more accurate lexicon.

We examined *NELL*'s lexicons when they reached a size similar to *BioNELL*'s final lexicons (at the 8th, 42nd and 39th iterations for *CC*, *Drug*, and *Disease*, respectively). *BioNELL*'s lexicons have both higher precision and higher recall (more correct learned instances) than the comparable *NELL* lexicons (Table 3.4, NELL $_{\text{by size}}$, last row).

3.4.3 Named-Entity Recognition using a Learned Lexicon

We examined the use of gene lexicons learned with BioNELL and NELL for the task of recognizing concepts in free text, using a simple strategy of matching words in the text

Lexicon	Precision	Correct	Total
BioNELL	.90	18	20
NELL	.02	5	268
BioNELL+Random	.03	3	82
Complete Dictionary	.09	153	1616
Filtered Dictionary	.18	138	675

Table 3.5: Precision, total number of predicted genes (*Total*), and correct predictions (*Correct*), in a named-entity recognition task using a complete lexicon, a filtered lexicon, and lexicons learned with BioNELL and NELL. BioNELL's lexicon achieves the highest precision, and makes more correct predictions than NELL.

with terms from the lexicon. We use data from the BioCreative challenge (Section 3.4.1), which includes text abstracts and the IDs of genes that appear in each abstract. We show that BioNELL's lexicon achieves both higher precision and recall in this task than NELL's.

We implemented an *annotator* for predicting what genes are discussed in text, which uses a gene lexicon as input. Given sample text, if any of the terms in the lexicon appear in the text, the corresponding gene is predicted to be discussed in the text. Following BioCreative's annotation format, the annotator emits as output the set of gene IDs of the genes predicted for the sample text.

We evaluated annotators that were given as input: the complete fly-genes dictionary, a filtered version of that dictionary, or lexicons learned using BioNELL and NELL. Using these annotators we predicted gene mentions for all text abstracts in the data. We report the average precision (over 108 text abstracts) and number of total and correct predictions of gene mentions, compared with the labeled annotations for each text (Table 3.5).

Many gene names are shared among multiple variants. For example, the name *Antennapedia* may refer to several gene variations, e.g., *Dgua\Antp* or *Dmed\Antp*. Thus, in our precision measurements, we consider a prediction of a gene ID as "true" if it is labeled as such by BioCreative, or if it shares a synonym name with another true labeled gene ID.

Using a Complete Dictionary

First, we used the complete fly gene dictionary for the recognition task. Any dictionary gene that is mentioned in the text was recovered, resulting in high recall. However, the full dictionary contains ambiguous gene names that contribute many false predictions to

the complete dictionary annotator, leading to a low precision of 9%.

Using a Manually-Filtered Dictionary

Some ambiguous terms can be detected using simple rules, e.g., short abbreviations and numbers. For example, *section 9* is a gene named after the functional unit to which it belongs, and abbreviated by the symbol *9*. Clearly, removing *9* from the full lexicon should improve precision without great cost to recall. We similarly filtered the full dictionary, removing one- and two-letter abbreviations and terms composed only of non-alphabetical characters, leaving 6253 terms. Using the filtered dictionary, precision has doubled (18%) with minor compromise to recall. Using complete or manually refined gene dictionaries for named-entity recognition has been shown before to produce similar high-recall and low-precision results [Bunescu et al., 2005].

Using a Learned Lexicon

We evaluated annotators on gene lexicons learned with BioNELL and NELL. *BioNELL*'s lexicon achieved significantly higher precision (90%) than other lexicons (2%-18%). It is evident that this lexicon contains few ambiguous terms as it leads to only 2 false predictions. Note also, that *BioNELL*'s lexicon has both higher precision and recall than *NELL*.

3.5 Conclusions

We have proposed a methodology for an information extraction system for biomedical scientific text, using an automatically derived ontology of categories and seeds. Our implementation is based on constrained bootstrapping in which seeds are ranked at every iteration.

We have demonstrated that, by starting with our derived ontology and adding constraints to NELL's bootstrapping method, we build a high-precision biomedical KB, including significantly less ambiguous lexicons for all the evaluated biomedical concepts. BioNELL shows 51% increase over NELL in the precision of a learned lexicon of chemical compounds, and 45% increase for a category of gene names. Importantly, when BioNELL and NELL learn lexicons of similar size, BioNELL's lexicons have both higher precision and recall.

The biomedical KB that we have built is useful in the context of an NLP task in this

domain: we have shown that BioNELL's learned gene lexicon is a high-precision annotator in an entity recognition task (with 90% precision). The results are promising, though it is currently difficult to provide a similar quantitative evaluation for a wider range of concepts.

Many interesting improvements could be made in the current system, mainly discovery of relations between existing ontology categories. In addition, we believe that Rank-and-Learn bootstrapping and iterative seed ranking can be beneficial in general, domain-independent settings, and we would like to explore further use of this method.

Chapter 4

Grounded Software Ontology Construction using Coordinate Term Relationships

In Chapter 3 we constructed a Knowledge Base for the biomedical domain, using pre-existing domain ontologies as a source of schema and seeds. We then manipulated a semi-supervised bootstrapping algorithm for KB population, in order to learn facts related to concepts introduced in the existing ontologies. The main disadvantage of the presented approach is that it relies on an input biomedical ontology; however, other technical domains, including software, are not as well-studied linguistically and there are no pre-existing ontologies which describe important concepts and terminology in these domains. This motivates unsupervised approaches for constructing ontologies based on domain corpora alone. In this work, we propose a method for discovering relations between domain entities found in text, and these relations are later combined into an ontology. The advantage of the resulting ontology is that it reflects statistics about the domain entities; this is in contrast to human-created ontologies such as the ones from the biomedical domain, or those typically used in open-domain KB population methods.

Our approach uses distant supervision to detect coordinate-term relationships between entities from the software domain that refer to Java classes. Usually, semantic relations are found by examining corpus statistics associated with text entities. In some technical domains, however, we have access to additional information about the real-world objects named by the entities, suggesting that coupling information about the "grounded" entities with corpus statistics might lead to improved methods for relation discovery. To this end, we develop a similarity measure for Java classes using distributional information about

how they are used in software, which we combine with corpus statistics on the distribution of contexts in which the classes appear in text. We propose two approaches for weakly-labeling extracted pairs of classes. We then show that using a combination of corpus and code statistics, prediction results on extracted pairs improved dramatically, from around 60% to 88%. Additionally, human labeling results show that our classifier has an F1 score of 86% over the top 1000 predicted pairs.

4.1 Introduction

Discovering semantic relations between text entities is a key task in natural language understanding. It is a critical component which enables the success of knowledge representation systems such as TextRunner [Yates et al., 2007], ReVerb [Fader et al., 2011], and NELL [Carlson et al., 2010], which in turn are useful for a variety of NLP applications, including, temporal scoping [Talukdar et al., 2012b], semantic parsing [Krishnamurthy and Mitchell, 2012] and entity linking [Lin et al., 2012].

In this work, we examine *coordinate* relations between words. According to the WordNet glossary, *X* and *Y* are defined as *coordinate terms* if they share a common hypernym [Miller, 1995, Fellbaum, 1998]. This is a symmetric relation that indicates a semantic similarity, meaning that *X* and *Y* are "a type of the same thing", since they share at least one common ancestor in some hypernym taxonomy (to paraphrase the definition of Snow et al. [Snow et al., 2006]). More broadly, here, we consider coordinate entities to be pairs of entities that exhibit some level of similarity among them.

Semantic similarity relations are normally discovered by comparing corpus statistics associated with the entities: for instance, two entities X and Y that usually appear in similar contexts are likely to be semantically similar [Pereira et al., 1993, Pantel, 2003, Curran, 2004]. However, in technical domains, we have access to additional information about the real-world objects that are named by the entities: e.g., we might have biographical data about a person entity, or a 3D structural encoding of a protein entity. In such situations, it seems plausible that a "grounded" NLP method, in which corpus statistics are coupled with data on the real-world referents of X and Y, might lead to improved methods for relation discovery.

Here we explore the idea of grounded relation discovery in the domain of software. In particular, we consider the detection of coordinate-term relationships between entities that (potentially) refer to Java classes. We use a software domain text corpus derived from the Q&A website StackOverflow (SO), in which users ask and answer questions about software development, and we extract posts which have been labeled by users as *Java* related.

From this data, we collected a small set of entity pairs that are labeled as coordinate terms (or not) based on high-precision Hearst patterns and frequency statistics, and we attempt to label these pairs using information available from higher-recall approaches based on distributional similarity.

We describe an entity linking method in order to map a given text entity to an underlying class type implementation from the Java standard libraries. Next, we describe corpus and code based information that we use for the relation discovery task. Corpus based methods include distributional similarity and string matching similarity. Additionally, we use two sources of code based information: (1) we define the *class-context* of a Java class in a given code repository, and are therefore able to calculate a code-based distributional similarity measure for classes, and (2) we consider the hierarchical organization of classes, described by the Java class type and namespace hierarchies. We demonstrate that using our approach, cross-validation accuracy on this dataset is improved from 60.9% to 88%. According to human labeling, our classifier has an F1-score of 86% over the highest-ranking 1000 predicted pairs.

We see this work as a first step towards building a knowledge representation system for the software domain, in which text entities refer to elements from a software code base, for example classes, methods, applications and programming languages. Understanding software entity relations will allow the construction of a domain specific taxonomy and knowledge base, which can enable higher reasoning capabilities in NLP applications for the software domain [Weimer et al., 2007, Wang et al., 2009, Branavan et al., 2010, Movshovitz-Attias and Cohen, 2013] and improve a variety of code assisting applications, including code refactoring and token completion [Han et al., 2009, Jacob and Tairas, 2010, Binkley et al., 2011, Schulam et al., 2013].

Figure 4.1 shows a visualization based on coordinate term pairs predicted using the proposed method. Java classes with similar functionality (highlighted in Figure 4.2) are highly connected in this graph, indicating that our method can be used to construct a code taxonomy.

4.2 Related Work

4.2.1 Semantic Relation Discovery

Previous work on semantic relation discovery, in particular, coordinate term discovery, has used two main approaches. The first is based on the insight that certain lexical patterns indicate a semantic relationship with high-precision, as initially observed by Hearst

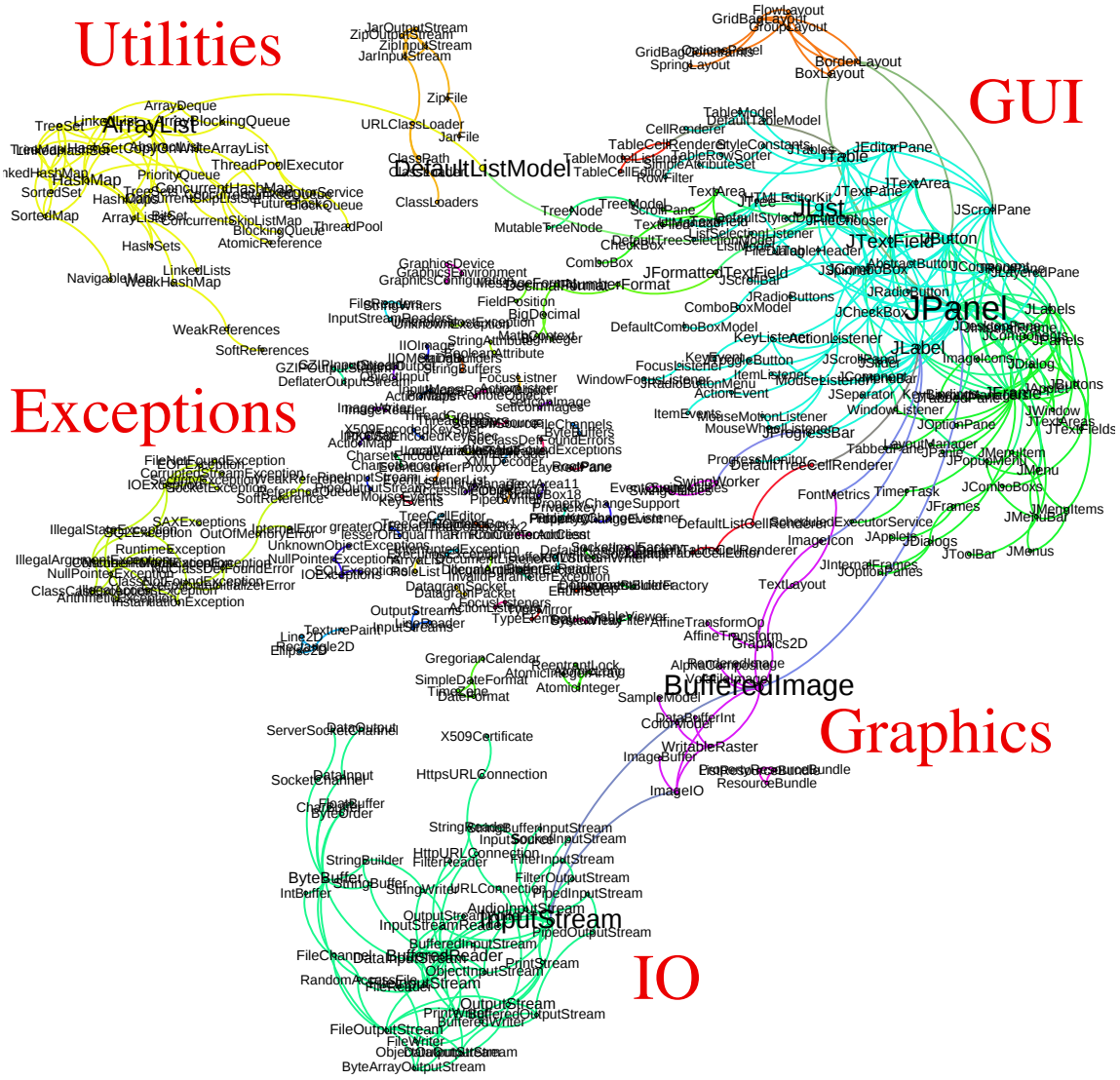
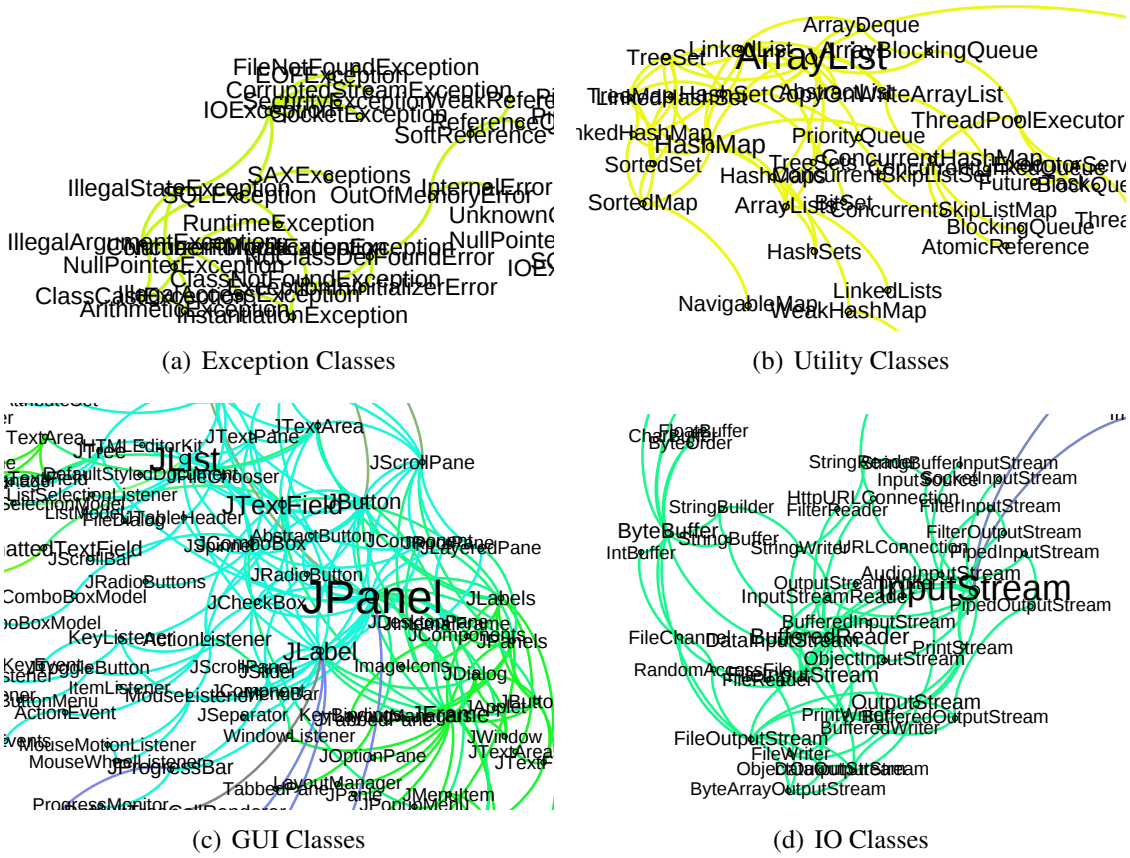


Figure 4.1: Visualization of predicted coordinate term pairs, where each pair of coordinate classes is connected by an edge. Highly connected components are labeled by edge color, and it can be noted that they contain classes with similar functionality. Some areas containing a functional class group have been labeled with the name of the class in red, and a magnified version of that area can be found in Figure 4.2 for easier readability.



(a) Exception Classes

(b) Utility Classes

(c) GUI Classes

(d) IO Classes

Figure 4.2: A magnification of several clusters from Figure 4.1. The classes in each cluster belong to a similar functional class group.

[Hearst, 1992]. For example, the conjuction pattern "X and Y" indicates that X and Y are coordinate terms. Other pattern-based classifier have been introduced for meronyms [Girju et al., 2003], synonyms [Lin et al., 2003], and general analogy relations [Turney et al., 2003]. The second approach relies on the notion that words that appear in a similar context are likely to be semantically similar. In contrast to pattern based classifiers, context distributional similarity approaches are normally higher in recall. [Pereira et al., 1993, Pantel, 2003, Curran, 2004, Snow et al., 2004]. In this work we attempt to label samples extracted with high-precision Hearst patterns, using information from higher-recall methods.

4.2.2 Grounded Language Learning

The aim of grounded language learning methods is to learn a mapping between natural language (words and sentences) and the observed world [Siskind, 1996, Yu and Ballard, 2004, Gorniak and Roy, 2007], where more recent work includes grounding language to the physical world [Krishnamurthy and Kollar, 2013], and grounding of entire discourses [Minh et al., 2013]. Early work in this field relied on supervised aligned sentence-to-meaning data [Zettlemoyer and Collins, 2005, Ge and Mooney, 2005]. However, in later work the supervision constraint has been gradually relaxed [Kate and Mooney, 2007, Liang et al., 2009]. Relative to prior work on grounded language acquisition, we use a very rich and complex representation of entities and their relationships (through software code). However, we consider a very constrained language task, namely coordinate term discovery.

4.2.3 Statistical Language Models for Software

In recent work by NLP and software engineering researchers, statistical language models have been adapted for modeling software code. NLP models have been used to enhance a variety of software development tasks such as code and comment token completion [Han et al., 2009, Jacob and Tairas, 2010, Movshovitz-Attias and Cohen, 2013, Schulam et al., 2013], analysis of code variable names [Lawrie et al., 2006, Binkley et al., 2011], and mining software repositories [Gabel and Su, 2008]. This has been complemented by work from the programming language research community for structured prediction of code syntax trees [Omar, 2013]. To the best of our knowledge, there is no prior work on discovering semantic relations for software entities.

4.3 Coordinate Term Discovery

In this section we describe a coordinate term classification pipeline, as depicted at high-level in Figure 4.3. All the following steps are described in detail in the sections below.

Given a software domain text corpus (StackOverflow) and a code repository (Java Standard Libraries), our goal is to predict a coordinate relation for $\langle X, Y \rangle$, where X and Y are nouns which potentially refer to Java classes.

We first attempt a baseline approach of labeling the pair $\langle X, Y \rangle$ based on corpus distributional similarity. Since closely related classes often exhibit morphological closeness, we use as a second baseline the string similarity of X and Y.

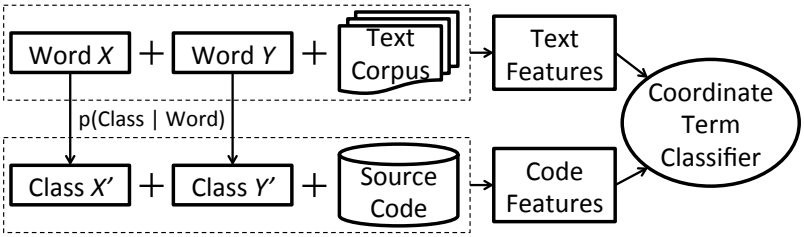


Figure 4.3: Classification Pipeline for determining whether nouns X and Y are coordinate terms. Each noun is mapped to an underlying class from the code repository with probability, $p(\text{Class}|\text{Word})$. Textual features are extracted based on the input words, code based features are extracted using the mapped classes, and all of these are given to the coordinate term classifier.

Next, we map noun X to an underlying class implementation from the code repository, named X', according to an estimated probability for $p(\text{Class } X'|\text{Word } X)$, s.t., $X' = \max_C \hat{p}(C|X)$, for all other classes C. X' is then the code referent of X. Similarly, we map Y to the class Y'. Given a code-based grounding for X and Y we extract information using the class implementations: (1) we define a code based distributional similarity measure, using code-context to encode the usage pattern of a class, and (2) we use the hierarchical organization of classes, described by the type and namespace hierarchies. Finally, we combine all the above information in a single SVM classifier.

4.3.1 Baseline: Corpus Distributional Similarity

As an initial baseline we calculate the corpus distributional similarity of nouns $\langle X, Y \rangle$, following the assumption that words with similar context are likely to be semantically similar. Our implementation follows Pereira et al. [Pereira et al., 1993]. We calculate the empirical context distribution for noun X

$$p_X = f(c, X) / \sum_{c'} f(c', X) \tag{4.1}$$

where $f(c, X)$ is the frequency of occurrence of noun X in context c, which includes a window of 3 words around X. We then measure the similarity of nouns X and Y using the *relative entropy* or *Kullback-Leibler divergence*

$$D(p_X||p_Y) = \sum_z p_X(z) \log \frac{p_X(z)}{p_Y(z)} \tag{4.2}$$

As this measure is not symmetric we finally consider the distributional similarity of X and Y as $D(p_X||p_Y) + D(p_Y||p_X)$.

4.3.2 Baseline: String Similarity

Due to naming convention standards, many related classes often exhibit some morphological closeness. For example, classes that provide Input/Output access to the file system will often contain the suffix `Stream` or `Buffer`. Likewise, many classes extend on the names of their super classes (e.g., `JRadioButtonMenuItem` extends the class `JMenuItem`). More examples can be found in Figure 4.1 and Table 4.4. We therefore include a second baseline which attempts to label the noun pair $\langle X, Y \rangle$ as coordinate terms according to their string matching similarity. We use the SecondString open source Java toolkit[1]. Each string is tokenized by camel case (such that *ArrayList* is represented as *Array List*). We consider the SoftTFIDF distance of the tokenized strings, as defined by Cohen et al. [Cohen et al., 2003].

4.3.3 Entity Linking

In order to draw code based information on text entities, we define a mapping function between words and class types. Our goal is to find $p(C|W)$, where C is a specific class implementation and W is a word. This mapping is ambiguous, for example, since users are less likely to mention the qualified class name (e.g., `java.lang.String`), and usually use the class *label*, meaning the name of the class not including its package (e.g., `String`). As an example, the terms `java.lang.String` and `java.util.Vector` appears 37 and 1 times respectively in our corpus, versus the terms `String` and `Vector` which appear 35K and 1.6K times. Additionally, class names appear with several variations, including, case-insensitive versions, spelling mistakes, or informal names (e.g., *array* instead of *ArrayList*).

Therefore, in order to approximate $p(C, W)$ in

$$p(C|W) = \frac{p(C, W)}{p(W)} \tag{4.3}$$

We estimate a word to class-type mapping that is mediated through the class label, L, as

$$\hat{p}(C, W) = p(C, L) \cdot p(L, W) \tag{4.4}$$

[1] http://secondstring.sourceforge.net/

Since $p(C, L) = p(C|L)p(L)$, this can be estimated by the corresponding MLEs

$$\hat{p}(C, L) = \hat{p}(C|L) \cdot \hat{p}(L) = \frac{f(C)}{\sum_{C' \in L} f(C')} \cdot \frac{f(L)}{\sum_{L'} f(L')} \tag{4.5}$$

where $f()$ is the frequency function. Note that since $\sum_{C' \in L} f(C') = f(L)$ we get that $\hat{p}(C, L) = \hat{p}(C)$, as the class label is uniquely determined by the class qualified name (the opposite does not hold since multiple class types may correspond to the same label). Finally, the term $p(L, W)$ is estimated by the symmetric string distance between the two strings, as described in Section 4.3.2. We consider the linking probability of $\langle X, Y \rangle$ to be $\hat{p}(X'|X) \cdot \hat{p}(Y'|Y)$, where X' is the best matching class for X s.t. $X' = \max_C \hat{p}(C|X)$ and similarly for Y'.

4.3.4 Code Distributional Similarity

Corpus distributional similarity evaluates the occurrence of words in particular semantic contexts. By defining the *class-context* of a Java class, we can then similarly calculate a *code distributional similarity* between classes. Our definition of class context is based on the usage of a class as an argument to methods and on the API which the class provides, and it is detailed in Table 4.1. We observe over 23K unique contexts in our code repository. Based on these definitions we can compute the distributional similarity measure between classes X' and Y' based on their code-context distributions, as previously described for the corpus distributional similarity (Section 4.3.1, following Pereira et al. [Pereira et al., 1993]). For the code-based case, we calculate the empirical context distribution of X' (see Equation 4.1) using $f(c, X')$, the occurrence frequency of class X' in context c, where c is one of the ARG-*Method* or API-*Method* contexts (defined in Table 4.1) for methods observed in the code repository. The distributional similarity of $\langle X', Y' \rangle$ is then taken, using the relative entropy, as $D(p_{X'}||p_{Y'}) + D(p_{Y'}||p_{X'})$.

4.3.5 Code Hierarchies and Organization

The words X and Y are defined as coordinate terms if they have the same hypernym in a given taxonomy, meaning they have at least one common ancestor in this taxonomy [Snow et al., 2004]. For the purpose of comparing two class types, we therefore define an ancestry relation between them using two taxonomies based on the code namespace and type hierarchies.

ARG-*Method*: The `Class` is being passed as an argument to the `Method`. We count an occurrence of this context once for the method definition,
`Method(Class class, ...)`
and for each method invocation,
`Method(class, ...)`
For example, given the statement
`str = toString(i);`
where i is an Integer, we would count an occurrence for the Integer class in the context ARG-toString.

API-*Method*: `Class` provides the API method `Method`. We count an occurrence of this context once for the method definition, and for every occurrence of method invocation, e.g., `class.Method(...)`. For example, given the statement `s = map.size();` where map is a HashMap, we would count an occurrence for the HashMap class in the context API-size.

Table 4.1: Definition of two types of code-contexts for a class type, `Class`, or an instantiation of that type (e.g., `class`).

Package Taxonomy: A package is the standard way for defining namespaces in the Java language. It is a mechanism for organizing sets of classes which normally share a common functionality. Packages are organized in a hierarchical structure which can be easily inferred from the class name. For example, `java.lang.String` is a class that belongs to the `java.lang` package, which belongs to the `java` package.

Type Taxonomy: The inheritance structure of classes and interfaces in the Java language defines a type hierarchy, such that class A is the ancestor of class B if B extends or implements A.

For each of the above taxonomies, we define type-ancestry and package-ancestry relations between classes $\langle X', Y' \rangle$. For the type taxonomy,

$A^n_{type}(X', Y')$ = {# of common ancestors X' and Y' share within *n* higher up

levels in the type taxonomy}

for *n* from 1 to 6. $A^n_{package}$ is defined similarly for the package taxonomy. As an example,

$$A^2_{package}(\text{ArrayList}, \text{Vector}) = 2$$

as these classes both belong in the package `java.util`, and therefore their common level 2 ancestors are: `java` and `java.util`. Moreover,

$$A^1_{type}(\text{ArrayList}, \text{Vector}) = 5$$

since both classes extend the `AbstractList` class, and also implement four joint interfaces: `List`, `RandomAccess`, `Cloneable`, and `Serializable`.

4.4 Experimental Settings

4.4.1 Data Handling

We downloaded a dump of the interactions on the StackOverflow website[2] from its launch date in 2008 and until 2012. We use only the 277K questions labeled with the user-assigned *Java* tag, and their 629K answers.

Text from the SO html posts was extracted with the Apache Tika toolkit[3] and then tokenized with the Mallet statistical NLP package [McCallum, 2002]. In this study, we use only the text portions of the SO posts, and exclude all raw code segments, as indicated by the user-labeled $<code>$ markup. Next, the text was POS tagged with the Stanford POS tagger [Toutanova et al., 2003] and parsed with the MaltParser [Nivre et al., 2006]. Finally, we extract noun pairs with the conjunction dependencies: *conj* or *inv-conj*, a total of 255,150 pairs, which we use as positive training samples.

We use the Java standard libraries code repository as a grounding source for Java classes, as we expect that users will often refer to these classes in the *Java* tagged SO posts. This data includes: 7072 source code files, the implementation of 10562 class and interface types, and 477 packages. The code repository is parsed using the Eclipse JDT compiler tools, which provide APIs for accessing and manipulating Abstract Syntax Trees.

[2]http://www.clearbits.net/creators/146-stack-exchange-data-dump
[3]http://tika.apache.org/

4.4.2 Classification

We follow the classification pipeline described in Figure 4.3, using the LibLinear SVM classifier [Fan et al., 2008, Chang and Lin, 2011] with the following features:

Corpus-Based Features

- *Corpus distributional similarity* (Corpus Dist. Sim.) - see Section 4.3.1.
- *String similarity* (String Sim.) - see Section 4.3.2.

Code-Based Features

- *Text to code linking probability* (Text-to-code Prob.) - see Section 4.3.3.
- *Code distributional similarity* (Code Dist. Sim.) - see Section 4.3.4.
- *Package and type ancestry* ($A^1_{package}$ - $A^6_{package}$ and A^1_{type} - A^6_{type}) - see Section 4.3.5.

Since the validity of the code based features above is directly related to the success of the entity linking phase, each of the code based features are used in the classifier once with the original value and a second time with the value weighted by the text to code linking probability.

Of the noun pairs $\langle X, Y \rangle$ in our data, we keep only pairs for which the linking probability $\hat{p}(X'|X) \cdot \hat{p}(Y'|Y)$ is greater than 0.1. Note that this guarantees that each noun must be mapped to at least one class with non-zero probability. Next, we evaluate the string morphology and its resemblance to a camel-case format, which is the acceptable formatting for Java class names. We therefore select alphanumeric terms with at least two upper-case and one lower-case characters. We name this set of noun pairs the *Coord* dataset.

A key assumption underlying statistical distributional similarity approaches is that "high-interest" entities are associated with higher corpus frequencies, therefore, given sufficient statistical evidence "high-interest" relations can be extracted. In the software domain, real world factors may introduce biases in a software-focused text corpus which may affect the corpus frequencies of classes: e.g., users may discuss classes based on the clarity of their API, the efficiency of their implementation, or simply if they are fundamental in software introduced to novice users. Another motivation for using grounded data, such as the class implementation, is that it may highlight additional aspects of interest, for example, classes that are commonly inherited from. We therefore define a second noun dataset, *Coord-PMI*, which attempts to address this issue, in which noun pairs are selected

High PMI	Low PMI
⟨**JTextField, JComboBox**⟩	⟨threads, characters⟩
⟨**yearsPlayed, totalEarned**⟩	⟨**server, user**⟩
⟨**PostInsertEventListener, PostUpdateEventListener**⟩	⟨**code, design**⟩
⟨**removeListener, addListener**⟩	⟨**Java, client**⟩
⟨MinTreeMap, MaxTreeMap⟩	⟨Eclipse, array⟩

Table 4.2: Sample set of word pairs with high and low *PMI* scores. Many of the high *PMI* pairs refer to software entities such as variable, method and Java class names, whereas the low *PMI* pairs contain more general software terms.

based on their pointwise mutual information (*PMI*):

$$PMI(X, Y) = \log \frac{p(X, Y)}{p(X)p(Y)} \tag{4.6}$$

where the frequency of the pair $\langle X, Y \rangle$ in the corpus is positive. In this set we include coordinate term pairs with high *PMI* scores, which appear more rarely in the corpus and are therefore harder to predict using standard NLP techniques. The negative set in this data are noun pairs which appear frequently separately but do not appear as coordinate terms, and are therefore marked by low *PMI* scores.

To illustrate this point, we provide a sample of noun pairs with low and high *PMI* scores in Table 4.2, where pairs highlighted with bold font are labeled as coordinate terms in our data. We can see that the high *PMI* set contains pairs that are specific and interesting in the software domain while not necessarily being frequent words in the general domain. For example, some pairs seem to represent variable names (e.g., ⟨*yearsPlayed, totalEarned*⟩), others likely refer to method names (e.g., ⟨*removeListener, addListener*⟩). Some pairs refer to Java classes, such as ⟨*JTextField, JComboBox*⟩ whose implementation can be found in the Java code repository. We can also see examples of pairs such as ⟨*PostInsertEventListener, PostUpdateEventListener*⟩ which are likely to be user-defined classes with a relationship to the Java class `java.util.EventListener`. In contrast, the low *PMI* set contains more general software terms (e.g., *code, design, server, threads*).

4.5 Results

4.5.1 Classification and Feature Analysis

Method	*Coord*	*Coord-PMI*
Code & Corpus	**85.3**	**88**
Baselines:		
Corpus Dist. Sim.	57.8	58.2
String Sim.	65.2	65.8
Corpus Only	64.7	60.9
Code Only	80.1	81.1
Code Features:		
Code Dist. Sim.	67 (60.2)	67.2 (59)
$A^1_{package}$	64.2 (63.8)	64.3 (63.9)
$A^2_{package}$	64.2 (63.8)	61.2 (64.8)
$A^3_{package}$	65.8 (64.3)	66 (64.6)
$A^4_{package}$	52.5 (52)	64.7 (58.7)
$A^5_{package}$	52.5 (52)	52.6 (58.7)
$A^6_{package}$	50.4 (51.6)	52.3 (52)
A^1_{type}	51.4 (51.4)	55.1 (53.7)
A^2_{type}	54 (53.9)	55.5 (54.3)
A^3_{type}	56.8 (56.7)	57 (56.9)
A^4_{type}	57.1 (56.9)	57.3 (57.1)
A^5_{type}	57.4 (57.6)	58 (57.9)
A^6_{type}	57.2 (57.4)	57.5 (57.5)
Text-to-code Prob.	55.7	55.8

Table 4.3: Cross validation accuracy results for the coordinate term SVM classifier (Code & Corpus), as well as baselines using corpus distributional similarity, string similarity, all corpus based features (Corpus Only), or all code based features (Code Only), and all individual code based features. The weighted version of the code based features (see Section 4.4.2) is in parenthesis. Results are shown for both the *Coord* and *Coord-PMI* datasets.

In Table 4.3 we report the cross validation accuracy of the coordinate term classifier (*Code & Corpus*) as well as baseline classifiers using corpus distributional similarity (*Cor-*

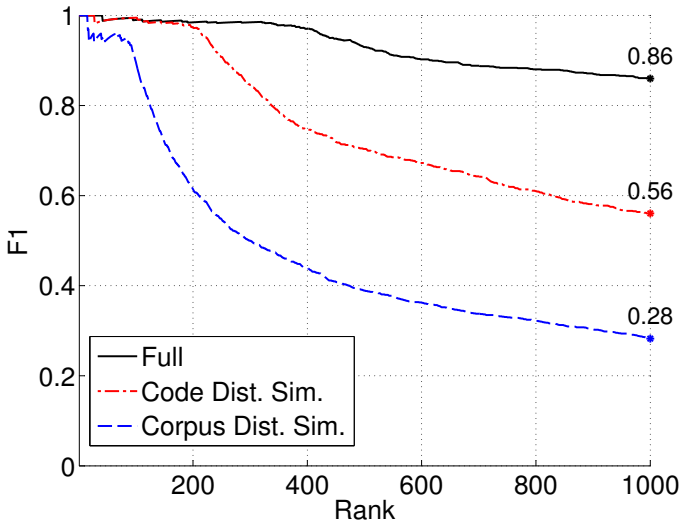


Figure 4.4: Manual Labeling Results. F1 results of the top 1000 predicted coordinate terms by rank. The final data point in each line is labeled with the F1 score at rank 1000.

pus Dist. Sim.), string similarity (*String Sim.*), all corpus features (*All Corpus*), or all code features (*All Code*). Note that using all code features is significantly more successful on this data than any of the corpus baselines (corpus baselines' accuracy is between 57%-65% whereas code-based accuracy is over 80%). When using both data sources, performance is improved even further (to over 85% on the *Coord* dataset and 88% on *Coord-PMI*).

We provide an additional feature analysis in Table 4.3, and report the cross validation accuracy of classifiers using each single code feature. Interestingly, code distributional similarity (*Code Dist. Sim.*) is the strongest single feature, and it is a significantly better predictor than corpus distributional similarity, achieving around 67% v.s. around 58% for both datasets.

4.5.2 Evaluation by Manual Labeling

The cross-validation results above are based on labels extracted using Hearst conjunction patterns. In Figure 4.4 we provide an additional analysis based on manual human labeling of samples from the *Coord-PMI* dataset, following a procedure similar to prior researchers exploring semi-supervised methods for relation discovery [Carlson et al., 2010, Lao et al., 2011]. After all development was complete, we hand labeled the top 1000 coordinate term pairs according to the ranking by our full classifier (using all code and corpus features)

Code Distributional Similarity	$A^3_{package}$	A^5_{type}
⟨FileOutputStream,OutputStream⟩	⟨KeyEvent,KeyListener⟩	⟨JMenuItem,JMenu⟩
⟨AffineTransform, AffineTransformOp⟩	⟨StyleConstants, SimpleAttributeSet⟩	⟨JMenuItems,JMenu⟩
⟨GZIPOutputStream, DeflaterOutputStream⟩	⟨BlockQueue, ThreadPoolExecutor⟩	⟨JMenuItems,JMenus⟩
⟨OutputStream, DataOutputStream⟩	⟨BufferedImage, WritableRaster⟩	⟨JLabel, DefaultTreeCellRenderer⟩
⟨AtomicInteger, AtomicIntegerArray⟩	⟨MouseListener, MouseWheelListener⟩	⟨JToggleButton, JRadioButtonMenu⟩
⟨ResourceBundle, ListResourceBundle⟩	⟨DocumentBuilderFactory, DocumentBuilder⟩	⟨JFrame,JDialogs⟩
⟨setIconImages,setIconImage⟩	⟨ActionListeners, FocusListeners⟩	⟨JTable,JTableHeader⟩
⟨ComboBoxModel, DefaultComboBoxModel⟩	⟨DataInputStream, DataOutputStream⟩	⟨JTextArea,JEditorPane⟩
⟨JTextArea,TextArea⟩	⟨greaterOrEqualThan, lesserOrEqualThan⟩	⟨JTextPane,JEditorPane⟩
⟨ServerSocketChannel, SocketChannel⟩	⟨CopyOnWriteArrayList, ConcurrentLinkedQueue⟩	⟨JTextArea,JTable⟩

Table 4.4: Top ten coordinate terms predicted by classifiers using one of the following features: code distributional similarity, package hierarchy ancestry ($A^3_{package}$), and type hierarchy ancestry (A^5_{type}). All of the displayed predictions are *true*.

and the top 1000 pairs predicted by the classifiers based on code and corpus distributional similarities only. We report the F1 results of each classifier by the rank of the predicted samples. According to our analysis, the F1 score for the text and code distributional similarity classifiers degrades quickly after the first 100 and 200 top ranked pairs, respectively. At rank 1000, the score of the full classifier is at 86%, whereas the code and text classifiers are only at 56% and 28%.

To highlight the strength of each of the code based features, we provide in Table 4.4 the top ten coordinate terms predicted using the most successful code based features. For example, the top prediction using type hierarchy ancestry (A^5_{type}) is ⟨JMenuItem, JMenu⟩. Since `JMenu` extends `JMenuItem`, the two classes indeed share many common interfaces and classes. Alternatively, all of the top predictions using the package hierarchy ancestry ($A^3_{package}$) are labels that have been matched to pairs of classes that share at

least 3 higher up package levels. So for example, `BlockQueue` has been matched to `java.util.concurrent.BlockingQueue` which was predicted as a coordinate term of `ThreadPoolExecutor` which belongs in the same package. Using code distributional similarity, one of the top predictions is the pair

$$\langle \text{GZIPOutputStream, DeflaterOutputStream} \rangle$$

which share many common API methods such as `write`, `flush`, and `close`. Many of the other top predicted pairs by this feature have been mapped to the same class and therefore have the exact same context distribution.

4.5.3 Taxonomy Construction

We visualize the coordinate term pairs predicted using our method (with all features), by aggregating them into a graph where entities are nodes and edges are determined by a coordinate term relation (Figure 4.1). Graph edges are colored using the Louvain method [Blondel et al., 2008] for community detection and an entity label's size is determined by its betweenness centrality degree. We can see that high-level communities in this graph correspond to class functionality (Figure 4.2), indicating that our method can be used to create an interesting code taxonomy.

Note that our predictions also highlight connections within functional groups that cannot be found using the package or type taxonomies directly. One example can be highlighted within the GUI functionality group. `Listener` classes facilitate a response mechanism to GUI `Actions`, such as *pressing a button*, or *entering text*, however, these classes belong in different packages than basic GUI components for historical reasons. In our graph, Action and Listener classes belong to the same communities of the GUI components they are normally used with.

4.6 Conclusions

We have presented an approach for grounded discovery of coordinate term relationships between text entities representing Java classes. Using a simple entity linking method we map text entities to an underlying class type implementation from the Java standard libraries. With this code-based grounding, we extract information on the usage-pattern of the class and its location in the Java class and namespace hierarchies.

Our experimental evaluation shows that using only corpus distributional similarity for the coordinate term prediction task is unsuccessful, achieving prediction accuracy of around 58%. However, adding information based on the entities' software implementation improves accuracy dramatically to 88%. Our classifier has an F1 score of 86% according to human labeling over the top 1000 predicted pairs.

The coordinate relations predicted here can be useful for collecting semantic information on Java classes. We have shown that by aggregating predicted relations, we get an interesting code taxonomy which draws from the functional connections, common usage-patterns, and implementation details that are shared between classes. This taxonomy can be viewed as a hierarchical clustering of classes into semantic classes, meaning that, in practice, it can be used as an ontology over Java classes; such as ontology is an important starting point for ontology-guided knowledge base population methods, which are aimed at constructing knowledge bases given an input ontological structure over entities in a target domain. The advantage of an ontology such as the one derived here is that it reflects statistics in both language and code. This is in contrast to the human-created ontologies which are typically used in open-domain KB population methods.

Chapter 5

Topic Model Based Approach for Learning a Complete Knowledge Base

In Chapter 4 we constructed a grounded ontology for the software domain, based on relations discovered between entities from the domain. Our grounding method links entities from a text corpus to a code repository, and relations are found using both sources. This leads to an ontology that reflects statistics in language and code. However, grounding software entities directly to code limits the scope of the learned ontology, as it can only reason over entities that appear directly in the code. The resulting ontology is, therefore, missing higher level software concepts, including ones that discuss the users of the applications, the computer resources it consumes, and the design patterns used in its implementation. To address this issue, we propose a general approach for Knowledge Base construction which exploits more information found in the text, and therefore includes higher level concepts.

Many existing KBs, including Freebase, Yago, and NELL, rely on a fixed ontology, given as an input to the system, which defines the data to be cataloged in the KB, i.e., a hierarchy of categories and relations between them. The system then extracts facts that match the predefined ontology, where in some cases, the input ontology is later extended with automatically discovered relations [Mohamed et al., 2011]. We propose an unsupervised model that jointly learns a latent ontological structure of an input corpus, and identifies facts from the corpus that match the learned structure. Our approach combines mixed membership stochastic block models and topic models to infer a structure by jointly modeling text, a latent concept hierarchy, and latent semantic relationships among the entities mentioned in the text. As a case study, we apply the model to a corpus of Web documents from the software domain, meaning that we are learning a software KB. We evaluate the accuracy of the various components of the learned structure. Additionally,

we show how a learned KB can be used to extract domain-specific facts from an open IE resource.

5.1 Introduction

Knowledge base (KB) construction methods can be broadly categorized along several dimensions. One dimension is ontology-guided construction, where the list of categories and relations that define the schema of the KB are explicit, versus open IE methods, where they are not. Another dimension is the type of relations and types included in the KB: some KBs, like WordNet, are hierarchical, in that they contain mainly concept types, supertypes and instances, while other KBs contain many types of relationships between concepts. Hierarchical knowledge can be learned by methods including distributional clustering [Pereira et al., 1993], as well as Hearst patterns [Hearst, 1992] and similar techniques [Snow et al., 2006]. Reverb [Fader et al., 2011] and TextRunner [Yates et al., 2007] are open methods for learning multi-relation KBs. Finally, NELL [Carlson et al., 2010, Mitchell et al., 2015], FreeBase [Google, 2011] and Yago [Suchanek et al., 2007, Hoffart et al., 2013] are ontology-guided methods for extracting KBs containing both hierarchies and relations.

One advantage of ontology-guided methods is that the extracted knowledge is easier to reason with. An advantage of open IE methods is that ontologies may be incomplete, and are expensive to construct for a new domain. Ontology design involves assembling a set of categories, organized in a meaningful hierarchical structure, often providing seeds, i.e., representative examples for each category, and finally, defining inter-category relations. This process is often done manually [Carlson et al., 2010] leading to a rigid set of categories. Redesigning a new ontology for a specialized domain represents an additional challenge as it requires extensive knowledge of the domain.

In this chapter, we propose an unsupervised model that learns a latent hierarchical structure of categories from an input corpus, learns latent semantic relations between categories, and also identifies facts from the corpus that match the learned structure. In other words, the model learns both the schema for a KB, and a set of facts that are related to that schema, thus combining the processes of KB population and ontology construction. The intent is to build systems that extract facts which can be interpreted relative to a meaningful ontology without requiring the effort of manual ontology construction.

The input to the learning method is a corpus of documents, plus two sets of resources extracted from the same corpus: (1) a set of hypernym-hyponym pairs (e.g., “animal”, “horse”) extracted using Hearst patterns, and (2) a set of subject-verb-object triples (e.g.,

"horse", "eats", "hay") extracted from parsed sentences. These resources are analogous to the output of open IE systems for hierarchies and relations, and as we demonstrate, our method can be used to highlight domain-specific data from open IE repositories.

Our approach combines mixed membership stochastic block models and topic models to infer a structure by jointly modeling text documents, and links that indicate hierarchy and relation among the entities mentioned in the text. Joint modeling allows information on topics of nouns (referred to as *instances*) and verbs (referred to as *relations*) to be shared between text documents and an ontological structure, resulting in a set of compelling topics. This model offers a complete solution for KB construction based on an input corpus, and we therefore name it *KB-LDA*.

We additionally propose a method for recovering meaningful names for concepts in the learned hierarchy. These are equivalent to category names in other KBs, however, following our method we extract from the data a set of potential alternative concepts describing each category, including probabilities for their strength of association.

To show the effectiveness of our method, we apply the model to a dataset of Web based documents from the software domain, and learn a software KB. This is an example of a specialized domain in which, to our knowledge, no broad-coverage ontology exists. We evaluate the model on the induced categories, relations, and facts, and we compare the proposed categories with an independent set of human-provided labels for documents. Finally, we use KB-LDA to retrieve domain-specific relations from an open IE resource. We provide the learned software KB as supplemental material, and it can be downloaded from: http://www.cs.cmu.edu/~dmovshov/data/kb_lda_acl2015_supplementaryMaterial.zip.

5.2 KB-LDA

Modeling latent sets of entities from observed interactions among them is a well researched task, often encountered in social network analysis for the purpose of identifying specialized communities in the network. Mixed Membership Stochastic Blockmodels [Airoldi et al., 2009, Parkkinen et al., 2009] model entities as graph nodes with pairwise relations drawn from latent blocks with mixed membership. A related approach is taken by topic models such as LDA (Latent Dirichlet Allocation; [Blei et al., 2003]), which model documents as generated by a mixture of latent topics, and words in the documents as generated by topic-specific word distributions. The KB-LDA model combines the two approaches. It models links between tuples of two or three entities using stochastic block models, and these are additionally influenced by latent topic assignments of the entities in

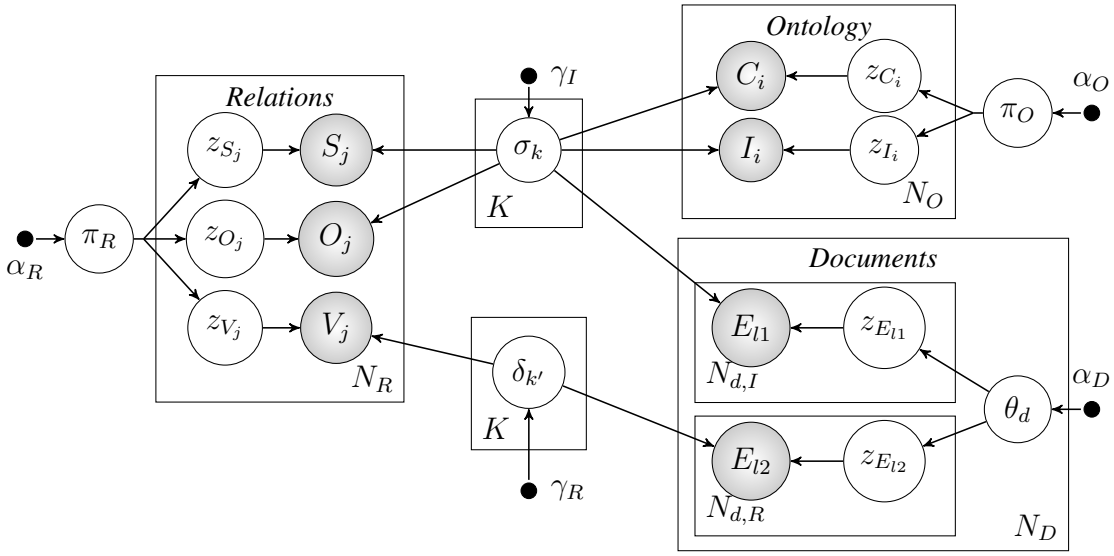


Figure 5.1: Plate diagram of KB-LDA: a model for constructing knowledge bases, which extends LDA topic models and mixed-membership stochastic block models. In this model, information is shared between three main components (*Ontology*, *Relations*, and *Documents*), through common latent topics over noun phrases (σ_k) and verb phrases ($\delta_{k'}$). Each of the presented components maps to an element in the learned knowledge base structure. For more details see Section 5.2.

a document corpus.

In the KB-LDA model, shown as a plate diagram in Figure 5.1 with notation in Table 5.1, information is shared between three components, through common latent topics over noun and verb entities. The *Ontology* component (upper right) models hierarchical links between Concept-Instance (CI) entity pairs. The *Relations* component (left) models links between Subject-Verb-Object (SVO) entity triples, where the subject and object are nouns and the verb represents a relation between them. Finally, the *Documents* component (lower right) is a link-LDA model [Erosheva et al., 2004] of text documents containing a combination of noun and verb entity types. In this formulation, distributions over noun and verb entities that are related according to hierarchical or relational constraints, are linked with a text model via shared parameters.

Each of the components presented in Figure 5.1 maps to an element in the learned KB structure. The *Documents* component provides the context in which noun and verb entities co-occur in text documents. It is modeled as a link-LDA model, an extension of LDA, viewing documents as sets of "bags of words"; in this case, each bag contains either noun

$\boldsymbol{\pi_O}$ – multinomial over ontology topic pairs, with Dirichlet prior α_O
$\boldsymbol{\pi_R}$ – multinomial over relation topic tuples, with Dirichlet prior α_R
$\boldsymbol{\theta_d}$ – topic multinomial for document d, with Dirichlet prior α_D
$\boldsymbol{\sigma_k}$ – multinomial over instances for topic k, with Dirichlet prior γ_I
$\boldsymbol{\delta_{k'}}$ – multinomial over relations for topic k', with Dirichlet prior γ_R
$\text{CI}_i = \langle C_i, I_i \rangle$ – i-th ontological assignment pair
$\text{SVO}_j = \langle S_j, O_j, V_j \rangle$ – j-th relation assignment tuple
$\boldsymbol{z_i^{CI}} = \langle z_{C_i}, z_{I_i} \rangle$ – topic pair chosen for example $\langle C_i, I_i \rangle$
$\boldsymbol{z_j^{SVO}} = \langle z_{S_j}, z_{O_j}, z_{V_j} \rangle$ – topic tuple chosen for example $\langle S_j, O_j, V_j \rangle$
$\boldsymbol{z^D_{E_1}}, \boldsymbol{z^D_{E_2}}$ – topic chosen for instance entity E_1, or relation entity E_2, in a document
$\boldsymbol{n^I_{z,i}}$ – number of times instance i is observed under topic z (in either z^D, z^{CI} or z^{SVO})
$\boldsymbol{n^R_{z,r}}$ – number of times relation r is observed under topic z (in either z^D or z^{SVO})
$\boldsymbol{n^O_{\langle z_c, z_i \rangle}}$ – count of ontological pairs assigned the topic pair $\langle z_c, z_i \rangle$ (in z^{CI})
$\boldsymbol{n^R_{\langle z_s, z_o, z_v \rangle}}$ – count of relation tuples assigned the topic tuple $\langle z_s, z_o, z_v \rangle$ (in z^{SVO})

Table 5.1: KB-LDA notation.

or verb entities. This component contributes to learning topics over the two entity types. As a result, each entity type has an individual topic-wise multinomial distribution over the set of entities in the vocabulary of that type. Learned topics over noun phrases (σ_k) correspond to categories in the KB, as they aggregate similar nouns. Consider, for example, a topic in which the most probable nouns include 'stackoverflow', 'google', and 'twitter', which are all names of websites. Similarly, topics over verb phrases ($\delta_{k'}$) aggregate verbs that describe similar actions. Verb topics correspond to relations in the learned KB. As an example, we learn a verb topic in which the most probable verbs include 'clicks', 'selects', and 'hits', all of which can be used to describe an interaction between users and user-interface elements.

The *Ontology* component is a generative model which learns hierarchal links between pairs of noun topics. The training examples for this component are extracted using a small collection of Hearst patterns indicating concept-to-instance or superconcept-to-subconcept links, including patterns like, 'X such as Y', and 'X including Y'. For example, the sentence "websites such as StackOverflow" indicates that Stackoverflow is a type of website, leading to the extracted noun pair ⟨websites, StackOverflow⟩, in which 'websites' is a concept and 'StackOverflow' is an instance of that concept. We refer to the examples extracted using these hierarchical patterns as *concept-instance pairs*, and to the individual entities as *instances*.

Let K be the number of target latent topics.

1. **Generate topics:** For topic $k \in 1, \ldots, K$, sample:
   - $\sigma_k \sim$ Dirichlet(γ_I), the per-topic instance distribution
   - $\delta_k \sim$ Dirichlet(γ_R), the per-topic relation distribution
2. **Generate ontology:** Sample $\pi_O \sim$ Dirichlet(α_O), the instance topic pair distribution.
   - For each concept-instance pair CI$_i$, $i \in 1, \ldots, N_O$:
     - Sample topic pair $z_i^{CI} \sim$ Multinomial(π_O)
     - Sample instances $C_i \sim$ Multinomial($\sigma_{z_{C_i}}$), $I_i \sim$ Multinomial($\sigma_{z_{I_i}}$), then CI$_i = \langle C_i, I_i \rangle$
3. **Generate relations:** Sample $\pi_R \sim$ Dirichlet(α_R), the relation topic tuple distribution.
   - For each tuple SVO$_j$, $j \in 1, \ldots, N_R$:
     - Sample topic tuple $z_j^{SVO} \sim$ Multinomial(π_R)
     - Sample instances, $S_j \sim$ Multinomial($\sigma_{z_{S_j}}$), $O_j \sim$ Multinomial($\sigma_{z_{O_j}}$), and sample a relation $V_j \sim$ Multinomial($\delta_{z_{V_j}}$)
4. **Generate documents:** For document $d \in 1, \ldots, D$:
   - Sample $\theta_d \sim$ Dirichlet(α_D), the topic mixing distribution for document d.
   - For every noun entity (E_{l1}) and verb entity (E_{l2}), $l1 \in 1, \ldots, N_{d,I}$, $l2 \in 1, \ldots, N_{d,R}$:
     - Sample topics $z_{E_{l1}}, z_{E_{l2}} \sim$ Multinomial(θ_d)
     - Sample entities $E_{l1} \sim$ Multinomial($\sigma_{z_{E_{l1}}}$) and $E_{l2} \sim$ Multinomial($\delta_{z_{E_{l2}}}$)

Table 5.2: KB-LDA generative process.

The extracted concept-instance pairs have an underlying block structure, which is derived from a sparse block model [Parkkinen et al., 2009], meaning that they indicate a latent hierarchical structure. Each example pair is generated by topic specific instance distributions conditioned on topic pair edges, which are defined by the multinomial π_O over the Cartesian product of the noun topic set with itself. The individual instances, therefore, have a mixed membership in topics. Note that we allow for a concept and instance to be drawn from different noun topics, defined by σ. As an example, we may learn one noun topic highlighting concept tokens like 'websites', 'platforms', and 'applications', and a second noun topic highlighting instances shared by these concepts, such as, 'stackoverflow', 'google', and 'facebook'. The observation that the former topic frequently contains concepts of instances from the latter topic, is encoded in the multinomial distribution π_O. From this we infer that the former topic should be placed higher in the induced hierarchy. More generally, the multinomial encodes a block structure over all pairs of noun topics, and this allows us to infer an optimal hierarchical structure between them. This derived structure corresponds to the ontology of the learned KB.

Similarly, the *Relations* component infers a block structure between a noun topic representing the subject of a relation, a verb topic representing the relation itself, and a second noun topic representing the object of a relation. The inferred block structure corresponds

to typed relation definitions in the learned KB. The *Relations* component is trained over examples of relational links between a noun subject, a verb, and a noun object. These examples are extracted from SVO patterns found in the document corpus, following the methodology of Talukdar et al. [2012a]. An extracted example looks like: ⟨websites, execute, javascript⟩. Subject and object topics are drawn from the noun topics (σ), while the verb topics is drawn from the verb topics, defined by δ. The multinomial π_R encodes the interaction of noun and verb topics based on the extracted relational links, and it is defined over the Cartesian product of the noun topic set with itself and with the verb topic set.

The generative process of KB-LDA is described in Table 5.2. Given the hyperparameters $(\alpha_O, \alpha_R, \alpha_D, \gamma_I, \gamma_R)$, the joint distribution over CI pairs, SVO tuples, documents, topics and topic assignments is given by

$$p(\pi_O, \pi_R, \sigma, \delta, \mathbf{CI}, \boldsymbol{z^{CI}}, \mathbf{SVO}, \boldsymbol{z^{SVO}}, \boldsymbol{\theta}, \mathbf{E}, \boldsymbol{z^D} | \alpha_O, \alpha_R, \alpha_D, \gamma_I, \gamma_R) = \tag{5.1}$$

$$\underbrace{\prod_{k=1}^{K} \text{Dir}(\sigma_k|\gamma_I)}_{\text{Instance topics}} \times \underbrace{\prod_{k'=1}^{K} \text{Dir}(\delta_{k'}|\gamma_R)}_{\text{Relation topics}} \times \underbrace{\prod_{d=1}^{N_D} \text{Dir}(\theta_d|\alpha_D) \prod_{l1=1}^{N_{d,I}} \theta_d^{z^D_{E_{l1}}} \sigma^{E_{l1}}_{z^D_{E_{l1}}} \prod_{l2=1}^{N_{d,R}} \theta_d^{z^D_{E_{l2}}} \delta^{E_{l2}}_{z^D_{E_{l2}}}}_{\textit{Documents} \text{ component}} \times$$

$$\underbrace{\text{Dir}(\pi_O|\alpha_O) \prod_{i=1}^{N_O} \pi_O^{\langle z_{C_i}, z_{I_i} \rangle} \sigma^{C_i}_{z_{C_i}} \sigma^{I_i}_{z_{I_i}}}_{\textit{Ontology} \text{ component}} \times \underbrace{\text{Dir}(\pi_R|\alpha_R) \prod_{j=1}^{N_R} \pi_R^{\langle z_{S_j}, z_{O_j}, z_{V_j} \rangle} \sigma^{S_j}_{z_{S_j}} \sigma^{O_j}_{z_{O_j}} \delta^{V_j}_{z_{V_j}}}_{\textit{Relations} \text{ component}}$$

5.2.1 Inference in KB-LDA

Exact inference is intractable in the KB-LDA model. We use a collapsed Gibbs sampler [Griffiths and Steyvers, 2004] to perform approximate inference in order to query the topic distributions and assignments. It samples a latent topic pair for a CI pair in the corpus conditioned on the assignments to all other CI pairs, SVO tuples, and document entities, using the following expression, after collapsing π_O:

$$\hat{p}(z_i^{CI} | \text{CI}_i, z_{\neg i}^{CI}, z^{SVO}, z^D, \text{CI}_{\neg i}, \alpha_O, \gamma_I) \propto \tag{5.2}$$

$$\left(n_{z_i^{CI}}^{O \neg i} + \alpha_O \right) \times \frac{(n^{I \neg i}_{z_{C_i}, C_i} + \gamma_I)(n^{I \neg i}_{z_{I_i}, I_i} + \gamma_I)}{(\sum_C n^{I \neg i}_{z_{C_i}, C} + T_I \gamma_I)(\sum_I n^{I \neg i}_{z_{I_i}, I} + T_I \gamma_I)}$$

where counts of observations from the training set are noted by n (see Table 5.1), and T_I is the number of instance entities (size of noun vocabulary).

We similarly sample topics for each SVO tuple conditioned on the assignments to all other tuples, CI pairs and document entities, using the following expression, after collapsing π_R:

$$\hat{p}(z_j^{SVO}|\text{SVO}_j, z_{\neg j}^{SVO}, z^{CI}, z^D, \text{SVO}_{\neg j}, \alpha_R, \gamma_I, \gamma_R) \propto \left(n_{z_j^{SVO}}^{R\neg j} + \alpha_R\right) \times \frac{(n_{z_{S_j},S_j}^{I\neg j} + \gamma_I)(n_{z_{O_j},O_j}^{I\neg j} + \gamma_I)(n_{z_{V_j},V_j}^{R\neg j} + \gamma_R)}{(\sum_I n_{z_{S_i},I}^{I\neg j} + T_I\gamma_I)(\sum_I n_{z_{O_i},I}^{I\neg j} + T_I\gamma_I)(\sum_V n_{z_{V_j},V}^{R\neg j} + T_R\gamma_R)} \tag{5.3}$$

We sample a latent topic for an entity mention in a document from the text corpus conditioned on the assignments to all other entity mentions after collapsing θ_d. The following expression shows topic sampling for a noun entity in a document:

$$\hat{p}(z_{E_{l1}}|E, \text{CI}, \text{SVO}, z^D, z^{CI}, z^{SVO}, \alpha_D, \gamma_I) \propto (n_{d,z}^{\neg l1} + \alpha_D)\frac{n_{z_{E_{l1}},E_{l1}}^{I\neg l1} + \gamma_I}{\sum_{E'_{l1}} n_{z_{E_{l1}},E'_{l1}}^{I\neg l1} + T_I\gamma_I} \tag{5.4}$$

The per-topic multinomial parameters and topic distributions of CI pairs, SVO tuples and documents can be recovered with MLE estimates using their observation counts:

$$\hat{\sigma}_k^I = \frac{n_{k,I}^I + \gamma_I}{\sum_{I'} n_{k,I'}^I + T_I\gamma_I} \tag{5.5}$$

$$\hat{\delta}_k^R = \frac{n_{k,R}^R + \gamma_R}{\sum_{R'} n_{k,R'}^R + T_R\gamma_R} \tag{5.6}$$

$$\hat{\theta}_d^z = \frac{n_{z,d} + \alpha_D}{\sum_{z'} n_{z',d} + K\alpha_D} \tag{5.7}$$

$$\hat{\pi}_O^{\langle z_C, z_I\rangle} = \frac{n_{\langle z_C, z_I\rangle}^O + \alpha_O}{\sum_{z'_C, z'_I} n_{\langle z'_C, z'_I\rangle}^O + K^2 \cdot \alpha_O} \tag{5.8}$$

$$\hat{\pi}_R^{\langle z_S, z_O, z_V\rangle} = \frac{n_{\langle z_S, z_O, z_V\rangle}^R + \alpha_R}{\sum_{z'_S, z'_O, z'_V} n_{\langle z'_S, z'_O, z'_V\rangle}^R + K^3 \cdot \alpha_R} \tag{5.9}$$

Using the KB-LDA model we can describe the latent topic hierarchy underlying the input corpus. We consider the multinomial of the *Ontology* component, π_O, as an adjacency matrix describing a directed network where the nodes are instance topics and edges

indicate a hypernym-to-hyponym relation. We recover a hierarchy over the instance topics by finding a spanning arborescence (maximum directed spanning tree) over this adjacency matrix, using Edmonds' optimum branchings algorithm [Edmonds, 1967]. We recover relations among instance topics by extracting from the Relations multinomial, π_R, the set of most probable tuples of a $\langle$subject topic, verb topic, object topic$\rangle$.

5.2.2 Parallel Implementation

Our model is implemented using a fast, parallel approximation of collapsed Gibbs sampling, following Newman et al. [2009a]. In each sampling iteration, topics are sampled locally on a subset of the training examples. At the end of each iteration, data from worker threads is joined and model parameters are updated with complete information. In the next iteration, thread-local sampling starts with complete topic assignment information from the previous iteration. In each thread, the process can be viewed as a reordering of the input examples, where the examples sampled in that thread are viewed first. It has been shown that parallel approaches considerably speed up iterative inference methods such as collapsed Gibbs sampling, resulting in test data log probabilities indistinguishable from those obtained using serial methods [Porteous et al., 2008, Newman et al., 2009a]. A parallel approach is especially important when training the KB-LDA model due to the large dimensions of the multinomials of the *Ontology* and *Relations* components (K^2 and K^3, respectively for a model with K topics). We train KB-LDA over 2000 iterations, more than what has traditionally been used for collapsed Gibbs samplers.

5.2.3 Data-driven discovery of topic concepts

The KB-LDA model described above clusters noun entities into sets of instance topics, and recovers a latent hierarchical structure among these topics. Each instance topic can be described by a multinomial distribution of the underlying nouns. It is often more intuitive, however, to refer to a topic containing a set of high probability nouns by a name, or category, just as traditional ontologies describe hierarchies over categories.

Our model is trained over nouns that originate from concept-instance example pairs (used to train the *Ontology* component). We describe a method for selecting a category name for a topic, based on concepts that best represent high probability nouns of the topic in the concept-instance examples.

We calculate the probability that a concept noun c describes the set of instances I that

have been assigned the topic z using

$$\begin{aligned} p(c, z|I) &\propto p(I|c, z) * p(c, z) \\ &= p(I|c, z) * p(z|c) * p(c) \end{aligned} \tag{5.10}$$

Let $rep(c, z) = \sum_{i:C_i=c} n^I_{z,I_i}$ describe how well concept c represents topic z according to the assignments of instances with concept c to the topic. Then,

$$p(z|c) = \frac{rep(c, z)}{\sum_{z'} rep(c, z')} \tag{5.11}$$

The concept prior, $p(c)$, is based on the relative weight of instances with concept c in the concept-instance example set, and is an indicator of the generality of a concept:

$$p(c) = \frac{\sum_{i:C_i=c} w_{c,I_i}}{\sum_{c'} \sum_{i:C_i=c'} w_{c',I_i}} \tag{5.12}$$

where $w_{C,I}$ is the number of occurrences of concept-instance pair $\langle C, I\rangle$ in the corpus.

Finally, $p(I|c, z)$ measures how specific are the topic instances to the concept c,

$$p(I|c, z) = \frac{\sum_{i:I_i \in I, C_i=c} w_{c,I_i}}{\sum_{i:C_i=c} w_{c,I_i}} \Big/ Z \tag{5.13}$$

where I is the set of training instances assigned with topic z, and Z is a normalizer over all concepts and topics.

Following this method we extract concepts that have a high probability $p(c, z|I)$ with respect to a topic z. These can be thought of as equivalent to the single, fixed, category name provided by traditional KB ontologies; however, here we extract *from the data* a set of potential alternative noun phrases describing each topic, including a probability for the strength of this association.

5.3 Experimental Evaluation

We evaluate the KB-LDA model on a corpus of 5.5M documents from the software domain; thereby we are using the model to construct a software domain knowledge base. Our evaluation explores the following questions:

- Does KB-LDA learn categories, relations, a hierarchy and topic concepts with high precision?

- How well do KB-LDA topics correspond with human-provided document labels?
- Is KB-LDA useful in extracting domain-specific facts from existing open IE resources?

5.3.1 Data

We use data from the Q&A website StackOverflow[1] where users ask and answer technical questions about software development, tools, algorithms, etc'. We extracted 562K concept-instance example pairs from the data, and kept the 17K examples appearing at least twice. Noun phrases in these examples make up our *Instance Dictionary*. Out of 6.8M SVO examples found in the data we keep 37K in which the subject and object are in the Instance Dictionary, and the example appears at least twice in the corpus. The verbs in these SVOs make up our *Relation Dictionary*. Finally, we consider as documents the 5.5M questions from StackOverflow with all their answers. The training data we use can be downloaded from http://www.cs.cmu.edu/~dmovshov/data/kb_lda_software_dataset_sample.tar

5.3.2 Evaluating the learned KB precision

In this section we evaluate the direct output of a model trained with 50 topics: the extracted instance topics, topic hierarchy, relations among topics and extracted topic concepts. A subset of a sample hierarchy learned with KB-LDA is presented in Figure 5.2, and sample relations can be found in Figure 5.3.

In each of the experiments below, we extract facts based on one of the learned components and evaluate each fact based on annotations from human judges: two experts and three non-expert users, collected using Mechanical Turk, that were pre-tested on a basic familiarity with concepts from the software domain, such as *programming languages, version control systems*, and *databases*.

Precision of Instance Topics

We measure the coherence of instance topics using an approach called *word intrusion* [Chang et al., 2009]. We extract the top 30 instance tokens of a topic ranked by the instance topic multinomial σ. We present to workers tokens 1-5,6-10,...,26-30, where

[1]Data source: https://archive.org/details/stackexchange

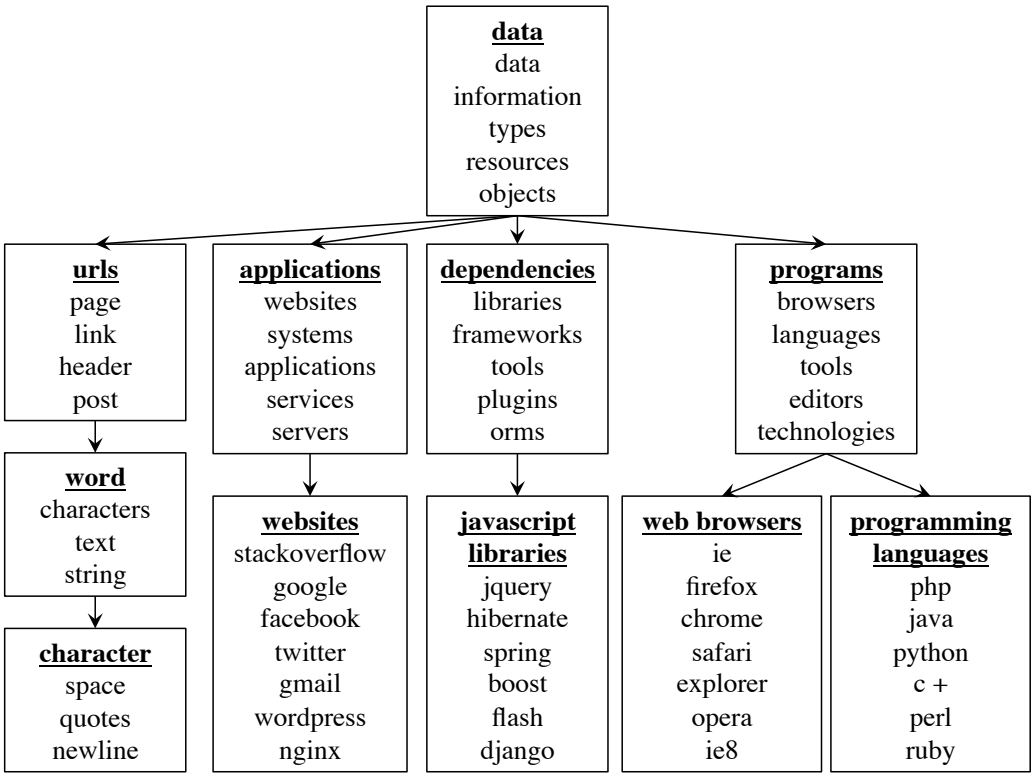


Figure 5.2: Subset of a sample of a hierarchy learned with KB-LDA with 50 topics. For each topic, we display a name extracted for the topic by our system (bold), and words selected from the top 20 words of that topic.

each 5 tokens are randomly ordered and augmented with an extra token that is ranked low for the topic, (the intruder). We ask workers to select all tokens that do not belong in the group (and at least one). We define the topic *Match Precision* as the fraction of questions for which the reviewer identified the correct intruder (out of 6 questions per topic), and the topic *Group Precision* as the fraction of correct tokens (those not selected as not belonging in the group). Thus *Match Precision* measures how well labelers understand the topic, and *Group Precision* measures what fraction of words appeared relevant to the topic.

Figure 5.4 shows the average Match and Group precision over the top tokens of all 50 topics learned with the model, as evaluated by expert and non-expert workers. Both groups find the intruder token in over 75% of questions. In the more subtle task of validating each topic token (Group precision) we see a greater variance among the two labeler groups. This highlights the difficulty of evaluating domain specific facts with non-expert users.

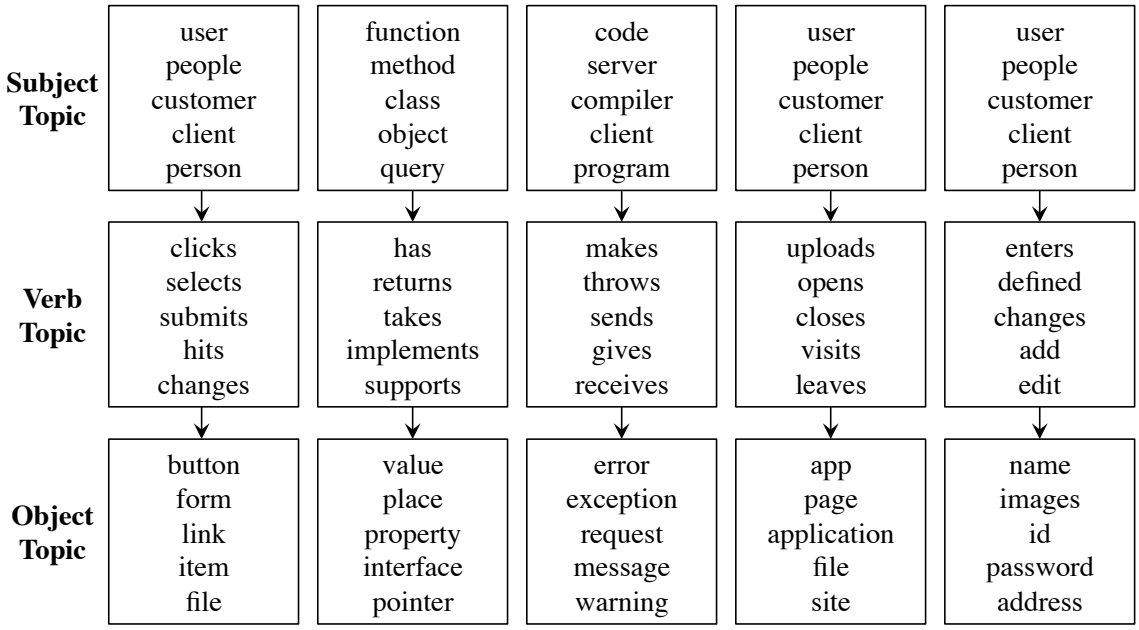


Figure 5.3: Sample relations learned with KB-LDA. For each relation we display words selected from the top 10 words of the subject topic, verb topic, and object topic.

Table 5.3 displays the top 20 instance topics learned with KB-LDA, ranked by expert Group precision.

Precision of Topic Concepts

We assess the precision of the top 5 concept names proposed for instance topics, following the method presented in Section 5.2.3. Top concepts for a subset of topics are shown in Table 5.3. For each topic, we present to the user a hypernym-hyponym pattern of the topic based on the top concepts and top instances of the topic. As an example, if the top 5 instances of a topic are *ie, firefox, chrome, buttons, safari* and the top 5 concepts for this topic are *web browsers, web browser, browser, ie, chrome*, the pattern presented to workers is

- [ie, firefox, chrome, buttons, safari] **is a** [web browsers, web browser, browser, ie, chrome]

Workers were asked to match at least 3 instances to a proposed concept name. In addition, the same assessment was applied for each topic using randomly sampled concepts. We

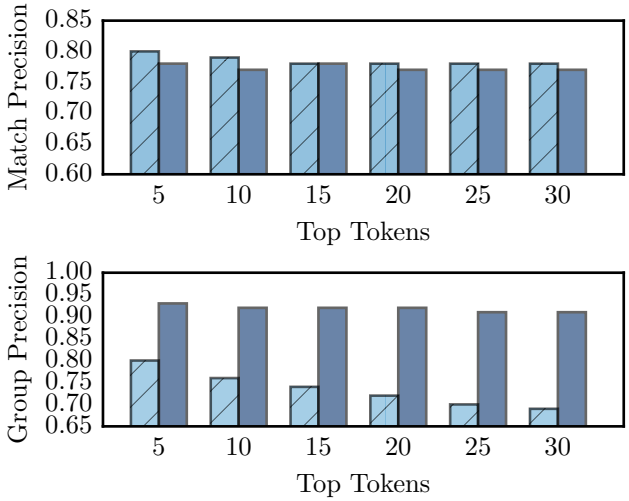


Figure 5.4: Average *Match* (top) and *Group* (bottom) precision of top tokens of 50 topics learned with KB-LDA, according to expert (dark blue) and non-expert (light blue, stripes) labeling.

present in Table 5.4 the number and precision of patterns based on extracted concepts (Concepts) and random concepts (Random), that were labeled by 1, 2 or 3 workers, as well as the average results among experts. We achieve nearly 90% precision according to expert labeling, however we do not observe large agreement among non-expert labelers.

Precision of Relations

To assess the precision of the relations learned in the KB-LDA model, we extract the top 100 relations learned according to their probability in the relation multinomial π_R. Relation patterns were presented to workers as sets of the top subject-verb-object tokens of the respective topics in the relation. An example relation is

- Subject words: [user, users, people, customer, client]
- Verb words: [clicks, selects, submits, click, hits]
- Object words: [function, method, class, object, query]

and workers are asked to state whether the pattern indicates a valid relation or not, by checking whether a reasonable number of combinations of subject-verb-object triples ex-

Top 2 Topic Concepts	Top 10 Topic Tokens
table, key	table, query, database, sql, column, data, tables, mysql, index, columns
properties, css	image, code, images, problem, point, color, data, size, screen, points
credentials, user information	name, images, id, number, text, password, address, strings, files, string
page, content	page, html, code, file, image, javascript, browser, http, jquery, js
orm tools, orm tool	tomcat, hibernate, server, boost, apache, spring, mongodb, framework, nhibernate, png
clients, apps	app, application, http, android, device, phone, code, api, iphone, google
applications, systems	devices, systems, applications, services, platforms, tools, sites, apps, system, service
systems, platforms	google, windows, linux, facebook, git, ant, database, gmail, android, so
limits, limit	memory, time, thread, code, threads, process, file, program, data, object
data, table	query, table, data, list, example, number, results, search, database, rows
type, value	code, function, value, type, pointer, array, memory, compiler, example, string
table, request	data, information, types, properties, details, fields, values, content, resources, attributes
dependencies, jar file	libraries, library, framework, frameworks, formats, format, database, databases, tools, server
type, object	value, focus, place, property, method, reference, interface, effect, pointer, data
kinds, code	languages, language, features, objects, functions, methods, code, operations, structures, types
element, elements	button, form, link, item, *file*, mouse, image, value, option, row
javascript libraries, javascript framework	jquery, mysql, http, json, xml, library, html, sqlite, *asp*, php
process, operating system	server, client, connection, *data*, http, socket, message, request, port, service
folder, files	file, files, directory, folder, path, *code*, name, resources, project, folders
value, array	array, list, value, values, number, string, code, elements, *loop*, object

Table 5.3: Top 20 instance topics learned with KB-LDA. For each topic we show the top 2 concepts recovered for the topic, and top 10 tokens. In *italics* are words marked as out-of-topic by expert labelers.

tracted from each of the relation groups can produce valid relations. We present in Table 5.4 the number and precision of patterns based on the top 100 relations (Relations) and 100 random relations (Random), that were labeled by 1, 2 or 3 workers, and the average results among experts. We achieve 80% precision according to experts, and only 18% on random relations. We observe similar agreement among expert and non-expert workers as in the concept evaluation experiment, however we note that random relations prove more confusing for non-experts and more of them are (falsely) labeled as correct.

Workers	Concepts		Relations		Subsumptions	
	KB-LDA (p)	Random (p)	KB-LDA (p)	Random (p)	KB-LDA (p)	Random (p)
1	48 (0.96)	6 (0.12)	90 (0.9)	69 (0.69)	31 (0.63)	28 (0.57)
2	42 (0.84)	0 (0.0)	63 (0.63)	22 (0.22)	16 (0.33)	9 (0.18)
3	26 (0.52)	0 (0.0)	15 (0.15)	4 (0.05)	3 (0.06)	4 (0.08)
Experts	44 (0.88)	0 (0.0)	70 (0.7)	13 (0.13)	25 (0.51)	4 (0.08)

Table 5.4: Precision of topic concepts, relations, and subsumptions. For items extracted from the model (KB-LDA), and randomly (Random), we show the number of items marked as correct, and precision in parentheses (p), as labeled by 1, 2, or 3 non-expert workers, and the average precision by experts.

Precision of Hierarchy

We assess the precision of subsumption relations making up the ontology hierarchy. Relations are extracted using the maximum spanning tree over the graph represented by the *Ontology* component, π_O (see Section 5.2.1 for details), resulting in 49 subsumption relations. We compare their quality to that of 49 randomly sampled subsumption relations. Subsumptions are presented to the worker using *is a* patterns, similar to the ones described above for concept evaluation, however in this case, the concept tokens are the top tokens of the hypernym topic. An example subsumption relation is

- [java, python, javascript, lists, ruby] **is a** [languages, language, features, objects, functions]

The results shown in Table 5.4 indicate a low precision among the extracted subsumption relations. This might be explained by the fact that at the final training iteration (2K) of the model, the perplexity of the *Ontology* component was still improving, while the perplexity of the other model components seemed closer to convergence. It is possible that the low precision observed here indicates that more training iterations are needed to achieve an accurate ontology using KB-LDA.

5.3.3 Overlap of KB-LDA topics with human-provided labels

We evaluated how well topics from KB-LDA correspond to document labels provided by humans, over a randomly sampled set of 40K documents from our corpus. In StackOverflow, questions (which we consider as documents) can be labeled with predefined tags.

Topic	string, character, characters, text, line
Tags	regex, string, python, php, ruby
Topic	element, div, css, elements, http
Tags	css, html, jquery, html5, javascript
Topic	table, query, database, sql, column
Tags	sql, mysql, database, performance, php
Topic	jquery, mysql, http, json, xml
Tags	jquery, json, javascript, ruby, string

Table 5.5: Top tags associated with sample topics.

Top Tags	Found in Dictionary	KB-LDA Docs	KB-LDA Tag	Frequent Tokens Docs	Frequent Tokens Tag
20	14	0.45	0.42	0.21	0.16
50	36	0.48	0.42	0.20	0.14
100	72	0.45	0.38	0.20	0.13
500	322	0.44	0.33	0.18	0.10

Table 5.6: Docs and Tag overlap of human-provided tags with KB-LDA topics, and frequent tokens.

Here, we estimate the overlap with the most frequently used tags. First, for topic k, we aggregate tags from documents where $k = \text{argmax}_{k'} \theta_d^{k'}$, where θ_d is the document topic distribution. Table 5.5 shows examples of the top tags associated with sample topics, indicating a good correlation between top topic words and the underlying concepts.

Next, for each tested document $d \in D$, let W_d be the top 30 words of the most probable topic in θ_d, and T_d the set of human provided document tags. We consider the following metrics:

$$\text{Docs-Overlap} = \frac{\sum_d^D \mathbb{1}_{\{\exists t \in T_d : t \in W_d\}}}{|D|} \tag{5.14}$$

measures the ratio of documents for which at least one tag overlaps with a top topic word.

Top 10 ranked triples: ⟨server, not found, error⟩, ⟨user, can access, file⟩, ⟨method, not found, error⟩, ⟨user, can change, password⟩, ⟨page, not found, error⟩, ⟨user, can upload, videos⟩, ⟨compiler, will generate, error⟩, ⟨users, can upload, files⟩, ⟨users, can upload, files⟩, ⟨object, not found, error⟩
Bottom 10 ranked triples: ⟨france, will visit, germany⟩, ⟨utilities, may include, heat⟩, ⟨iran, has had, russia⟩, ⟨russia, can stop, germany⟩, ⟨macs, do not support, windows media player⟩, ⟨cell phones, do not make, phone calls⟩, ⟨houses, have made, equipment⟩, ⟨guests, will find, restaurants⟩, ⟨guests, can request, bbq⟩, ⟨inspectors, do not make, appointments⟩

Table 5.7: Top and bottom ReVerb software triples ranked with KB-LDA (the tuple ⟨users, can upload, files⟩ is repeated in the data).

The average ratio of overlapping tags per document is

$$\text{Tag-Overlap} = \frac{1}{|D|}\sum_{d}^{D}\frac{|t : t \in T_d \wedge t \in W_d|}{|T_d|} \tag{5.15}$$

As a baseline, we measure similar overlap metrics using the 30 most frequent instance tokens in the document corpus. The results in Table 5.6 indicate an overlap of nearly half of the 20, 50, 100, and 500 most frequent tags with top topic tokens – significantly higher than the overlap with frequent token. Our evaluation is based on the subset of tags found in the instance dictionary of KB-LDA.

5.3.4 Extracting facts from an open IE resource

We use KB-LDA to extract domain specific triples from an existing open IE KB, the 15M relations extracted using ReVerb [Fader et al., 2011] from ClueWeb09. By extracting the relations in which the subject, verb and object noun phrases are included in the KB-LDA dictionary, we are left with under 5K triples, indicating the low coverage of software related triples using open domain extraction, in comparison with the 37K triples extracted from StackOverflow and given as an input to KB-LDA.

Due to word polysemy, many of the 5K extracted triples are themselves not specific to the domain. This suggests a hybrid approach in which KB-LDA is used to rank open IE triples for relevance to a domain. We ranked the 5K open triples by the probability of the

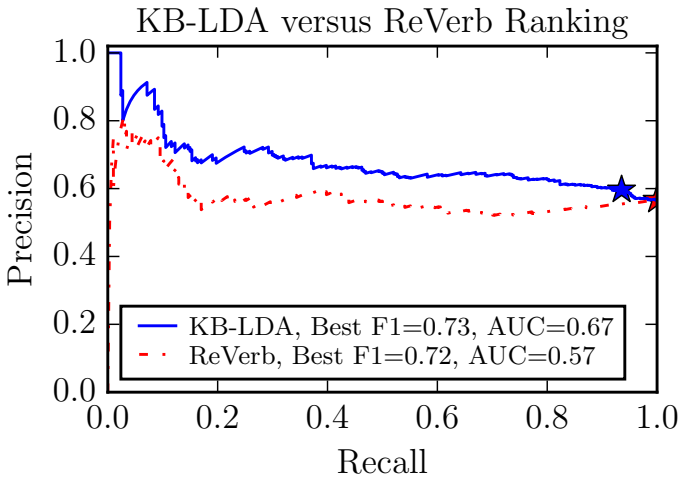


Figure 5.5: Precision-recall curves of rankers of open IE triples by software relevance, based on KB-LDA probabilities (blue), and ReVerb confidence (red). A star is pointing the highest F1.

triple given a trained KB-LDA model:

$$p(s,v,o) = \sum_{k_s}^{K} \sum_{k_v}^{K} \sum_{k_o}^{K} \pi_R^{\langle k_s,k_v,k_o \rangle} \sigma_{k_s}^{s} \sigma_{k_o}^{o} \delta_{k_v}^{v} \tag{5.16}$$

Table 5.7 shows the top and bottom 10 triples according to this ranking, which suggests that the triples ranked higher by KB-LDA are more relevant to the software domain.

We compare the ranking based on KB-LDA to a ranking using a confidence score for the triple as assigned by ReVerb. We manually labeled 500 of the triples according to their relevance to the software domain, and measured the precision and recall of the two rankings at any cutoff threshold. Figure 5.5 shows precision-recall curves for the two rankings, demonstrating that the ranking using probabilities based on KB-LDA leads to a more accurate detection of domain-relevant triples (with AUC of 0.67 for KB-LDA versus 0.57 for ReVerb).

5.4 Related Work

KB-LDA is an extension to LDA and link-LDA [Blei et al., 2003, Erosheva et al., 2004], modeling documents as a mixed membership over entity types with additional annotated

metadata, such as links [Nallapati et al., 2008, Chang and Blei, 2009]. It is a generalization of Block-LDA [Balasubramanyan and Cohen, 2011], however, KB-LDA models two link components, and the input links have a meaningful semantic correspondence to a KB structure (hierarchical and relational). In a related approach, Dalvi et al. [2012] cluster web table concepts to non-probabilistically create hierarchies with assigned concept names.

Our work is related to latent tensor representation of KBs, aimed at enhancing the ontological structure of existing KBs with relational data in the form of tensor structures. Nickel et al. [2012] factorized the ontology of Yago 2 for relational learning. A related approach was using Neural Tensor Networks to extract new facts from an existing KB [Chen et al., 2013, Socher et al., 2013]. In contrast, in KB-LDA, relational data is learned jointly with the model through the *Relations* component.

Statistical language models have recently been adapted for modeling software code and text documents. Most tasks focused on enhancing the software development workflow with code and comment completion [Hindle et al., 2012, Movshovitz-Attias and Cohen, 2013], learning coding conventions [Allamanis et al., 2014], and extracting actionable tasks from software documentation [Treude et al., 2014]. In related work, specific semantic relations, coordinate relations, have been extracted for a restricted class of software entities, ones that refer to Java classes [Movshovitz-Attias and Cohen, 2015b]. KB-LDA extends previous work by reasoning over a large variety of semantic relations among general software entities, as found in a document corpus.

5.5 Conclusions

We presented a model that jointly learns a latent ontological structure of a corpus augmented by relations, and identifies facts matching the learned structure. The quality of the produced structure was demonstrated through a series of real-world evaluations employing human judges, which measured the semantic coherence of instance topics, relations, topic concepts, and hierarchy. We further validated the semantic meaning of topic concepts, by their correspondence to an independent source of human-provided document tags. The experimental evaluation validates the usefulness of the proposed model for exploration of corpora from new domains, as we have demonstrated here for the software domain.

The results highlight the benefits of generalizing pattern-based facts (in this case, hypernym-hyponym pairs and subject-verb-object tuples), using text documents in a topic model framework. This modular approach offers opportunities to further improve an induced KB structure by posing additional constraints on corpus entities in the form of additional components to the model. As an example, in some domains, high-quality domain

resources exists, which include information that can be used to train additional components. Incorporating pre-existing domain information into the learning process has the potential to further improve the quality of the learned KB.

Chapter 6

Aligning Grounded and Learned Relations: A Comparison of Relations From a Grounded Corpus with a Topic-Model Guided Knowledge Base

In Chapter 5 we introduced KB-LDA, an unsupervised topic model for knowledge base construction. As a case study, we used the model to create a knowledge base for the software domain, for which the were no existing structured knowledge resources. In contrast, in the biomedical domain, many ontologies exist which describe sub-areas in the domain, including proteins, small chemical molecules and organism species. Grounding the KB learning process, by augmenting it with available domain resources such as biomedical ontologies, can potentially improve the learned structure. In order to estimate this potential, in this work, we investigate an alignment of relations emerging from biomedical ontologies with those learned from a corpus using KB-LDA.

We propose a method for extracting entity-to-entity relations from a corpus in which entities were annotated using state-of-the-art named-entity recognition systems, which integrate information from multiple known biomedical ontologies. Annotated entities in the corpus are grounded to specific entries in the source ontologies. We use KB-LDA to learn a KB over the same corpus, regardless of the given annotations, meaning that the learning process is *not* grounded. We then align relations found using the two methods, and we consider the following research questions:

- How well does KB-LDA learn relations and concepts found in existing ontologies?

- Does KB-LDA discover "new" relations and concepts, which were not annotated in the original corpus? Are the "new" discoveries correct?

- What relations or concepts are missing in the learned KB?

The answers to these questions hint at the potential of grounding the KB learning process, by combining corpus-based information with information from known ontologies.

6.1 Introduction

Computational advances in the past decade have lead to an abundance of publicly available biomedical ontologies, describing a large variety of biomedical entities. Many of the ontologies are manually crafted, or their generation has involved significant human effort in the form of labeling and annotation. The ontologies are stored and maintained in large repositories such as the NCBO (National Center for Biomedical Ontology) and HeTOP (Health Terminology/Ontology Portal), which also provide tools and web services for accessing this wealth of information. The challenge of incorporating information from the various available resources, has lead to efforts such as the CALBC (Collaborative Annotation of a Large Biomedical Corpus) [Rebholz-Schuhmann et al., 2010] which addresses the automatic integration of biomedical entity annotation based on a variety of popular named-entity recognition systems.

The available biomedical ontologies are largely narrow: they describe restricted subdomain entities (entities such as Genes, Organisms and Chemical Compounds are all described by distinct ontologies), and do not expand on cross-area entity relations. The BioNELL system [Movshovitz-Attias and Cohen, 2012a], described in Chapter 3, combines available biomedical ontologies and exploits seeds extracted from the same ontologies for improving a knowledge base population algorithm. However, the ontology merging method used in BioNELL is based solely on entity name matching. Importantly, it does not offer new cross-area relations, and the merged concepts do not reflect language statistics. For these reasons, it cannot be viewed as a true broad ontology for the biomedical domain. In contrast, in Chapter 5 we describe KB-LDA, an unsupervised topic model based system that learns a knowledge base of entities, including a hierarchical category schema and relations among categories, based on a target corpus [Movshovitz-Attias and Cohen, 2015a]. By building on corpus statistics, combined with relations extracted from the same corpus, KB-LDA learns a broad language-based ontology over entities found in domain text.

The system was initially evaluated on the task of automatically generating a KB for the software domain, for which a dedicated KB did not previously exist. It was additionally shown that the inferred KB is in-turn useful for extracting domain-specific information from open-domain resources. The software KB generated by KB-LDA was a first best-effort for this domain, and it was shown to produce high precision annotations and relations. However, the plethora of existing biomedical ontologies provide an opportunity for a more in-depth investigation of the capabilities of KB-LDA, in comparison with those existing resources.

In this chapter, we evaluate a biomedical KB created with KB-LDA, by aligning relations inferred by the learned KB to known biomedical ontologies. We rely on entity annotations from the CALBC corpus to extract entity-to-entity relations that have been validated by state-of-the-art named entity recognition systems. We then align the extracted relations with three types of potentially matching relations inferred by our learned KB: hierarchical relations, synonymous entity pairs, and general entity-to-entity relations, which emerge from the KB ontology, categories (represented by topics) and topic relations, respectively.

We demonstrate that many entity-to-entity relations found through CALBC are also found through KB-LDA. These span a wide variety of concepts from multiple biomedical sub-domains. Importantly, on top of discovering known concepts and relations, we find that KB-LDA also extracts new biomedical entities and suggests new relations between them. This means that while significant human effort went into creating existing biomedical ontologies, they are incomplete. To give an example, we find that current ontologies lack in describing terminology related to the scientific research process. We additionally describe newly discovered relations among known concepts, which have been found by our model but not in the CALBC relations. We conclude that models such as KB-LDA are useful for augmenting our existing knowledge, even in domains as well-researched as biomedicine. The results allude to a potential in models combining known relations, such as the ones found in existing ontologies or extracted from the CALBC corpus, with language-based relations, to reach a more complete mapping of this domain.

6.2 Data

Our data is based on the CALBC corpus, which contains 714K Medline immunology-related abstracts, and includes an annotation of the entities in the abstracts. The entities were automatically annotated using state-of-the-art named-entity recognition systems which attempt to integrate information from multiple biomedical ontologies. The annota-

tions have been crossed-referenced among several significant resources, including UMLS [Bodenreider, 2004], InterPro [Hunter et al., 2009], JoChem [Hettne et al., 2009], PIR Biothesaurus [Liu et al., 2006], and PIR iProClass [Wu et al., 2004].

We use this corpus as input to the KB-LDA pipeline, meaning that we extract relations from this corpus, and use it to build a biomedical knowledge base (see Section 6.3). A detailed analysis and statistics about the entities included in the corpus can be found in the experiments (see Section 6.4.1).

6.3 Methodology

Entities in the CALBC corpus are annotated with IDs which identify them with one of the following semantic groups: Proteins and genes, chemicals, diseases and disorders, and living beings. The ID codes are taken from terminology services such as the UMLS (Unified Medical Language System) described above. Each entity instance in the corpus is annotated with a set of IDs based on the harmonization of multiple alignment methods (the details of which can be found in [Rebholz-Schuhmann et al., 2010]). The objective of this annotation approach is to label entities based on their text token, their context, and a high level of agreement of the annotation according to multiple ontological sources.

An an example, instances of the entity 'lupus erythematosus', an autoimmune inflammatory disorder, are annotated throughout the corpus with a variety of IDs denoting various specializations of this disease (such as the Cutaneous, Subacute Cutaneous, Discoid and Systemic forms), the fact that this is an autoimmune disease, and that it is linked with cytomegaloviruses.

As a result of the annotation technique used in this corpus, many entities in the data are annotated with the IDs of their hyponyms (e.g., 'lupus erythematosus' is annotated with the ID of 'cutaneous lupus erythematosus') or hypernyms (e.g., 'lupus erythematosus' is annotated with the ID of 'autoimmune disease'). Indeed, by linking each pair of annotated entities that share an ID annotation, we arrive at a noisy ontology such as the one depicted in Figure 6.1. This noisy ontology combines information drawn from the biomedical ontologies used to annotate the corpus, and it represents the linking provided by the employed named-entity recognition systems used for the annotation process. We align relations from this ontology to relations inferred using KB-LDA.

We refer to a pair of entities that share an annotation ID as *matching*, and we formalize this relation by the following boolean *match* functions among the entities e_1 and e_2, or

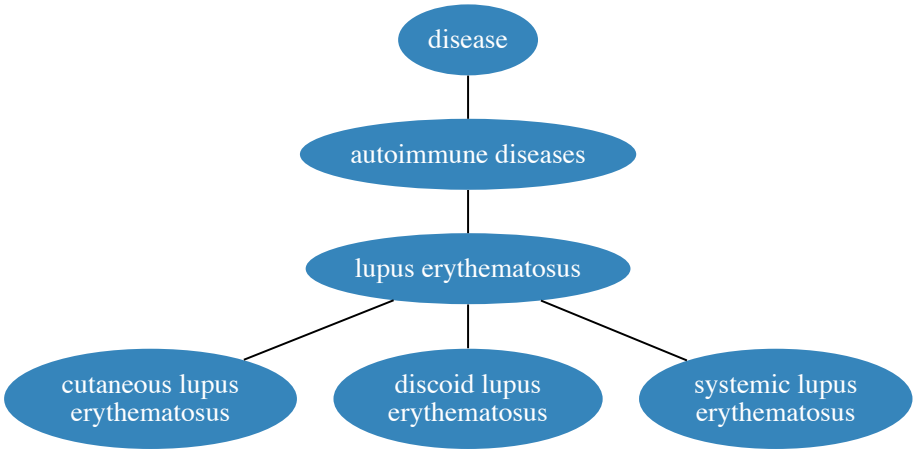


Figure 6.1: Fragment of CALBC ID-based ontology. This ontology is the product of connecting annotated entities that share annotation IDs.

among the entity e_1 and the entity set E_2:

$$\text{match}(e_1, e_2) = \{\exists i \in \text{ID}_{e_1} | i \in \text{ID}_{e_2}\} \quad (6.1)$$

$$\text{match}(e_1, E_2) = \{\exists i \in \text{ID}_{e_1} \wedge \exists e_2 \in E_2 | i \in \text{ID}_{e_2}\} \quad (6.2)$$

It is worth noting that the *match* relation is not strictly a hypernym-hyponym relation, but rather it is not well defined under this hierarchy. Matching entities can also be synonyms (e.g., 'disease' and 'illness'), abbreviations (e.g., 'cutaneous lupus erythematosus', and 'CLE'), or hold some other unspecified relation (for example, the term 'phosphate diabetes', a condition marked by excessive phosphate in the urine, matches multiple phosphate variants including 'glucose-6-phosphate', 'fructose-1-phosphate', and 'triose phosphate isomerase'). Moreover, the *match* relation is symmetric, whereas relations such as hypernym-hyponym are not. Despite these caveats, we consider the CALBC ontology induced by the *match* relation graph as a target ontology, and we compare existing *match* relations from this ontology with the relations induced by the KB-LDA model predictions.

We construct a KB over biomedical entities that appear in relations extracted directly from the corpus. We use the implementation of the KB-LDA model (described in detail in Chapter 5, [Movshovitz-Attias and Cohen, 2015a]), to construct a set of topics representing clusters of the entities, a hierarchical ontology over the topics and relations among entity and verb topics. Our relation extraction process does not consider the CALBC annotations, but rather, the entities that we consider are noun phrases that are found in relations extracted by the KB-LDA pipeline. KB-LDA is based on a topic modeling approach,

related to the LDA model [Blei et al., 2003], and it contains three learning components which share topics and are optimized jointly. It infers topics over noun phrases (termed 'instance topics') and verb phrases (termed 'relation topics') extracted from a corpus. The *Ontology* component is used to learn a hierarchical structure between instance topics. It is trained over hypernym-hyponym examples extracted from the input corpus, and it learns the latent mixed membership of noun entities in the input to topics, as well as the topic to topic relations induced by the input examples. The *Relations* component learns relations between noun and relation topics and it does so using input data extracted using subject-verb-object patterns from the corpus. Finally, the *Documents* component is an extension of link-LDA [Erosheva et al., 2004] which brings in corpus statistics about the co-occurrence of noun and verb phrases directly in the text documents.

Next, we evaluate the alignment of the knowledge base induced by the KB-LDA model with the CALBC ontology. We first explore the differences between the entities annotated in CALBC and those extracted by KB-LDA. Then, we examine the relations emerging from of each of the KB-LDA components, and compare them with the relations found in the CALBC ontology.

6.4 Experimental results

In this section we explore the differences between the entities and relations emerging from the CALBC ontology and those learned by KB-LDA. Through this comparison we answer the following questions:

- Does KB-LDA learn entities and relations that are also found in the CALBC ontology? Such examples indicate a validation of the model results.

- Does KB-LDA discover "new" entities and relations that are not found in the CALBC ontology? If the new discoveries are correct, this indicates the added value of learning from language statistics.

- Which entities and relation that are found in the CALBC ontology are missing from what we learn with KB-LDA? Here lies the potential of grounding the KB learning model, since the missing entities and relations could be directly added to the KB, and also influence the learning of additional new ones.

Incorrect Annotations from CALBC
bro*nchial asthma*
pn*eumoniae*
beta-1,2-*mannotriose*
d*ermatophagoid*es
colore*ctal tumo*r

Table 6.1: Partial or incorrect annotations found in the CALBC corpus. Each row contains an entity that has been partially or otherwise incorrectly annotated in the corpus. The characters that have been annotated are marked in red and italicized.

6.4.1 Entity Analysis

The CALBC corpus includes 69,618 distinct annotated entities, while we extract 712,131 entities with the KB-LDA relation extraction pipeline, with 22,418 entities common to both systems. As described above, the common entities validate the extractions made by KB-LDA. Next, we evaluate the entities that are unique to each system.

First, we examine the 47,200 entities that are annotated in the corpus but are not extracted by KB-LDA. We randomly sample 100 of those entities and manually evaluate them. We found that 35 of the sampled entities included partial or incorrect annotations, including the examples shown in Table 6.1. As an example, the annotation 'nchial asthma' is a mis-parse of the entity 'bronchial asthma', which has been correctly extracted by KB-LDA. Out of the 35 incorrectly annotated entities, 20 are only annotated in the corpus in an incorrect form. Overall, the sampled entities had a low frequency in the corpus (their average frequency is 4.87), which means they are less likely to appear in meaningful relations in the corpus, and therefore also less likely to be extracted by KB-LDA. Out of the sampled entities, we did however find 12 entities that had a frequency higher than 10, and these indicate that there is a potential to incorporating grounded information (such as the grounded entities and relations found in this corpus) directly in the KB learning process.

The KB-LDA pipeline recognized a total of 712,131 entities that appear in any concept-instance or subject-verb-object relation in the corpus. Out of those, only 25,981 entities are found in frequent relations (which appear more than twice) and are therefore used in training the KB-LDA model. 5,498 of these entities are annotated in the CALBC corpus, leaving 20,483 entities that are extracted by KB-LDA, found in frequent relations in the corpus, and yet are not annotated. We randomly sample 100 of those entities and manually evaluate them. We found 97 of the 100 entities to be correct and useful entities, which

Biological Entities and Processes	Experimental Terminology
linkage	techniques
leukotoxin	samples
chemotactic response	evidence
plasma cell-associated markers	experiments
bip-homologues	authors
eosinophils	exposure
dendritic cells	study
secretion	specimens
pro-inflammatory cytokines	hapten inhibition experiments
axons	sodium dodecyl sulfate-polyacrylamide gel

Table 6.2: New entities discovered by KB-LDA. We include a sample of entities that were extracted by KB-LDA an found in frequent relations, and yet were not annotated in the corpus. The discovered entities are binned in one of the following categories: Biological entities and processes (left table), and Experimental terminology (right table). See more details in section 6.4.1.

should reasonably be included in a biomedical KB (see examples in Table 6.2). The newly discovered entities indicate the added value of learning from language statistics. We additionally identified 2033 verb phrases appearing in frequent subject-verb-object relations in the corpus, which are used by the KB-LDA model (the CALBC annotation do not include any verb phrases).

The entities discovered by KB-LDA can be binned into two high-level categories: 'Biological Entities and Processes' (including 81 out of the 100 sampled entities) or 'Experimental Terminology' (16 out of 100). The former group includes terms such as *linkage*, *leukotoxin*, *plasma cell-associated markers*, which are more likely to be found in the type of biomedical ontologies that were used to annotated the corpus, and so it is more surprising that they were not recognized by the NER tagging process. The latter group includes terms that describe experimental research processes, rather than the biological entities on which they operate. Some examples include the terms *techniques*, *samples*, but also more specific types of experiments (e.g., *hapten inhibition experiments*) and lab equipment or resources (e.g., *sodium dodecyl sulfate-polyacrylamide gel*). While some ontology might exist which describes this type of terminology, it is not known to us, and no such ontology was used to annotate the CALBC corpus; this means that entities describing experimental terminology are not expected to be annotated and will not appear in the relations extracted

from the corpus.

6.4.2 Relation Analysis

We use the KB-LDA model (described in detail in Chapter 5, [Movshovitz-Attias and Cohen, 2015a]) to construct a KB of biomedical entities, with a set of topics representing a clustering of the entities, an inferred hierarchical ontology and relations among entity topics. In the following sections we compare *match* relations from the CALBC ontology with the relations induced by the outcome of the KB-LDA model, namely: hierarchical subsumption relations among topics (Section 6.4.3), intra-topic relations (Section 6.4.4), and relations between subject, verb, and object topics (Section 6.4.5). Note that our KB contains both entities annotated by the source corpus as well as new entities identified by our pipeline (see Section 6.4.1). Therefore, only relations among annotated entities can be recovered with *match* relations. Below we also note the proportion of "new" relations discovered by our KB that did not exist in the original ontology. These are relations between pairs of new entities, or a new and an existing entity.

In each analysis below we compare multiple KBs created with KB-LDA. For each group we describe the overlap of the discovered relations in that KB with the CALBC ontology. The KBs we compare are based on:

Number of topics: We tested models using 20, 50 and 100 topics.

Full versus sampled corpus: In all of the KBs presented in this section, the *Documents* component of KB-LDA has been trained on a sample of 50,000 documents (around 7% of the data). In this analysis, however, we compare a KB with 50 topics trained on the sampled corpus versus the full corpus.

Ablation models: We compare the full KB-LDA model, with ablated models that are missing one or two of the three main model components: *Ontology*, *Relations*, or *Documents*. For each analysis below, we describe the result on the subset of ablated models that are relevant for the evaluated relations, compared with the full model.

6.4.3 Ontology Coherence

The hierarchical ontology induced by KB-LDA suggests a set of subsumption relations among noun entity topics. In a learned KB with n topics, the induced ontology is composed of $n-1$ subsumption relations among topic pairs, selected according to the maximal

Topic	Top words in topic
Super-topic	diseases, factors, disorders, conditions, pathogens
Sub-topic	cancer, rheumatoid arthritis, hiv, multiple sclerosis, diabetes

Table 6.3: Example subsumption relation.

spanning tree over the graph of topic to topic relations (for further details on the ontology creation process, see Chapter 5, Section 5.2.1) . As an example, Table 6.3 shows the top entities of two topics that were found in a subsumption relation according to KB-LDA in one of the evaluated KBs. The super-topic contains super-concepts or hypernyms, and the sub-topic contains sub-concepts or hyponyms. This induced relation implies that each entity from the sub-topic can be matched to at least one entity from the super-topic.

We can measure the precision of an induced KB ontology by verifying that the implied subsumptions among entity pairs are also found in the CALBC ontology. For each ⟨super-topic, sub-topic⟩ pair, we examine the top 30 entities in each topic. For each entity in the sub-topic, we search for at least one matching entity in the super-topic, according to the *match* relation defined in Equation 6.1, meaning that the relation between the two entities is validated in a known biomedical ontology. The overlap of each subsumption relation with the CALBC ontology is then measured by the proportion of sub-topic entities that were matched with some super-concept entity. Finally, we summarize the precision of the ontology induced by a model based on the average overlap of the set of subsumption relations S making up the maximum spanning tree for that model. We report this precision over each of the top N entities ($N \in \{1, \ldots, 30\}$):

$$\text{Ontology-Precision}(N) = \frac{1}{|S|} \sum_{s \in S} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\text{match}(\text{Instances}(S_i), \text{Concepts})} \tag{6.3}$$

where Instances(S_i) is the top i-th entity associated with the sub-topic of subsumption relation S, and Concepts are all top instances associated with the super-topic of this subsumption relation.

Known versus Suggested Relations

Figure 6.2 examines the precision of the ontology of a KB learned over 50 topics, by evaluating the relations inferred by this ontology. The relations in the dark blue area, marked as 'Known Relations', were matched with the CALBC ontology as described above. The

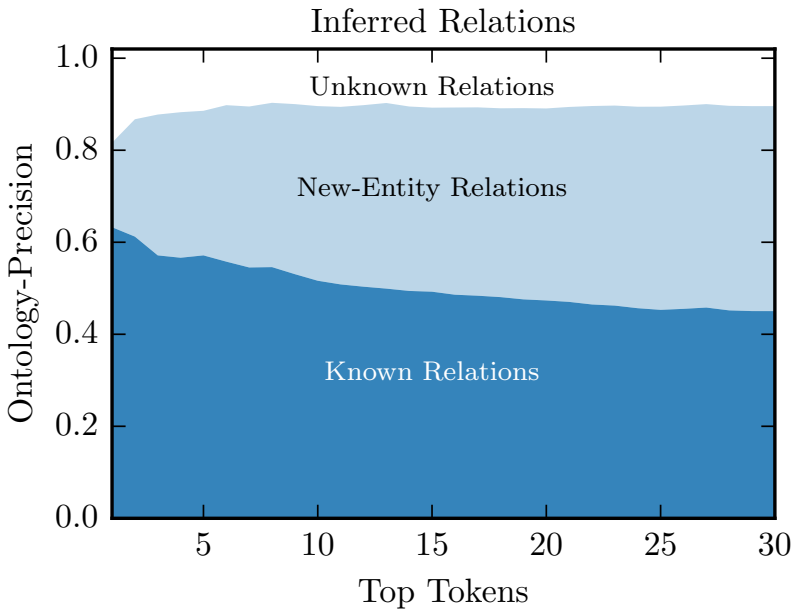


Figure 6.2: Relations inferred by an ontology learned over 50 topics.

rest of the relations were not matched and their precision was therefore manually evaluated. The relations in the light blue area, marked as 'New-Entity Relations', could not be matched with the CALBC ontology using the *match* function in Equation 6.3 since either the evaluated instance entity or all concept entities were new, i.e., they were extracted by KB-LDA and were never annotated in the CALBC corpus. We manually evaluated a random sample of 50 relations from this group and found 62% to be correct.

As described above, the newly extracted terms can be binned into two categories: 'Biological Entities and Processes' or 'Experimental Terminology'. Entities that describe experimental terminology are not expected to be annotated in the corpus, or be found in any of the grounded relations, since no ontology which includes experimental terminology was used in annotating the CALBC corpus (see Section 6.4.1 and Table 6.2). Moreover, we would expect that terms from these two categories would ideally belong in two disjoint ontologies. However, in KB-LDA, we enforce a single ontology by extracting the maximum spanning tree over the multinomial of the *Ontology* component. The resulting ontology, therefore, necessarily includes subsumption relations between 'experimental' topics (which describe experimental terminology) and 'biological' topics (which describe biomedical entities and processes), even if these are low probability relations, simply since a single structure is enforced. If we consider only the 31 out of 50 predicted relations in the

sample above that include non-experimental instances, we find that 90% of those are correct. Indeed, many of the incorrect predictions are due to enforced subsumptions between 'experimental' and 'biological' topics.

Finally, relations in the white region of Figure 6.2, marked as 'Unknown Relations', are relations between instances and concepts that are annotated in the corpus but do not share an annotated ID. These relations are predicted by our model, yet are unknown according to the CALBC annotations. We manually evaluated a random sample of 50 unknown relations and found that 54% are correct. For example, our model found a relation between the concept 'antibody' and the instances 'ema' and 'dcs', which abbreviate the antibodies 'Endomysial autoantibodies' and 'Anti-Cyclin D1 antibody', also known as 'dcs-6'. It also correctly predicted that the bone marrow is a type of tissue, a fact that was missing in the CALBC ontology. Interestingly, the model incorrectly predicted that 'airway' is a type of 'organ', and that 'killer' is a type of 'cancer' or 'disease'.

Comparison by Number of Topics

KB-LDA generates a hierarchy over a pre-defined number of topics. This number reflects the "resolution" of the induced ontology, i.e., a higher number of learned topics can potentially result in an ontology with more specialized topics, and therefore reflect a more diverse set of latent subsumptions hidden in the corpus. Alternatively, a high number of topics may result in similar and redundant topics. Finding an optimal number of topics for a topic modeling approach given a corpus largely remains an open question, as many related methods, including Latent Dirichlet Allocation [Blei et al., 2003], Probabilistic Latent Semantic Indexing [Hofmann, 1999], and Non-Negative Matrix Factorization [Lee and Seung, 1999], assume that this number is pre-defined.

Here, we evaluate precision of ontologies created based on 20, 50 and 100 topics, as measured by their inferred known and suggested relations. Figure 6.3 compares those ontologies, where solid lines represent the amount of discovered known relations, and dashed lines represent additional suggested relations among new entities. The 19 subsumption relations in the ontology of the model trained over 20 topics includes the highest proportions of previously known relations, with a precision ranging between 63%-51% over the top 30 topic tokens. The 50 topic model, on the other hand, creates a larger hierarchy with 49 subsumption relations with lower precision of known relations (between 63%-45%). This is likely due to the fact that the larger number of topics, now includes significantly more new entities propagated to the evaluated top 30 tokens, leading to a higher proportion of new suggested relations compared with the 20 topic model. As noted above, suggested relations among biological entities was estimated at 90% according to our evaluation. The

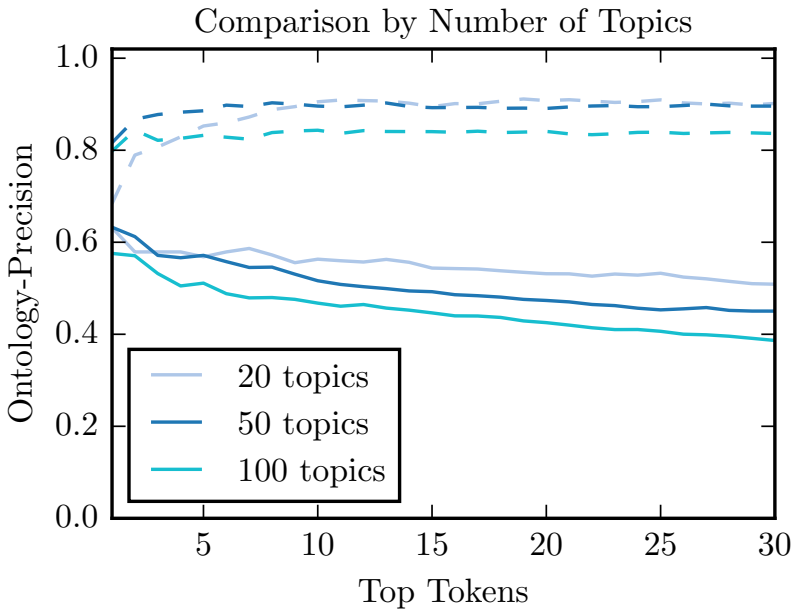


Figure 6.3: Comparison of ontologies learned by models over 20, 50, or 100 topics.

total proportions of known relations and ones suggested among new entities is similar for the 20 and 50 topic models, at around 90%, leading to an estimated similar estimate of overall precision. Surprisingly, the 100 topic model produces less known relations and also less suggested relations among new entities (in proportion to the size of the ontology), resulting in lower precision in comparison to the other models.

Comparison with Ablated Models

We have previously evaluated the precision of the ontology created by a complete KB-LDA model. However, the model is composed of three components which jointly optimize a topic hierarchy, relations among those topics, and background statistics from a documents corpus. Figure 6.4 (left) compares the precision of ontologies created by the full KB-LDA model, and by ablated models, missing the *Relations* component, the *Documents* component, or both. In this experiment, we are not able to omit the *Ontology* component, since the topic hierarchy is created using the multinomial π_O which is learned using that component. All of the plotted experiments measure the precision according to Equation 6.3, and they are all learned over 50 topics. Surprisingly, the ablated model optimizing only the topic hierarchy (marked 'ablation: Ont'), has the lowest precision, though

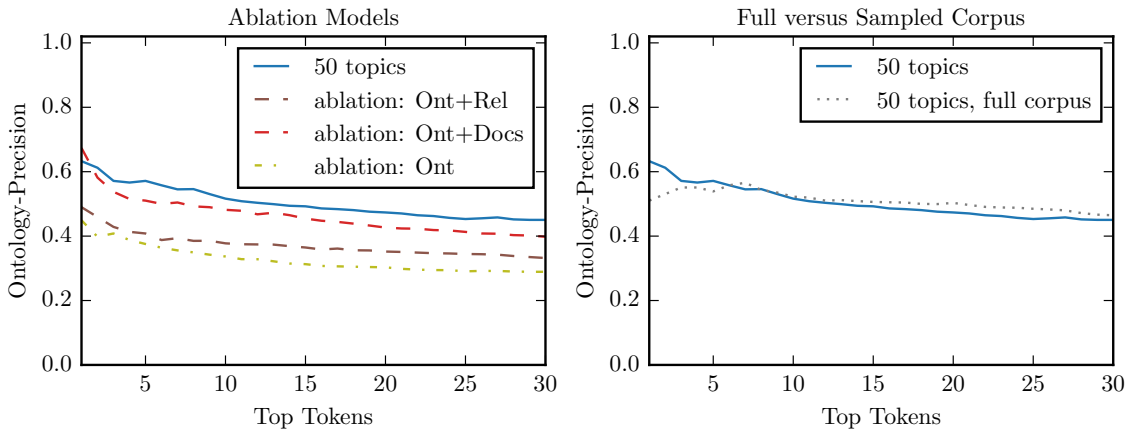


Figure 6.4: Left: Comparison of ontologies created by the full KB-LDA model, and by ablated models, missing the *Relations* component, the *Documents* component, or both. Each depicted ablation model lists the components that were included in that model. Right: Comparison of ontologies created based on the complete corpus versus a sample of 50K documents.

the ontology is solely directly optimized in this case. Both the corpus- and relation-based statistics have a positive effect on the precision of the ontology, but the complete model with all three components achieves the highest performance.

Full versus Sampled Corpus

We compare the ontologies of a model whose *Documents* component is trained on the full available corpus ('50 topics, full corpus'), versus a sample of 50K documents ('50 topics'). For both models, the *Ontology* and *Relations* components are trained on hypernym-hyponym and subject-verb-object data extracted from the complete corpus. Figure 6.4 (right) shows that the information coming from an additional 665K documents does not dramatically change the precision of the induced ontology over known subsumption relations. This result is especially surprising since the addition of the *Documents* component with the initial sample of 50K documents had a significant effect on ontology precision, as seen in the ablation model analysis.

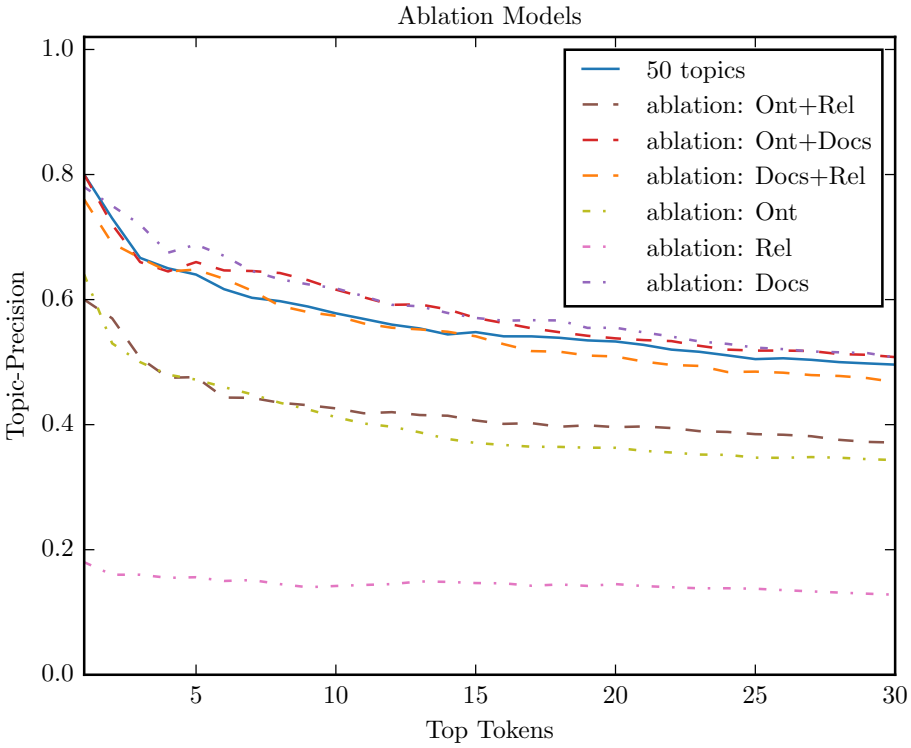


Figure 6.5: Comparison of topic precision of the full KB-LDA models and ablated models using only one or two of the main components: *Ontology*, *Relations*, and *Documents*. Each depicted ablation model lists the components that were included in that model.

6.4.4 Topic Coherence

In a similar analysis to the ontology coherence described above, in this section we examine the inner-topic coherence of topics produced by KB-LDA. We consider two entities as related if they have been clustered by the model into the same topic, and we attempt to verify the existence of this relation in the CALBC-induced ontology. As mentioned above, the *match* relation is symmetric and is often found between synonymous entities, and entities and their abbreviations. It seems reasonable, then, to measure inner-topic coherence using this relation. We therefore measure topic precision by matching all pairs of entities within a topic, over each of the top N entities ($N \in \{1, \ldots, 30\}$):

$$\text{Topic-Precision}(N) = \frac{1}{|T|} \sum_{t \in T} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\text{match}(\text{Tokens}(t_i), \text{Tokens}(t_{\neg i}))} \tag{6.4}$$

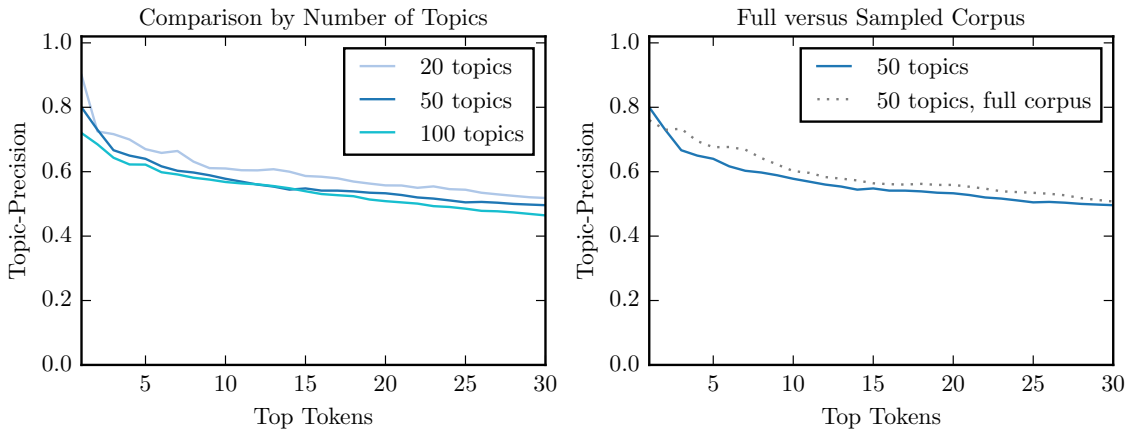


Figure 6.6: Comparison of topic precision by the number of models used during training (left), and of models created with a complete corpus versus a sample of documents (right).

where Tokens(t_i) is the i-th top token of topic t, and Tokens($t_{\neg i}$), are the top 30 tokens of topic t, excluding the i-th token.

Figure 6.5 shows a comparison of the topic precision of the full KB-LDA model with ablation models, using only one or two of the three main components: *Ontology*, *Relations*, and *Documents*. In this analysis, we are able to include all ablated variations, as they all produce concept and instance topics. The results indicate that the performance of the topics is mainly contributed by corpus statistics, used to train the *Documents* component. All the ablated models that contain this component have similar performance, with a significant decrease in the models that are missing it. Importantly, the performance of the full model does not suffer greatly from the additional components required for creating a complete KB schema.

In Figure 6.6 (left) we examine the performance of topic precision in models trained on 20, 50 and 100 topics, with the 20 topic model showing slightly higher performance, relative to the others. On the right hand side of this figure, we see a comparison of a model trained with a sampled corpus versus a full corpus, and, as in the case of ontology precision, training on a sample of the corpus had only a modest effect on precision.

6.4.5 Relation Coherence

In this section, we evaluate the precision of the relations induced from KB-LDA between subject and object instance topics, through verb topics. The *Relations* component of the model is trained on SVO triples extracted from the corpus, then using the multinomial π_R we are able to extract the top inferred relations among instance and verb topics. Here we test whether these relation can be matched with the CALBC ontology. The *match* relations in the CALBC ontology do not specify a verb that represents the type of relation among the two matched entities, but rather only that such a relation exist, and in previous section we have considered it to be a potential subsumption or synonymy relation. Here we treat the *match* function as a general relation. We extract pairs of subject and object entities from the corresponding topics in the top relations inferred by a learned model, and we measure the precision of each relation based on whether the pairs can be found in the CALBC ontology, in the following two ways:

$$\text{Relation-Precision Subject-To-Object}(N) = \frac{1}{|R|} \sum_{r=(s,v,o)\in R} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\text{match}(o_i,s)} \tag{6.5}$$

$$\text{Relation-Precision Object-To-Subject}(N) = \frac{1}{|R|} \sum_{r=(s,v,o)\in R} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\text{match}(s_i,o)} \tag{6.6}$$

where s and o are the sets of top entities of the subject and object topics in the relation r, and the analysis is done over the top N entities ($N \in \{1, \ldots, 30\}$).

Figure 6.7 shows results comparing the relation precision of various learned models. Similar to the case of ontology learning, the results indicate that optimizing the relations component detached from corpus statistics is not sufficient, and this ablated model suffers from very low precision (around 20%, top row). However, while the addition of hypernym-hyponym relations has minimal effect on the *Relations*-only model, it also significantly decreases the performance of the *Documents+Relations* model. This means that the hierarchical constraints imposed by the hypernym-hyponym data are less consistent with some of the relations that can otherwise be inferred by the model.

As observed for topic and subsumption relations, learning less topics seems to propagate the more reliable entities to the top of each topic, resulting in a higher proportion (but not a higher absolute number) of known relations discovered by the model (Figure 6.7, middle row). In a stark contrast to topic and subsumption relations, learning general relations from a sampled corpus results in a dramatic decrease of around 30% in precision results (Figure 6.7, bottom row). In all metrics described above, we observe similar results whether a subject entity is matched against all object entities or the other way around.

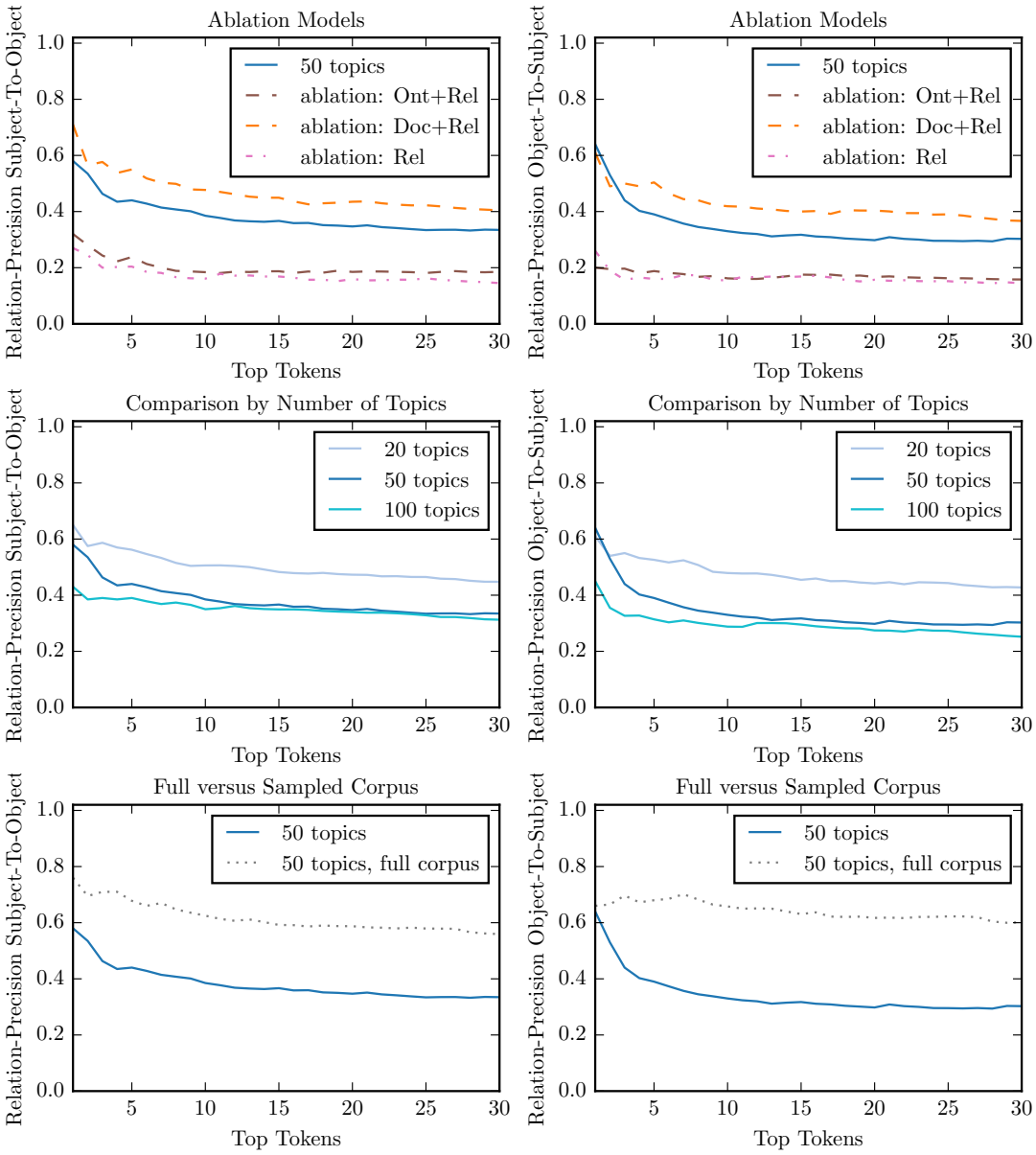


Figure 6.7: Relation precision. Precision is measured from Subject-to-Object (left figures) or Object-to-Subject (right). Top: Comparison between full and ablation models. Middle: Comparison by number of topics. Bottom: Comparison of models trained using full corpus statistics or a sample of 50K documents.

6.5 Related Work

Topics generated by LDA-inspired topic models have traditionally been evaluated based on perplexity. In this case, a model is trained using a training corpus, and then evaluated by measuring the log probability of an unseen test corpus. Perplexity is a standard way to select model parameters, such as the number of topics, and is therefore often used for model selection and comparison. Wallach et al. [2009] present efficient methods for estimating probabilities on held-out documents, and discuss the disadvantages and biases in model comparison based on several standard evaluation methods, including the harmonic mean method [Newton and Raftery, 1994], simple and annealed importance sampling [Neal, 2001].

KB-LDA was evaluated in [Movshovitz-Attias and Cohen, 2015a] by a series of Mechanical Turk evaluations for topic, ontology and relation coherence, following a study by Chang et al. [2009] that presented counter-intuitive results of *semantic* topic evaluation, in which humans preferred models with higher perplexity. These results, in turn, lead to other evaluation methods that rely on topic semantics. For example, Newman et al. [2010] suggest that users prefer topics whose words closely co-occur in a background corpus (such as in Wikipedia or in Google search results; [Newman et al., 2009b]), where term co-occurrence is measured using pointwise mutual information [Pecina, 2010].

The human evaluation of KB-LDA showed that it generates a semantic structure that is compelling to humans. In this work, however, we consider the KB learned by the model as a collection of inferred relations between pairs of entities, which are evaluated in terms of overlap with a reference set of relations. In this sense, our work is more closely related to ontology evaluation methods, with the caveat that our model generates three types of relations: hypernym-hyponym relations, synonymous/inner-topic relations, and general relations as expressed be extracted verb phrases.

Brank et al. [2005] and Staab and Studer [2013] survey ontology evaluation approaches in the context of four categories. The first is evaluation by humans who try to asses predefined criteria regarding the ontology [Lozano-Tello and Gómez-Pérez, 2004]. The second approach uses the ontology in a downstream application and evaluates the results [Porzel and Malaka, 2004]. These approaches have been used in [Movshovitz-Attias and Cohen, 2015a] for evaluating relations inferred from KB-LDA. A third approach measures how well the ontology covers a source of domain-relevant data or documents (as in [Brewster et al., 2004]), and finally, the ontology may be compared against a gold-standard resource [Maedche and Staab, 2002]. Our work combines the latter two approaches by comparing learned relations to an ontology extracted from a corpus, where entities were annotated based on domain-ontologies. Our work is also related to semantic integration

and the task of merging ontologies, surveyed for example by Noy [2004].

6.6 Conclusions

An abundance of publicly available biomedical ontologies provides an excellent starting ground for construction of knowledge bases for this domain. They describe a wealth of interest areas within the domain, and they were created by a combination of significant human effort and semi-automation. Here, we consider the CALBC corpus, that was annotated using entities from several prominent biomedical ontologies; we use it as a source of grounded entities and relations, which we compare with ones automatically learned by our KB construction methods, KB-LDA.

We show that much of what is learned by KB-LDA are entities and relations that are also found in the biomedical ontologies, which verifies our learning model. On top of that, we find that KB-LDA discovers new entities and relations that were not found in the biomedical ontologies. Our model extracts terms that describe biological entities and process, which we expected to be present in biomedical ontologies, and yet were annotated in the corpus. Additionally, we extract a a class of entities which describe experimental terminology related to the biomedical research process. This terminology is well represented in the corpus and in our learned KB, and yet no experimental terms were annotated in the corpus. Overall, the high precision observed among the new discoveries inferred from the model suggests that there is added value in learning from language statistics to augment our existing domain-knowledge, even in domains as well-researched as biomedicine.

Finally, we observe some examples of entities and relations that are found in biomedical ontologies and are annotated in the CALBC corpus, but are not learned by KB-LDA. Our evaluation indicates that these examples include some entities that frequently occur in the data, and are therefore important to extract, and might participate in significant biomedical relations. This gap between known ontologies and what was learned by our model suggests a potential for grounding. As an example, the modularity of the KB-LDA learning framework enables the addition of learning components that can be trained using known ontologies. The results indicate that this type of modification can improve the final learned KB.

Chapter 7

Conclusion

7.1 Summary

The key insight behind the work presented in this thesis is that grounding, the process of linking an individual word token to the real-world entity it represents, has the potential to advance statistical modeling approaches for knowledge base construction. This is especially important, and seems attainable, in scientific domains where high-quality grounding data is available. This idea extends a growing body of work in continuous vector-space modeling, in which continuous representations of words, documents and phrases are used to model language. As an example, latent Dirichlet allocation models use a mapping of words and documents into a continuous topic space. In this work, vector statistics are drawn not only directly from text-based corpora but are also extended by statistics from domain-specific resources.

We explore modifications to well-known statistical models in the context of learning about technical domains, and we learn that encoding knowledge of the domain into our models leads to improved predictions. We design unsupervised and semi-supervised methods for domain-specific relation discovery and for knowledge base population and construction. Using our knowledge base construction approach we build KBs for two technical domains and show that they learn concepts and relations that are compelling to humans, and are validated by existing known information on the domain, where such information exists.

The work done here suggests that there is a potential for improved KB construction by grounding the learning process, or by including weak human-supervision, with detailed suggestions included in the following sections. In this thesis, we have often explored

downstream tasks to evaluate, or indeed motivate, better understanding and structuring of domain-knowledge, but a more in-depth look at improving applications using technical-domain knowledge bases deserves future research. Below are additional thoughts, lessons, and ideas for future directions that follow from this work.

7.2 Closing Thoughts

7.2.1 Limitations of Grounding

In Chapter 4 we proposed a method for building a grounded software ontology, representing Java class entities. Links in the ontology were optimized based on both the text- and code-based similarity of pairs of classes. This approach highlights a limitation of grounding: a code-based inferred ontology can only be composed of entities that can be found directly in the code, such as methods, classes and types; however, it lacks higher-level software concepts like programming paradigms, parallels between different programming languages, and programming-supporting tools (e.g., IDEs).

In the biomedical domain, there is abundant experimental data on protein structure, interaction, regulation, and so on. For example, in Chapter 3 we used a resource of protein names of the fruit-fly. Similarly, sequencing data provides in-depth insights into DNA, RNA and related replication mechanisms. However, as in the software domain, there is a variety of biomedical concepts beyond DNA and its products, proteins, that are not as richly described in data available for grounding.

Grounded language application that rely on a grounding resource may, therefore, learn ontologies and knowledge bases that are unbalanced in their representation of the target domain. One solution to this problem is to use a combined approach, which draws information from text and is refined based on additional available resources, such as the method described briefly in Section 7.3.2. This approach is supported by the results described in Chapter 6, which show that direct corpus statistics contribute significantly to learning all types of relations. In comparison, baseline models that learn only from extracted relations (which could, in theory, be extracted based on grounded resources) are not as well performing on their own.

7.2.2 Evaluating Knowledge Base Systems

A great frustration of this PhD work, and indeed much effort, has revolved around the task of evaluating predicted concepts, relations, and other structured and formated knowledge predicted by the proposed algorithms. Consider, for example, an evaluation of the assignment of an entity to a category or concept, such as "Is Pittsburgh a City?". A seemingly simple task is revealed as challenging in the following scenarios: (1) Many of the facts predicted by our models, specifically ones drawing from scientific, or otherwise specialized domains, require considerable prior knowledge for this type of evaluation. Can you answer the following question: "Is Indy a gene of the Fruit Fly"? The answer is 'yes', it is short for *I'm not dead yet*, and mutant variations of this gene have doubled the average life span of the organism in a controversial experiment. (2) As eluded by the previous example, some of the information extracted from our source corpora may not be accurate, and yet it may be repeated often enough that our models confidently propagate it. This phenomenon is more often encountered when considering Web and user-generated data. Whether or not a commonly repeated, yet factually incorrect, fact should be treated as correct is surely open for debate, and may depend on the downstream application. An advantage, however, of grounding entities from a scientific domain, is that the grounded data may come from authoritative resources and be more reliable. Finally, (3) the information extracted by NLP models encompasses the full variety of subtleties and complications capable by creative human brains. Ambiguity, subjectivity, temporal and local variations, political affiliations and other biases make coherent evaluations rare. Some of these challenges are discussed at length in [Movshovitz-Attias et al., 2015], and they have been a subject of interest throughout this work.

In practice, knowledge bases are often evaluated by a combination of manual labeling, qualitative analysis of the extracted information, and the capability of the KB in improving downstream applications, and we have experimented with all of these strategies in our work. In Chapter 5 we additionally harnessed the power of the crowds for an extensive evaluation. We identified a group of Mechanical Turk workers with knowledge of software terminology and concepts, and were assisted by them in evaluating a software knowledge base. The automation and expansion of the evaluation process for Web-based and KB predictions, remains a challenge for this research area.

7.2.3 The Distinction Between Grounding and Semi-Supervision

Semi-supervised learning can be confused with grounding in some cases. Biomedical ontologies, such as the ones we have used in Chapters 3 and 6, are (mainly) created

by humans. Therefore, grounding an algorithm with a human-curated ontology can also be thought of as adding some amount of human supervision. However, this is not the case when grounding with a code repository, since it does not contain and direct human labeling, but rather implicit statistics that are similar to ones extracted from text corpora. Note that while grounded data is not directly equivalent to labeled data, in this work we find it to be useful for several tasks, including code comment prediction (Chapter 2) and grounded ontology construction (Chapter 4). Domain-specific data for grounding is also potentially more easily available than labeled data, which can motivate a preference for grounding over supervised or semi-supervised approaches.

7.2.4 Latent Tensor Representation of Knowledge Bases

The topic modeling components included in the KB-LDA model (Chapter 5) learn subsumption and SVO relations among "categories" based on input relations extracted directly from text. A related alternative approach is to represent similar extracted relations in a tensor, using tensor decomposition algorithms to recover relations among latent categories. This area has recently been explored, in particular for the tasks of relation extraction [Chang et al., 2014] and knowledge base completion [Socher et al., 2013], but it remains as future work to compare the advantages of each approach. One advantage of the topic model seems to be that it is well suited for jointly learning multiple types of relations, which can also be directly combined with corpus statistics. The analysis in Chapter 6 shows that ontology learning greatly benefits from the inclusion of relation examples in the model, and that all learned KB relations benefit from added corpus statistics. If corpus statistics could be incorporated in the tensor factorization process, it seems reasonable that this approach could display similar benefits.

7.3 Future Directions

7.3.1 Improving Software Language Modeling with a Software Ontology

Given an ontology for the software domain, such as the one described in Chapters 4 and 5, it will be interesting to see if we can improve a popular task in this domain. Recently, there has been numerous efforts in creating a language model which describes software code. As an example, a simple model may use n-grams of code tokens trained over a code

repository, as in [Schulam et al., 2013]. In this case, given a history of code tokens h_i, the model would predict the next token to be the t_i which satisfies $t_i = \arg\max_{t_i} p(t_i|h_i)$.

Such models can be used to assist a programming workflow by making real-time suggestions to the programmer, detecting bugs, or by learning mappings between API usages in different languages [Ray et al., 2015, Nguyen et al., 2014]. To address the potential sparsity in software language models, for example in modeling infrequently used objects, or in modeling variable names which are not consistent between projects, one can use a higher-level ontology category of an object, as a backoff approach. In the context of language modeling, for some tokens, instead of considering $p(t_i|h_i)$ we might consider $p(c_i|h_i)$, where c_i is the category of t_i in some ontology. These types of backoff strategies have been explored before for natural language where c_i represented some semantic classification of the token t_i, such as its part of speech tag [Rosenfeld, 2000]. One way of assigning such a semantic context in the software domain, is to ground the examined tokens in a pre-defined ontology. It would therefore be interesting to explore the usefulness of our derived ontologies in this setting.

7.3.2 Learning a Grounded Ontology

The KB-LDA model described in Chapters 5 and 6, currently draws information only from corpus statistics. There are, however, several advantages to learning an ontology based on additional, grounded, sources of information, such as the implementation of classes and systems, software traces which represent run-time method calls, or other representations of the software structure, such as the type hierarchy. As we have seen in Chapters 4 and 2, the way people talk about software is apparently quite different than the way it is used and implemented. This is evident, for example, by the fact that coordinate terms learned from corpus-based distributional similarity are different than ones learned by similar code-based statistics. Another evidence of this is in Figure 5.3 which indicates that a major interest in software literature revolves around the relationship and interaction between *users* and *code*, as well as between code fragments. A learned ontology which takes into account both code and text information, and possibly additional ones, will therefore be more accurate, and probably also more robust, as the learned relations are backed by more evidence. Another important advantage of grounded resources is that they may contain "common sense" knowledge, such as facts that are sufficiently known, clear, or common, that they are not mentioned in natural discourse. For example, sets and queues are types of containers and integers and booleans are examples of primitive Java data types – these basic and obvious relations are known to any Java programmer, and they can be easily learned from the Java type hierarchy, but not all of these basic relations may be enumerated

in any given text corpus.

The proposed KB-LDA model (depicted in Figure 5.1) can be intuitively extended to include grounded data from a target domain, where grounded components can model information taken directly from a grounded source. A possible extension is shown in Figure 7.1, which incorporates coordinate term relations into the model. Coordinate relations indicate sets of instances that belong in the same hierarchy sub-tree, and as we have shown before, they can be learned by a combination of text and code based information [Movshovitz-Attias and Cohen, 2015b]. In the suggested model, coordinate pairs are assigned a single topic and therefore can drive the model to learning coherent functional groups, based on information coming directly from the code. In the suggested formulation,the learned coordinate topics are shared with the instance topics used by other noun-based components, and they therefore affect the induced ontology and relations. This formulation suggests the following change to the joint distribution given in Equation 5.1

$$
\begin{aligned}
&p(\pi_O, \pi_R, \sigma, \delta, \mathbf{CI}, \boldsymbol{z^{CI}}, \mathbf{SVO}, \boldsymbol{z^{SVO}}, \boldsymbol{\theta}, \mathbf{E}, \boldsymbol{z^{D}}, \mathbf{CT}, \boldsymbol{z^{CT}} | \\
&\quad \alpha_O, \alpha_R, \alpha_D, \alpha_{CT}, \gamma_I, \gamma_R) = \\
&\underbrace{\prod_{k=1}^{K} \mathrm{Dir}(\sigma_k|\gamma_I)}_{\text{Instance topics}} \times \underbrace{\prod_{k'=1}^{K} \mathrm{Dir}(\delta_{k'}|\gamma_R)}_{\text{Relation topics}} \times \underbrace{\prod_{d=1}^{N_D} \mathrm{Dir}(\theta_d|\alpha_D) \prod_{l1=1}^{N_{d,I}} \theta_d^{z^D_{E_{l1}}} \sigma^{E_{l1}}_{z^D_{E_{l1}}} \prod_{l2=1}^{N_{d,R}} \theta_d^{z^D_{E_{l2}}} \delta^{E_{l2}}_{z^D_{E_{l2}}}}_{\textit{Documents} \text{ component}} \times \\
&\underbrace{\mathrm{Dir}(\pi_O|\alpha_O) \prod_{i=1}^{N_O} \pi_O^{\langle z_{C_i}, z_{I_i} \rangle} \sigma^{C_i}_{z_{C_i}} \sigma^{I_i}_{z_{I_i}}}_{\textit{Ontology} \text{ component}} \times \underbrace{\mathrm{Dir}(\pi_R|\alpha_R) \prod_{j=1}^{N_R} \pi_R^{\langle z_{S_j}, z_{O_j}, z_{V_j} \rangle} \sigma^{S_j}_{z_{S_j}} \sigma^{O_j}_{z_{O_j}} \delta^{V_j}_{z_{V_j}}}_{\textit{Relations} \text{ component}} \times \\
&\underbrace{\mathrm{Dir}(\pi_{CT}|\alpha_{CT}) \prod_{k=1}^{N_{CT}} \pi_{CT}^{z_{CT_k}} \sigma^{CT_{1k}}_{z_{CT_k}} \sigma^{CT_{2k}}_{z_{CT_k}}}_{\textit{Coordinate-Terms} \text{ component}}
\end{aligned}
\tag{7.1}
$$

In a related work, Chemudugunta et al. [2008] suggest combining concept hierarchies in an LDA model. Their formulation uses a binary switch (x) to determine whether each word in a document is generated from the word distribution associated with topic t, or from one of C concepts in a concept tree. The conditional word distribution is, then, modeled as:

$$
p(w|d) = p(x = 0|d) \sum_t p(w|t)p(t|d) + p(x = 1|d) \sum_c p(w|c)p(c|d) \tag{7.2}
$$

where $p(c|d)$ reflects the path from the root of the ontology to concept c. Note that in this formulation, the ontology hierarchy is pre-defined and not learned from the data, and so

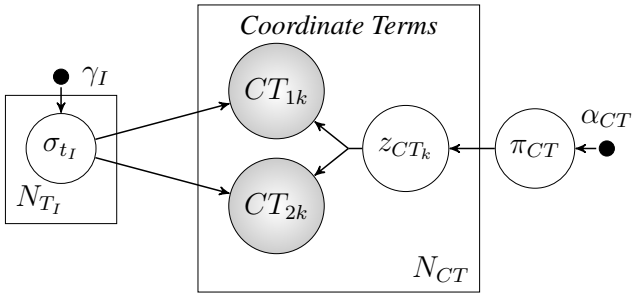


Figure 7.1: Coordinate terms extension to KB-LDA.

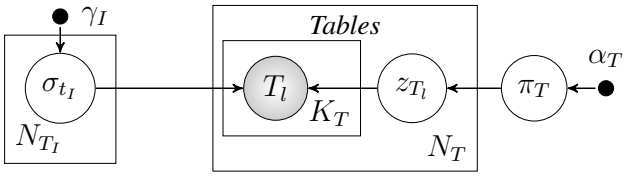


Figure 7.2: Tables extension of KB-LDA.

while the ontology affects the learning of topics, the topics do not affect the ontology.

Similarly to the suggested coordinate terms component, we can consider an additional component based on data extracted from tables, which aggregate a larger set of terms belonging to a single topic (Figure 7.2). We can additionally include components, similar to the Ontology component described in Chapter 5, which draw directly from the available type and package hierarchies. We omit here the derivation of the complete joint distribution and Gibbs update rules for the suggested components.

We note that combining these components in a single learning framework allows us a certain amount of control over the significance and weight given to each input resource, which can be tuned using the parameters of the model. This level of control is much harder to achieve when working directly with the corpus or grounded statistics.

7.3.3 Semi-Supervised Ontology Learning

The KB-LDA topic model is fully unsupervised. The learning pipeline starts with extraction of concept-instance and subject-verb-object relations from a text corpus, and continues with learning topics over this data using the model. In some domains, however, we have existing pre-defined knowledge that can help guide ontology learning, for example, we may have access to an incomplete ontology, or we might be particularly interested in

representing specified sets of objects. We have started exploring a semi-supervised variant of this model, which takes in ”hints” of interesting areas in the ontology and expands on them. Our initial experiments in this area have included modifying the Gibbs sampling process, specifically in the update equations 5.2 and 5.3 described above. One possibility of introducing supervision here is by starting the topic update process by addressing only the supervised (provided) terms, and then with every update iteration, extend the set of addressed terms to include to the most relevant ones which are connected to the current set. This means that the set of updated terms grows in each iteration according to the connections presented by the input examples. This is in contrast to the normal Gibbs sampling process where all term topics are updated in every iteration. In effect, this process simulates the idea of bootstrapping, with the additional advantage of doing so while jointly considering ontology and relation constraints. More approaches for introducing semi-supervision to mixed-membership models have been discussed by Balasubramanyan et al. [2013].

Bibliography

Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. In *Advances in Neural Information Processing Systems*, pages 33–40, 2009. 5.2

Miltiadis Allamanis, Earl T Barr, and Charles Sutton. Learning natural coding conventions. *arXiv preprint arXiv:1402.4182*, 2014. 5.4

Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25, 2000. 3.3.3

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007. 1.1

Ramnath Balasubramanyan and William W Cohen. Block-lda: Jointly modeling entity-annotated text and entity-entity links. In *Proceedings of the 7th SIAM International Conference on Data Mining*, 2011. 2.2.1, 5.4

Ramnath Balasubramanyan, Bhavana Dalvi, and William W Cohen. From topic models to semi-supervised learning: Biasing mixed-membership models to exploit topic-indicative features in entity clustering. In *Machine Learning and Knowledge Discovery in Databases*, pages 628–642. Springer, 2013. 7.3.3

Jonathan Bard, Seung Y Rhee, and Michael Ashburner. An ontology for cell types. *Genome Biology*, 6(2):R21, 2005. 3.3.3

Dave Binkley, Matthew Hearn, and Dawn Lawrie. Improving identifier informativeness using part of speech information. In *Proc. of the Working Conference on Mining Software Repositories*. ACM, 2011. 2.1, 4.1, 4.2.3

David M Blei and Michael I Jordan. Modeling annotated data. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2003. 2.1

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 2003. 2.1, 2.2.1, 2.2.1, 2.2.2, 5.2, 5.4, 6.3, 6.4.3

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008. 4.5.3

Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270, 2004. 6.2

SRK Branavan, Luke S Zettlemoyer, and Regina Barzilay. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. ACL, 2010. 4.1

Janez Brank, Marko Grobelnik, and Dunja Mladenic. A survey of ontology evaluation techniques. In *Proceedings of the conference on data mining and data warehouses (SiKDD 2005)*, pages 166–170, 2005. 6.5

Christopher Brewster, Harith Alani, Srinandan Dasmahapatra, and Yorick Wilks. Data driven ontology evaluation. *Proceedings of the International Conference on Language Resources and Evaluation*, 2004. 6.5

Marcel Bruch, Martin Monperrus, and Mira Mezini. Learning from examples to improve code completion systems. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 213–222. ACM, 2009. 2.3.2

Razvan Bunescu, Ruifang Ge, Rohit J Kate, Edward M Marcotte, Raymond J Mooney, Arun K Ramani, and Yuk Wah Wong. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2), 2005. 3.4.3

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. Clueweb09 data set. *http://boston.lti.cs.cmu.edu/Data/clueweb09/*, 2009. 3.3.2

Andrew Carlson, Justin Betteridge, Estevam R Hruschka Jr, and Tom M Mitchell. Coupling semi-supervised learning of categories and relations. *Semi-supervised Learning for Natural Language Processing*, page 1, 2009. 3.1

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010. 1.1, 3, 3.2, 3.3.1, 3.4.1, 4.1, 4.5.2, 5.1

Bob Carpenter. Phrasal queries with lingpipe and lucene: ad hoc genomics text retrieval. *NIST Special Publication: SP*, pages 500–261, 2004. 3.2

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011. 4.4.2

Jeffrey T Chang, Hinrich Schütze, and Russ B Altman. Gapscore: finding gene and protein names one word at a time. *Bioinformatics*, 20(2):216, 2004. 3.2

Jonathan Chang and David M Blei. Relational topic models for document networks. In *International Conference on Artificial Intelligence and Statistics*, pages 81–88, 2009. 5.4

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296, 2009. 5.3.2, 6.5

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1579, 2014. 7.2.4

Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. Combining concept hierarchies and statistical topic models. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1469–1470. ACM, 2008. 7.3.2

Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *arXiv preprint arXiv:1301.3618*, 2013. 5.4

Lifeng Chen, Hongfang Liu, and Carol Friedman. Gene name ambiguity of eukaryotic nomenclatures. *Bioinformatics*, 21(2):248, 2005. 3.2

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990. 3.3.4

William W Cohen, Pradeep D Ravikumar, Stephen E Fienberg, et al. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, 2003. 4.3.2

James R Curran, Tara Murphy, and Bernhard Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, 2007. 3.2

James Richard Curran. *From distributional to semantic similarity*. PhD thesis, University of Edinburgh. College of Science and Engineering. School of Informatics., 2004. 4.1, 4.2.1

Bhavana Bharat Dalvi, William W Cohen, and Jamie Callan. Websets: Extracting sets of entities from the web using unsupervised information extraction. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 243–252. ACM, 2012. 5.4

Kirill Degtyarenko, Paula De Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic acids research*, 36(suppl 1):D344, 2008. 3.3.3

Andrew Dolbey, Michael Ellsworth, and Jan Scheffczyk. Bioframenet: A domain-specific framenet extension with links to biomedical ontologies. In *Proceedings of KR-MED*, pages 87–94. Citeseer, 2006. 3.2

Xin Luna Dong, K Murphy, E Gabrilovich, G Heitz, W Horn, N Lao, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014. 1.1

Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240, 1967. 5.2.1

Karen Eilbeck, Suzanna E Lewis, Christopher J Mungall, Mark Yandell, Lincoln Stein, Richard Durbin, and Michael Ashburner. The sequence ontology: a tool for the unification of genome annotations. *Genome biology*, 6(5):R44, 2005. 3.3.3

Elena Erosheva, Stephen Fienberg, and John Lafferty. Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences of the United States of America*, 2004. 2.1, 2.2.1, 5.2, 5.4, 6.3

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. ACL, 2011. 4.1, 5.1, 5.3.4

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9, 2008. 4.4.2

Christiane Fellbaum. Wordnet: An electronic lexical database, 1998. 4.1

Yansong Feng and Mirella Lapata. How many words is a picture worth? automatic caption generation for news images. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010. 2.1

Yansong Feng and Mirella Lapata. Automatic caption generation for news images. *IEEE transactions on pattern analysis and machine intelligence*, 2013. 2.1

Mark Gabel and Zhendong Su. Javert: fully automatic mining of general temporal properties from dynamic traces. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. ACM, 2008. 2.1, 4.2.3

Ruifang Ge and Raymond J Mooney. A statistical semantic parser that integrates syntax and semantics. In *Computational Natural Language Learning*. ACL, 2005. 4.2.2

Roxana Girju, Adriana Badulescu, and Dan Moldovan. Learning semantic constraints for the automatic discovery of part-whole relations. In *North American Chapter of the Association for Computational Linguistics on Human Language Technology*. ACL, 2003. 4.2.1

Google. Freebase data dumps. http://download.freebase.com/datadumps/, 2011. 1.1, 3.4.1, 5.1

Peter Gorniak and Deb Roy. Situated language understanding as filtering perceived affordances. *Cognitive Science*, 2007. 4.2.2

Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proc. of the National Academy of Sciences of the United States of America*, 2004. 2.2.1, 5.2.1

Sangmok Han, David R Wallace, and Robert C Miller. Code completion from abbreviated input. In *Automated Software Engineering, 2009. ASE'09. 24th IEEE/ACM International Conference on*, pages 332–343. IEEE, 2009. 2.1, 2.3.2, 4.1, 4.2.3

Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*. ACL, 1992. 4.2.1, 5.1

Kristina M Hettne, Rob H Stierum, Martijn J Schuemie, Peter JM Hendriksen, Bob JA Schijvenaars, Erik M Van Mulligen, Jos Kleinjans, and Jan A Kors. A dictionary to identify small molecules and drugs in free text. *Bioinformatics*, 25(22):2983–2991, 2009. 6.2

Abram Hindle, Earl T Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. On the naturalness of software. In *Software Engineering (ICSE), 2012 34th International Conference on*. IEEE, 2012. 2.1, 5.4

Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. Overview of biocreative: critical assessment of information extraction for biology. *BMC bioinformatics*, 6(Suppl 1):S1, 2005. 3.4.1

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: a spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013. 1.1, 5.1

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999. 6.4.3

Sarah Hunter, Rolf Apweiler, Teresa K Attwood, Amos Bairoch, Alex Bateman, David Binns, Peer Bork, Ujjwal Das, Louise Daugherty, Lauranne Duquenne, et al. Interpro: the integrative protein signature database. *Nucleic acids research*, 37(suppl 1):D211–D215, 2009. 6.2

Ferosh Jacob and Robert Tairas. Code template inference using language models. In *Southeast Regional Conference*. ACM, 2010. 2.1, 4.1, 4.2.3

Rohit J Kate and Raymond J Mooney. Learning language semantics from ambiguous supervision. In *AAAI*, 2007. 4.2.2

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95*. IEEE, 1995. 2.2.1

Zornitsa Kozareva and Eduard Hovy. Not all seeds are equal: measuring the quality of text mining seeds. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 618–626. Association for Computational Linguistics, 2010. 3.2

Martin Krallinger, Alfonso Valencia, and Lynette Hirschman. Linking genes to literature: text mining, information extraction, and retrieval applications for biology. *Genome Biol*, 9(Suppl 2):S8, 2008. 3.2

Jayant Krishnamurthy and Thomas Kollar. Jointly learning to parse and perceive: Connecting natural language to the physical world. *TACL*, 2013. 4.2.2

Jayant Krishnamurthy and Tom M. Mitchell. Which noun phrases denote which concepts? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2011. 3.1

Jayant Krishnamurthy and Tom M Mitchell. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. ACL, 2012. 4.1

Naveen Kumar and Benjamin Carterette. Time based feedback and query expansion for twitter search. In *Advances in Information Retrieval*. Springer, 2013. 2.1

Ni Lao, Tom Mitchell, and William W Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011. 4.5.2

Dawn Lawrie, Christopher Morrell, Henry Feild, and David Binkley. What's in a name? a study of identifiers. In *ICPC 2006. 14th IEEE International Conference on*, 2006. 2.1, 4.2.3

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. 6.4.3

Percy Liang, Michael I Jordan, and Dan Klein. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009. 4.2.2

Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. Identifying synonyms among distributionally similar words. In *IJCAI*, 2003. 4.2.1

Thomas Lin, Oren Etzioni, et al. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. ACL, 2012. 4.1

Hongfang Liu, Zhang-Zhi Hu, Jian Zhang, and Cathy Wu. Biothesaurus: a web-based thesaurus of protein and gene names. *Bioinformatics*, 22(1):103–105, 2006. 6.2

Adolfo Lozano-Tello and Asunción Gómez-Pérez. Ontometric: A method to choose the appropriate ontology. *Journal of Database Management*, 2(15):1–18, 2004. 6.5

Alexander Maedche and Steffen Staab. Measuring similarity between ontologies. In *Knowledge engineering and knowledge management: Ontologies and the semantic web*, pages 251–263. Springer, 2002. 6.5

Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th Biennial Conference on Innovative Data Systems Research*. CIDR 2015, 2014. 1.1

Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002. 2.3.1, 4.4.1

Tara McIntosh and James R Curran. Reducing semantic drift with bagging and distributional similarity. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 396–404. Association for Computational Linguistics, 2009. 3.2

George A Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 1995. 4.1

Thang Luong Minh, Michael C Frank, and Mark Johnson. Parsing entire discourses as very long strings: Capturing topic continuity in grounded language learning. *TACL*, 2013. 4.2.2

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapa Nakashole, Emmanouil Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015. 5.1

Thahir P Mohamed, Estevam R Hruschka Jr, and Tom M Mitchell. Discovering relations between noun categories. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1447–1455. Association for Computational Linguistics, 2011. 5

Alexander A Morgan, Lynette Hirschman, Marc Colosimo, Alexander S Yeh, and Jeff B Colombe. Gene name identification and normalization using a model organism database. *Journal of Biomedical Informatics*, 37(6):396–410, 2004. 3.2

Dana Movshovitz-Attias and William W Cohen. Bootstrapping biomedical ontologies for scientific text using nell. Technical report, Carnegie Mellon University, CMU-ML-12-101, 2012a. 1.1, 6.1

Dana Movshovitz-Attias and William W Cohen. Bootstrapping biomedical ontologies for scientific text using nell. In *BioNLP: Biomedical Natural Language Processing at NAACL*, pages 11–19, Montréal, Canada, June 2012b. Association for Computational Linguistics. 1.1, 3.3.4

Dana Movshovitz-Attias and William W Cohen. Natural language models for predicting programming comments. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 35–40, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. 1.1, 4.1, 4.2.3, 5.4

Dana Movshovitz-Attias and William W Cohen. Kb-lda: Jointly learning a knowledge base of hierarchy, relations, and facts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1449–1459, Beijing, China, July 2015a. Association for Computational Linguistics. 1.1, 6.1, 6.3, 6.4.2, 6.5

Dana Movshovitz-Attias and William W Cohen. Grounded discovery of coordinate term relationships between software entities. *ArXiv e-prints*, May 2015b. 1.1, 5.4, 7.3.2

Dana Movshovitz-Attias, Steven Euijong Whang, Natalya Noy, and Alon Halevy. Discovering subsumption relationships for web-based ontologies. In *Proceedings of the 18th International Workshop on Web and Databases (WebDB)*, pages 62–69. ACM, 2015. 7.2.2

Ramesh M Nallapati, Amr Ahmed, Eric P Xing, and William W Cohen. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–550. ACM, 2008. 5.4

Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001. 6.5

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828, 2009a. 5.2.2

David Newman, Sarvnaz Karimi, and Lawrence Cavedon. External evaluation of topic models. In *in Australasian Doc. Comp. Symp., 2009*. Citeseer, 2009b. 6.5

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108. Association for Computational Linguistics, 2010. 6.5

Michael A Newton and Adrian E Raftery. Approximate bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 3–48, 1994. 6.5

Anh Tuan Nguyen, Tung Thanh Nguyen, Hoan Anh Nguyen, Ahmed Tamrawi, Hung Viet Nguyen, Jafar Al-Kofahi, and Tien N Nguyen. Graph-based pattern-oriented, context-sensitive source code completion. In *Proceedings of the 34th International Conference on Software Engineering*, pages 69–79. IEEE Press, 2012. 2.3.2

Anh Tuan Nguyen, Hoan Anh Nguyen, Tung Thanh Nguyen, and Tien N Nguyen. Statistical learning approach for mining api usage mappings for code migration. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 457–468. ACM, 2014. 7.3.1

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280. ACM, 2012. 5.4

Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, 2006. 4.4.1

Natalya F Noy. Semantic integration: a survey of ontology-based approaches. *ACM Sigmod Record*, 33(4):65–70, 2004. 6.5

Cyrus Omar. Structured statistical syntax tree prediction. In *Proceedings of the 2013 companion publication for conference on Systems, programming, & applications: software for humanity*. ACM, 2013. 4.2.3

Cyrus Omar, YoungSeok Yoon, Thomas D LaToza, and Brad A Myers. Active code completion. In *Proceedings of the 34th International Conference on Software Engineering*, pages 859–869. IEEE Press, 2012. 2.3.2

John D Osborne, Jared Flatow, Michelle Holko, Simon M Lin, Warren A Kibbe, Lihua J Zhu, Maria I Danila, Gang Feng, and Rex L Chisholm. Annotating the human genome with disease ontology. *BMC genomics*, 10(Suppl 1):S6, 2009. 3.3.3

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*. Association for Computational Linguistics, 2009. 3.2

Patrick Andre Pantel. *Clustering by committee*. PhD thesis, Department of Computing Science, University of Alberta, 2003. 4.1, 4.2.1

Juuso Parkkinen, Janne Sinkkonen, Adam Gyenge, and Samuel Kaski. A block model suitable for sparse graphs. In *Proceedings of the 7th International Workshop on Mining and Learning with Graphs (MLG 2009), Leuven*, 2009. 5.2

Adam Pauls and Dan Klein. Faster and smaller n-gram language models. In *Proceedings of the 49th annual meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011. 2.2.1

Pavel Pecina. Lexical association measures and collocation extraction. *Language resources and evaluation*, 44(1-2):137–158, 2010. 6.5

Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of english words. In *ACL*, 1993. 4.1, 4.2.1, 4.3.1, 4.3.4, 5.1

Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577. ACM, 2008. 5.2.2

Robert Porzel and Rainer Malaka. A task-based approach for ontology evaluation. In *ECAI Workshop on Ontology Learning and Population, Valencia, Spain*. Citeseer, 2004. 6.5

Sarah Rastkar, Gail C Murphy, and Alexander WJ Bradley. Generating natural language summaries for crosscutting source code concerns. In *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*. IEEE, 2011. 2.1

Baishakhi Ray, Vincent Hellendoorn, Zhaopeng Tu, Connie Nguyen, Saheel Godhane, Alberto Bacchelli, and Premkumar Devanbu. On the" naturalness" of buggy code. *arXiv preprint arXiv:1506.01159*, 2015. 7.3.1

Veselin Raychev, Martin Vechev, and Eran Yahav. Code completion with statistical language models. In *ACM SIGPLAN Notices*, volume 49, pages 419–428. ACM, 2014. 2.3.2

Dietrich Rebholz-Schuhmann, Antonio José Jimeno Yepes, Erik M Van Mulligen, Ning Kang, Jan Kors, David Milward, Peter Corbett, Ekaterina Buyko, Elena Beisswanger, and Udo Hahn. Calbc silver standard corpus. *Journal of bioinformatics and computational biology*, 8(01):163–179, 2010. 6.1, 6.3

Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-99)*, pages 474–479, 1999. 3.2

Romain Robbes and Michele Lanza. Improving code completion with program history. *Automated Software Engineering*, 17(2):181–212, 2010. 2.3.2

Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 2000. 2.1, 2.2.1, 7.3.1

Eric W Sayers, Tanya Barrett, Dennis A Benson, Evan Bolton, Stephen H Bryant, Kathi Canese, Vyacheslav Chetvernin, Deanna M Church, Michael DiCuccio, Scott Federhen, et al. Database resources of the national center for biotechnology information. *Nucleic acids research*, 39(suppl 1):D38–D51, 2011. 3.3.3

Peter Schulam, Roni Rosenfeld, and Premkumar Devanbu. Building statistical language models of code. In *Proc. DAPSE*. IEEE, 2013. 4.1, 4.2.3, 7.3.1

David Shepherd, Zachary P Fry, Emily Hill, Lori Pollock, and K Vijay-Shanker. Using natural language program analysis to locate and understand action-oriented concerns. In *Proceedings of the 6th international conference on Aspect-oriented software development*. ACM, 2007. 2.1

Jeffrey Mark Siskind. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 1996. 4.2.2

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*, 2004. 4.2.1, 4.3.5

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2006. 4.1, 5.1

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013. 5.4, 7.2.4

Giriprasad Sridhara, Emily Hill, Divya Muppaneni, Lori Pollock, and K Vijay-Shanker. Towards automatically generating summary comments for java methods. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*. ACM, 2010. 2.1

Steffen Staab and Rudi Studer. *Handbook on ontologies*. Springer Science & Business Media, 2013. 6.5

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007. 1.1, 5.1

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217, 2008. 1.1

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. Acquiring temporal constraints between relations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 992–1001. ACM, 2012a. 5.2

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. Coupled temporal scoping of relational facts. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 2012b. 4.1

Lorraine Tanabe and W John Wilbur. Tagging gene and protein names in biomedical text. *Bioinformatics*, 18(8):1124, 2002. 3.2

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. Association for Computational Linguistics, 2003. 4.4.1

Christoph Treude, M Robillard, and Barthélémy Dagenais. Extracting development tasks to navigate software documentation. *IEEE Transactions on Software Engineering*, 2014. 5.4

Yuen-Hsien Tseng and Da-Wei Juang. Document-self expansion for text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003. 2.1

Peter Turney, Michael L Littman, Jeffrey Bigham, and Victor Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, 2003. 4.2.1

Vishnu Vyas, Patrick Pantel, and Eric Crestan. Helping editors choose better seed sets for entity set expansion. In *Proceeding of the 18th ACM conference on Information and knowledge management*. ACM, 2009. 3.2

Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM, 2009. 6.5

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. Single document summarization with document expansion. In *Proc. of the National Conference on Artificial Intelligence*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007. 2.1

Richard C Wang and William W Cohen. Character-level analysis of semi-structured documents for set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1503–1512. Association for Computational Linguistics, 2009. 3.3.1

Xiaoyin Wang, David Lo, Jing Jiang, Lu Zhang, and Hong Mei. Extracting paraphrases of technical terms from noisy parallel software corpora. In *Proceedings of the ACL-IJCNLP*. ACL, 2009. 4.1

Tuangthong Wattarujeekrit, Parantu K Shah, and Nigel Collier. Pasbio: predicate-argument structures for event extraction in molecular biology. *BMC bioinformatics*, 5(1):155, 2004. 3.2

Markus Weimer, Iryna Gurevych, and Max Mühlhäuser. Automatically assessing the post quality in online discussions on software. In *Proceedings of the 45th Annual Meeting of the ACL*. ACL, 2007. 4.1

Cathy H Wu, Hongzhan Huang, Anastasia Nikolskaya, Zhangzhi Hu, and Winona C Barker. The iproclass integrated database for protein functional analysis. *Computational biology and chemistry*, 28(1):87–96, 2004. 6.2

Roung-Shiunn Wu and Po-Chun Li. Video annotation using hierarchical dirichlet process mixture model. *Expert Systems with Applications*, 2011. 2.1

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics, 2007. 1.1, 4.1, 5.1

Chen Yu and Dana H Ballard. On the integration of grounding language and learning objects. In *AAAI*, 2004. 4.2.2

Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *Uncertainty in Artificial Intelligence*, 2005. 4.2.2