

Towards Literate Artificial Intelligence

Mrinmaya Sachan

June 2019
CMU-ML-19-110

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

Thesis Committee

Eric P. Xing, Chair
Jaime Carbonell
Tom Mitchell
Dan Roth

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

Copyright © 2019 Mrinmaya Sachan

This material is based upon work supported by NSF IIS1218282, IIS1304939, IIS1265301, and IIS1447676; ONR N000141410684 and N000141712463; AFOSR FA95501010247; DARPA FA875009C0172; and AFRL FA87501220342. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, Office of Naval Research, Air Force Office of Scientific Research, Defense Advanced Research Projects Agency, or Air Force Research Laboratory.

Keywords: Machine Learning, Natural Language Understanding, Computational Linguistics, Question Answering, Knowledge Representation and Reasoning, Machine Comprehension

*To my loving parents and my brother.
Thanks for always being there for me.*

Abstract

Standardized tests are used to test students as they progress in the formal education system. These tests are readily available and have clear evaluation procedures. Hence, it has been proposed that these tests can serve as good benchmarks for AI. In this thesis, we propose approaches for solving some common standardized tests taken by students such as reading comprehensions, elementary science exams, geometry questions in the SAT exam and mechanics questions in the AP physics exam. Answering these test problems requires deep linguistic (and sometimes visual) understanding and reasoning capabilities which is challenging for modern AI systems.

In the first part of this thesis, we explore novel approaches to answer natural language comprehension tests such as reading comprehensions and elementary science tests (chapters 4, 5 and 6). These tests evaluate the system's ability to understand text through a question-answering task. We present new latent structure models for these tasks. We posit that there is a hidden (latent) structure that explains the relation between the question, the correct answer, and the piece of text. We call this the answer-entailing structure; given the structure, the correctness of the answer is evident. Since the structure is latent, it must be inferred. We present a unified max-margin framework that learns to find these hidden structures given a corpus of question-answer pairs, and uses what it learns to answer questions on novel texts. We also describe a simple but effective extension of this framework to incorporate multi-task learning on the different subtasks that are required to perform the overall task (chapter 4), a deeper representation of language based on AMRs (chapter 5) and how can we incorporate external knowledge in the answer-entailing structure (chapter 6). These advances help us obtain state-of-the-art performance on two well-known natural language comprehension benchmarks.

In the second part of this thesis (chapter 7), we tackle some hard reasoning problems in the domains of math and science - geometry questions in the SAT exam and mechanics question in the AP physics exam. Solving these problems requires an ability to incorporate the rich domain knowledge as well as an ability to perform reasoning based on this knowledge. We propose a parsing to programs (*P2P*) approach for these problems. *P2P* assumes a formal representation language of the domain and domain knowledge written down as programs. This domain knowledge can be manually provided by a domain expert, or, as we show in our work, can be extracted by reading a number of textbooks in an automated way. When presented with a question, *P2P* learns a representation of the question in the formal language via a multi-modal semantic parser. Then, it uses the formal question interpretation and the domain knowledge to obtain an answer by using a probabilistic reasoner.

A key bottleneck in building these models is the amount of domain-specific supervision required to build them. Thus, in the final part of this thesis (chapter 8), we propose a self-training method based on curriculum learning that jointly learns to generate and answer questions. This method obtains near state-of-the-art models on a number of natural language comprehension tests with lesser supervision.

Acknowledgments

I will undoubtedly forget to mention a few names in this note. If you are one of those people, please forgive my limited memory and know that I am grateful to all of you. I am very fortunate to have had the opportunity to work with some of the most brilliant and helpful people during my time as a grad student at CMU.

First and foremost, I thank my advisor Eric Xing and my committee members: Jaime Carbonell, Tom Mitchell and Dan Roth for their constant guidance and support. Thanks Eric for giving me the freedom to work on problems that really excited me. Thanks Jaime, Tom and Dan for your support and advice. You all have taught me to appreciate the richness, nuance, complexity and diversity of problems in AI and NLP. I will also forever be grateful to Ed Hovy for his advice and wisdom in my early formative years. Ed, your passion for semantics has had an impact on my research philosophy and my understanding of important problems. I am grateful to all of you for your invaluable support during my job search. All of you are role models for me. You are deeply knowledgeable in many different areas and impactful in all the work that you do.

My research has benefitted from my interactions with fantastic internship mentors and a number of amazing collaborators – Matt Richardson, Mark Hopkins, Avinava Dubey, Minjoon Seo and Hannaneh Hajishirzi. I have learnt a lot working with you all. Also, special thanks to Diane, Amy, Mary, Lilly, Stacey and all the wonderful staff at MLD and LTI for making my academic life at CMU so simple.

I'd also like to give a shout out to my amazing bunch of friends – *so bad that it can be laughed at movie* buddies (Abhishek, Akanksha, Avinava, Chandan, Kartik, Gagan, Shashank, Snigdha, Sudhi, Sujay) – thanks for making my time in Pittsburgh so much fun, *Edvisees and LTI old timers* (Huiying, Jiwei, Keerthi, Manaal, Pradeep, Whitney), *Morons* (Abu, Anthony, Avi, Chris, Mariya, Maruan, Otilia), *Dbot wingies* (Abhishek(s), Amit, Ankit, Ashish, Deepak, JD, Kamal, Manish, Mayank, Puneet, Rupesh(s), Sharad, Shashank, Shrey – yes, it was hard recalling your first names) and all the members of the *Sailing* lab.

Finally, I would like to thank my parents for supporting me through this unique and long PhD experience. Mummy-papa, you have been a constant source of inspiration, love and support. Nothing that I do in this life will repay the debt of love I owe you. I wish my dear brother Appy was here today to see me graduate. Appy, words cannot describe how much I miss you.

Contents

1	Introduction	1
1.1	Thesis statement	4
1.2	Contributions of the thesis	4
1.3	Structure of the thesis	5
2	Related Work	7
3	Background	11
3.1	Supervised Machine Learning as Optimization	11
3.2	Structured Prediction	12
3.2.1	Inference and Learning	12
3.3	Structured Prediction with Latent Variables	13
3.4	Special Cases	13
3.4.1	Latent Structure SVM	13
3.4.2	Constraint or Rule driven learning	14
4	Answering Reading Comprehension Problems: Alignment as a proxy for Reasoning	17
4.1	Methodology	19
4.1.1	Alignment based approach to Question Answering	19
4.1.2	Multi-task Learning	20
4.2	Experiments	22
4.3	Conclusion	29
5	AMR as a Semantic Representation	31
5.1	The AMR Meaning Representation Graph	31
5.2	Redefined Scoring Function and Inference	33
5.3	Experiments	35
5.4	Conclusion	37
6	Science Question Answering from Instructional Materials: Incorporating External Knowledge in the Alignment Model	39
6.1	The Answer-Entailing Structure	40
6.2	Inference and knowledge selection	40
6.3	Features	42

6.4	Experiments	43
6.5	Conclusion	45
7	Mathematical Question Answering: A Case for Explicit Reasoning	47
7.1	Related Work	49
7.2	Methodology	52
7.3	Answering Geometry Questions	55
7.3.1	Diagram and Question Parsing	55
7.3.2	Combining Text and Diagram representations	62
7.3.3	Probabilistic Logic for Geometry Question Answering	63
7.3.4	Harvesting Axiomatic Knowledge from Textbooks	63
7.4	Experiments	74
7.4.1	A Learning from Demonstrations Approach	77
7.4.2	Feature Ablation	86
7.4.3	Axioms Harvested	88
7.4.4	Example Solutions and Error Analysis	88
7.5	Answering Newtonian Physics Questions	91
7.5.1	Diagram Parsing Pipeline	91
7.5.2	Domain Theory as Programs	96
7.5.3	The Deductive Solver	97
7.5.4	Experiments	98
7.6	Key Insights and Learnings from P2P systems	99
7.7	Conclusion	100
8	Joint Question Answering and Question Generation	101
8.1	The Question Answering Model	102
8.2	The Question Asking Model	103
8.3	A Semi-supervised Framework for Joint Training of QA and AQ models	105
8.4	Experiments	108
8.5	Conclusion	112
9	Conclusion and Future Work	113
	Bibliography	115

List of Figures

4.1	An example reading comprehension question from the MCTest500 dataset. . . .	18
4.2	The <i>answer-entailing structure</i> for an example from MCTest500 dataset. The question and answer candidate are combined to generate a hypothesis sentence. Then latent alignments are found between the hypothesis and the appropriate snippets in the text. The solid red lines show the word alignments from the hypothesis words to the passage words, the dashed black lines show auxiliary co-reference links in the text and the labelled dotted black arrows show the RST relation (elaboration) between the two sentences. Note that the two sentences do not have to be contiguous sentences in the text.	21
4.3	Comparison of variations of our method against several baselines on the MCTest-500 dataset. The figure shows two statistics, accuracy (on the left) and NDCG ₄ (on the right) on the test set of MCTest-500. All differences between the baselines and LSSVMs, the improvement due to negation and the improvements due to multi-task learning are significant ($p < 0.01$) using the two-tailed paired T-test. The exact numbers are available in the supplementary.	26
5.1	Example latent <i>answer-entailing structure</i> from the MCTest dataset. The question and answer candidate are combined to generate a hypothesis. This hypothesis is AMR parsed to construct a hypothesis meaning representation graph after some post-processing (section 5.1). Similar processing is done for each sentence in the passage as well. Then, a subset (not necessarily contiguous) of these sentence meaning representation graphs is found. These representation subgraphs are further merged using coreference information, resulting into a structure called the relevant text snippet graph. Finally, the hypothesis meaning representation graph is aligned to the snippet graph. The dashed red lines show node alignments, solid red lines show edge alignments, and thick solid black arrow shows the rhetorical structure label (elaboration).	32
5.2	AMR parse for the hypothesis in Figure 5.1. The person nodes are merged to achieve the hypothesis meaning representation graph.	33

6.1	An example <i>answer-entailing structure</i> for science question answering. The answer-entailing structure consists of selecting a particular textbook from the curriculum, picking a chapter in the textbook, picking a section in the chapter, picking sentences in the section and then aligning words/mwe’s in the hypothesis (formed by combining the question and an answer candidate) to words/mwe’s in the picked sentences or some related “knowledge” appropriately chosen from additional knowledge stores. In this case, the relation (greenhouse gases, cause, greenhouse effect) and the equivalences (e.g. carbon dioxide = CO_2) – shown in violet – are hypothesized using external knowledge resources. The dashed red lines show the word/mwe alignments from the hypothesis to the sentences (some word/mwe are not aligned, in which case the alignments are not shown), the solid black lines show coreference links in the text and the RST relation (elaboration) between the two sentences. The picked sentences do not have to be contiguous sentences in the text. All mwe’s are shown in green.	41
6.2	Variations of our method vs several baselines on the Science QA dataset. Differences between the baselines and LSSVMs, the improvement due to negation, the improvements due to multi-task learning and joint-learning are significant ($p < 0.05$) using the two-tailed paired T-test.	45
6.3	Ablation on <i>MTLSSVM(Qword, JT) model</i>	45
7.1	The framework of our Parsing to Programs approach. The approach solves the question in two stages. The first stage, <i>Question Parsing</i> , parses the question text and any associated diagram into an equivalent (weighted) logical expression in a typed first-order logic language. The second stage, <i>Programmatic Solving</i> takes this formal representation of the question and solves it using the domain specific theory provided to the system.	53
7.2	Above: An example Newtonian physics problem. Below: Diagram parsed in formal language.	56
7.3	A pipeline for diagram parsing with various (possibly multiple) pre-trained functions, existing software and rules. The pre-learnt components are shown in blue, the existing software are shown in red and rule-based components are shown in green.	57
7.4	Some example high-level diagram elements: (a) Arrow, (b) Dotted line, (c) Ground, (d) Coordinate System, (e) Block, (f) Wedge, and (g) Pulley. We describe rules to form these elements in Table 7.2.	58

7.5	A sample logical program (in prolog style) that solves the problem in Figure 7.2. The program consists of a set of data structure declarations that correspond to types in the prolog program, a set of declarations from the diagram and text parse and a subset of the geometry axioms written as horn clause rules. The axioms are used as the underlying theory with the aforementioned declarations to yield the solution upon logical inference. Normalized confidence weights from the diagram, text and axiom parses are used as probabilities. For readers understanding, we list the axioms in the order (1 to 7) they are used to solve the problem. However, this ordering is not required. Other (less probable) declarations and axiom rules are not shown here for clarity but they can be assumed to be present.	64
7.6	An excerpt of a textbook from our dataset that introduces the Pythagoras theorem. The textbook has a lot of typographical features that can be used to harvest this theorem: The textbook explicitly labels it as a “theorem”; there is a colored bounding box around it; an equation writes down the rule and there is a supporting figure. Our models leverages such rich contextual and typographical information (when available) to accurately harvest axioms and then parses them to horn-clause rules. The horn-clause rule derived by our approach for the Pythagoras theorem is: $isTriangle(ABC) \wedge perpendicular(AC, BC) \implies BC^2 + AC^2 = AB^2$	65
7.7	An illustration of the three operations to sample axiom blocks.	71
7.8	An example demonstration on how to solve the problem in Figure 1: (1) Use the theorem that the sum of interior angles of a triangle is 180° and additionally the fact that $\angle AMO$ is 90° to conclude that $\angle MOA$ is 60° . (2) Conclude that $\triangle MOA \sim \triangle MOB$ (using a similar triangle theorem) and then, conclude that $\angle MOB = \angle MOA = 60^\circ$ (using the theorem that corresponding angles of similar triangles are equal). (3) Use angle sum rule to conclude that $\angle AOB = \angle MOB + \angle MOA = 120^\circ$. (4) Use the theorem that the angle subtended by an arc of a circle at the centre is double the angle subtended by it at any point on the circle to conclude that $\angle ADB = 0.5 \times \angle AOB = 60^\circ$	78
7.9	State sequence corresponding to the demonstration in Figure 2. Theorems applied are marked in green and the state information is marked in red. Here S_0 corresponds to the state derived from question interpretation and each theorem application subsequently adds new predicates to the logical formula corresponding to S_0 . The final state contains the answer: $measure(ADB, 60^\circ)$. This annotation of states and theorem applications is provided only for illustrative purposes. It is not required by our model.	80
7.10	Horn clause rules for some popular named theorems in geometry harvested by our approach. We also show the confidence our method has on the rule being correct (which is used in reasoning via the problog solver).	89

7.11	An illustrative representation of our approach on a sample question from the Newtonian Physics dataset. The approach solves the question in two stages. The first stage, <i>Question Parsing</i> , parses the question text and any associated diagram into an equivalent (weighted) logical expression in a typed first-order logic language. The logical expression for this example is shown in the two rectangular white boxes. Ideally, each literal in this expression is weighted. However, the weights are not shown in this figure for simplicity. The second stage, <i>Programmatic Solving</i> takes this formal representation of the question and solves it by performing a (probabilistic) search over a set of pre-defined programs. One of the paths in the search (which corresponds to the process required to solve the question) leads to the solution (shown in green), whereas others do not lead to any solution and are rejected (shown in red).	93
7.12	Example programs used by our approach.	96

List of Tables

4.1	Comparison of accuracies on the variations of our method against several baselines on 20 Tasks of the bAbI dataset. All integer differences are significant ($p < 0.01$) using the two-tailed paired T-test.	28
5.1	Comparison of variations of our method against several baselines on the MCTest-500 dataset. The table shows accuracy on the test set of MCTest-500. All differences between the baselines (except SYN+FRM+SEM) and our approaches, and the improvements due to negation and multi-task learning are significant ($p < 0.05$) using the two-tailed paired T-test.	36
6.1	Example questions for <i>Qtype</i> classification	43
7.1	On the left: An example SAT geometry problem. Right: An example AP Newtonian physics problem.	48
7.2	Various components of the diagram parsing pipeline. We denote the pre-learnt functions by ●, software by ● and rules by ● in each stage of the pipeline.	60
7.3	Examples of geometry theorems as horn clause rules.	63
7.4	Feature set for our axiom identification model. The features are based on content and typography.	67
7.5	Feature set for our axiom alignment model. The features are based on content, structure and typography.	70
7.6	Feature set for our axiom parsing model.	73
7.7	Test set Precision, Recall and F-measure scores for axiom identification when performed alone and when performed jointly with axiom alignment. We show results for both strict as well as relaxed comparison modes. For the joint model, we show results when we model ordering constraints as hard or soft constraints.	74
7.8	Test set Precision, Recall, F-measure and NMI scores for axiom alignment when performed alone and when performed jointly with axiom identification. For the joint model, we show results when we model ordering constraints as hard or soft constraints.	75
7.9	Test set Precision, Recall and F-measure scores for axiom parsing. These scores are computed over literals derived in axiom parses or full axiom parses. We show results for the old <i>GEOS</i> system, for the improved <i>GEOS++</i> system with expanded entity types, functions and predicates, and for the multi-source parsers presented in this paper.	75

7.10	Scores for solving geometry questions on the SAT practice and official datasets and a dataset of questions from the 20 textbooks. We use SAT’s grading scheme that rewards a correct answer with a score of 1.0 and penalizes a wrong answer with a negative score of 0.25. <i>Oracle</i> uses gold axioms but automatic text and diagram interpretation in our logical solver. All differences between <i>GEOS</i> and our system are significant ($p < 0.05$ using the two-tailed paired t-test).	76
7.11	User study ratings for <i>GEOS</i> and our system (O.S.) by students in grade 6-10. Ten students in each grade were asked to rate the two systems on a scale of 1-5 on two facets: ‘interpretability’ and ‘usefulness’. Each cell shows the mean rating computed over ten students in that grade for that facet.	77
7.12	The feature set for our joint semantic-parsing and deduction model. Features ϕ_1 and ϕ_2 are motivated from <i>GEOS</i>	83
7.13	Scores of various approaches on the SAT practice (P) and official (O) datasets and a dataset of questions from the 20 textbooks (T). We use SAT’s grading scheme that rewards a correct answer with a score of 1.0 and penalizes a wrong answer with a negative score of 0.25. O.S. represents our system trained on question-answer (QA) pairs, demonstrations, or a combination of QA pairs and demonstrations.	84
7.14	Precision, Recall and F1 scores of the parses induced by <i>GEOS</i> and our models when only the parsing model or the joint model is used.	85
7.15	Accuracy of the programs induced by various versions of our joint model trained on question-answer pairs, demonstrations or a combination of the two. We provide results when we use the deduction model or the joint model.	85
7.16	User study ratings for <i>GEOS++</i> and our system (O.S.) trained on question-answer pairs and demonstrations by a number of grade 6-10 student subjects. Ten students in each grade were asked to rate the two systems on a scale of 1-5 on two facets: ‘interpretability’ and ‘usefulness’. Each cell shows the mean rating computed over ten students in that grade for that facet.	86
7.17	Ablation study results for the axiom identification component. We remove features of the axiom identification component one by one as listed in Table 7.4 and observe the fall in performance in terms of the axiom identification performance as well as the overall performance to gauge the value of the various features.	86
7.18	Ablation study results for the axiom alignment component. We remove features of the axiom alignment component one by one as listed in Table 7.5 and observe the fall in performance in terms of the axiom alignment performance as well as the overall performance to gauge the value of the various features.	87
7.19	Ablation study results for the axiom parsing component. We remove features of the axiom parsing component one by one as listed in Table 7.6 and observe the fall in performance in terms of the axiom parsing performance as well as the overall performance to gauge the value of the various features.	88
7.20	Some correctly answered questions along with explanations generated by our deductive solver for these problems.	90

7.21	Some example failure cases of our approach for solving SAT style geometry problems. In (i) the axiom set contains an axiom that internal angle of a regular hexagon is 120° and that each side of a regular polygon is equal. But there's no way to deduce that the angle CBO is half of the internal angle ABC (by symmetry). The other hand, the coordinate geometry solver can exploit these three facts as maximizing the satisfiability of the various constraints can answer the question. (ii) The solver does not contain any knowledge about construction. The question cannot be correctly interpreted and the coordinate geometry solver also gets it wrong. (iii) The solver does not contain any knowledge about construction or prisms. The question cannot be correctly interpreted and the coordinate geometry solver also gets it wrong. (iv) The question as well as the answer candidates cannot be correctly interpreted (as the concept of perpendicular to plane is not in the vocabulary). Both solvers get it wrong. (v) The parser cannot interpret that angle AC is indeed angle AEC. This needs to be understood by context as it defies the standard type definition of an angle. Both solvers get it wrong. (vi) Both diagram and text parsers fail here. Both solvers answer incorrectly.	92
7.22	Jaccard similarity between detected diagram elements and gold elements for Edge Boxes and two versions of our system's (O.S.) diagram parser: our parser which does not use text information and our full parser.	98
7.23	Precision, Recall and F1 scores of parses induced by our system's text parser compared to a rule based parser.	99
7.24	Scores of our system compared to average score by 10 students on our dataset from physics textbooks (T), AP Physics C Mechanics - Section 1 practice test (P) and official tests for two years: 1998 and 2012.	99
8.1	Statistics of the three machine comprehension datasets used in evaluating our QA and AQ models.	106
8.2	Performance of our model variants and the four QA baselines on <i>SQUAD</i> and <i>MS MARCO</i> datasets. The baseline numbers are taken from [274]. The grey part of the table shows the effect of various question selection heuristics on our SelfTrain(10000) models.	109
8.3	Performance of our model variants and the four QA baselines on <i>WikiQA</i> . The baseline numbers are taken from [274]. The grey part of the table shows the effect of various question selection heuristics on our SelfTrain(10000) models.	110
8.4	Performance of our model variants and the four AQ baselines on the <i>SQUAD</i> , <i>MS MARCO</i> and <i>WikiQA</i> datasets. The grey part of the table shows the effect of various question selection heuristics on our SelfTrain(10000) models.	111

Chapter 1

Introduction

Alan Turing, one of the pioneers of computer science, proposed that it would someday be possible for a sufficiently advanced computer to think and to have some form of consciousness [276]. This was accompanied by a proposal for a test for artificial intelligence known as the *Turing test*. The Turing test proposes that a human evaluator judge natural language conversations between a human and a machine designed to generate human-like responses. If the evaluator cannot reliably tell the machine from the human, the machine is said to have passed the test. Since then, there have been a number of attempts [55, 260, 290] to build a system that can pass the test. While a number of these proposals have indeed come close to technically passing the test, most AI practitioners believe that the AI dream is still a non-reality. Despite the explosion of impressive data-driven AI applications in recent years, computers, unlike humans, largely lack a deeper understanding of the world. More specifically, they cannot efficiently extract, represent or reason with the information that is provided to them. Consequently, if posed with questions about what they have read, they cannot answer questions that go beyond a prototypical typecast – information either explicitly stated in the text, or simple questions relating to recognising objects, etc. in images and videos. On the other hand, human learning is much more general, robust and powerful. Humans display common sense, judgment, reasoning and creative abilities well beyond the capability of modern AI systems.

The formal education system plays an important role in imbibing these abilities in humans. Children typically learn incrementally grade by grade covering instructional content with varying levels of difficulty. Children are frequently tested by various standardized tests to gauge their understanding of the instructional content. As many researchers [93, 164, 195, 201, 222, 249, 252] have pointed out, the definition of the Turing test has resulted in researchers focusing on the wrong task, namely, fooling human judges, rather than achieving true intelligence. Shoehorning the research to meet the goal of appearing human-like is a red herring. To this end, standardised tests have often been proposed as replacements to the Turing test as a driver for progress in AI [50]. These include tests on understanding passages and stories and answering questions about them [227], math and science question answering using instructional material [158, 241, 253], visual question answering [7], etc. Many of these tests require sophisticated understanding of the world, aiming to push the boundaries of AI.

Standardised tests are easily accessible, comprehensible, incremental, and easily measurable. These tests do not cover all aspects of intelligence [50]. For example, spatial/kinematic reason-

ing, some types of commonsense reasoning, and interaction/dialogue are under-represented or absent, and thus the exams do not constitute a full test of machine intelligence, they are necessary but not sufficient. Nonetheless, they cover a wide variety of problem types and levels of difficulty in representing and extracting the knowledge relevant for the task and reasoning, making them a driver for progress in AI. Furthermore, the mark down in the aspects of intelligence required to solve them allows us to tease out and test various components of our AI systems. Standardized tests could potentially allow us to study the skills of AI in contrast with the skill set of humans which are believed to be acquired by humans through education.

In this thesis, we describe some of these tests and propose some approaches for solving them. These include approaches for answering reading comprehension based questions (chapters 4 and 5), answering science questions using instructional material (chapter 6), and answering math and science questions in geometry and Newtonian physics (chapter 7). Such problems frequent the curriculum of students and also appear in standardised tests such as the *SAT* or *Advanced Placement* college level courses.

Reading comprehension tests require the system to answer multiple-choice questions based on a text passage. We show that we can answer such questions well by modeling this as a textual entailment task and learning latent *answer-entailing structures* similar to the structures often used in various models for machine translation [21]. The *answer-entailing structure* aligns parts of the question and answer candidate with candidate snippets in the passage. Furthermore, we show that we can account for question types by introducing a multi-task learner.

Language representation has been a key and open question in natural language understanding and a number of language representation formalisms have been proposed in the recent years. These include domain-specific sembanks like GeoQuery, and more recently larger, broad-coverage formalisms such as the Groningen Meaning Bank, UCCA, Semantic Treebank, the Prague Dependency Treebank and UNL. We also extend our learning latent *answer-entailing structure* learning approach by using various types of language representation – bag of words, syntactic parses and in particular the abstract meaning representation (AMR) [14]. Our results show that in scenarios such as children story comprehension where AMR parsers are typically accurate, using a rich language representation based on AMR indeed leads to improved performance.

We also applied our approach of learning latent *answer-entailing structures* in the *Allen Institute of AI's* Kaggle challenge for 8th grade science question answering which requires a rich integration of language understanding and external knowledge. In this setting, we re-described the *answer-entailing structure* as a search through the student's curriculum comprising of a number of textbooks followed by alignment of the hypothesis to the snippet returned by retrieval and used external domain-specific knowledge resources such as science dictionaries, study guides and semi-structured tables to further refine the answer-entailing structure. We achieved significant improvements over a number of lexical and neural network baselines which have been shown to do well on this task.

Reasoning is another unique human ability. This ability allows us to make sense of things, verify facts, apply logic, and justifying or deny various hypotheses based on prior beliefs and existing information. Standardized tests in the domains of math and science offer us problems which are quite challenging and require significant reasoning abilities. Thus, as a small step towards this goal, we proposed an approach to tackle some hard reasoning problems in various math and science standardized tests – primarily geometry and mechanics problems in

pre-university examinations. Our approach, called *parsing to programs*, combines ideas from semantic parsing [139, 311, 312] and probabilistic programming [100]. When presented with a novel question, the system learns a formal representation of the question by combining interpretations from the question text and any associated diagram. Finally, the system uses this formal representation to solve the question using the relevant domain knowledge provided to it in the form of programs. Our approach can be seen as a natural language interface to expert systems [130]. We use this technique to build two systems – the first answers SAT style geometry questions and the second answers mechanics problems from the AP Physics exams. Both systems achieve near human performance on multiple datasets taken from a variety of textbooks, SAT practice and official tests and section 1 of AP Physics C mechanics exams held in 1998 and 2012.

A key issue in many NLP approaches for question answering (including the work presented in this thesis) is that all these approaches are very data hungry and depend on a large number of labeled question answer pairs. This is expensive to obtain in many domains and is a key impediment in the adoption of this technology. Thus, as a first step in this direction, in the final part of this thesis, we looked at the inverse of the question answering problem – i.e. question generation. We show that question answering and question generation problems are closely related (in fact dual problems) and we proposed a self-training method to jointly ask as well as generate questions by leveraging unlabeled text along with a much smaller set of labeled question answer pairs for learning. Self-training is a common unsupervised data augmentation technique that augments the original training set with model predictions on an unlabeled data. The addition of this synthetic labeled data needs to be performed carefully. During self-training, typically the most *confident* samples, along with their predicted labels, are added to the training set [318]. We show that the performance of our QA model can be used as a proxy for computing the *confidence* value of the questions. We describe a suite of heuristics inspired from curriculum learning [17] to select the unlabeled samples (sentences) to be labeled and added to the training set at each epoch. Curriculum learning is inspired from the incremental nature of human learning and orders training samples on the *easiness* scale so that *easy* samples can be introduced to the learning algorithm first and *harder* samples can be introduced successively. We show that selecting training samples in increasing order of *easiness* leads to improvements over a random sample introduction baseline.

Our approaches of *answer-entailing structures* for reading comprehensions as well as the *parsing to programs* approach for math and science questions provide comprehensive, easy-to-understand and interpretable solution to the questions. This is beneficial as these techniques can assist the students by providing them the deductive or derivational process used to obtain the answer. By conducting various small scale experiments on human subjects (students), we plan to answer the questions if the subjects would find our tools for answering as well as asking questions useful.

In the future, we would like to expand the *answer-entailing structure* and *P2P* frameworks to support question answering and reasoning problems in various other domains. In order to make these frameworks scalable and robust, we would like to tackle the issue of *supervision* by working on *representation learning*, *active (perhaps interactive) learning*, *few shot learning*, *weak supervision*, *domain transfer* and *multi-task learning*. We would like to pursue research that *combines symbolic and non-symbolic methods*. In order to achieve this, we are particularly

interested in advancing some recent work on combining symbolic methods with distributional models of meaning [54], neural networks [257] and statistical learning [68].

1.1 Thesis statement

Motivation: Turing test has resulted in researchers focusing on the wrong task; namely, fooling human judges, rather than achieving true intelligence. We believe that standardized tests can serve as replacements to the Turing test as drivers for progress in AI.

Contributions: We build solvers for various standardized tests, including reading comprehensions and elementary school tests where the goal is to find the support for an answer from the student curriculum, and intermediate level math and science tests which require the system to reason using its prior subject knowledge.

Implications: These solvers provide easy-to-understand solutions, along with the deductive process used to obtain the answer, and can potentially be used as assistive tools in education.

1.2 Contributions of the thesis

In this thesis, we made the following contributions:

- We proposed a *latent answer-entailing structure* approach for solving reading comprehension questions where we model various latent structures to model the reasoning process needed to answer various comprehension questions.
- We further proposed a multi-task learning model that models the various kinds of questions/answers/reasoning types required to solve reading comprehension problems. We show that multi-task learning leads to significant improvements over the base model.
- We also extended the *latent answer-entailing structure* model to use rich graph based semantic representations such as AMR and then used it to obtain state-of-the-art results on popular benchmarks in the community.
- Reasoning is a key challenge in NLP and AI. Thus, as a step in that direction, we proposed a model to answer problems such as geometry and mechanics problems in SAT and AP Physics exams which require rich axiomatic reasoning. Our approach called *parsing to programs* combines semantic parsing and probabilistic reasoning to solve these reasoning problems.
- As a part of our approach to solve geometry reasoning problems, we also developed a model to automatically extract geometry axioms from math textbooks in rich structured forms. This model was driven by formatting features in textbooks and the notion of redundancy to successfully extract and parse geometry axioms.
- A key issue in all of the above models is supervision. Thus, as a step in that direction, we proposed a self-training method based on curriculum learning that jointly learns to generate and answer questions. This model helps us obtain near state-of-the-art models on a number of question answering tasks with less supervision.

1.3 Structure of the thesis

This thesis is structured as follows:

- Chapter 1 introduces the thesis.
- Chapter 2 describes the related work on all the past work on solving various standardized tests.
- Chapter 3 pins down some of the relevant background – in particular structured prediction models, latent variable models and constraint and rule driven learning.
- Chapter 4 describes our *latent answer-entailing structure* approach for solving reading comprehension questions.
- Chapter 5 describes an extension of the *latent answer-entailing structure* approach to rich graph based semantic representations, particularly the AMR representation.
- Chapter 6 describes the extension of our *latent answer-entailing structure* approach with external sources of knowledge for solving science question answering problems .
- Chapter 7 describes our *parsing to programs* approach to answer geometry and mechanics reasoning problems.
- Chapter 8 describes the self-training method based on curriculum learning to jointly learn to generate and answer questions.
- We conclude and describe future work in chapter 9.

This thesis summarizes the work previously published as the following:

- Mrinmaya Sachan, Minjoon Seo, Hannaneh Hajishirzi and Eric P. Xing. Parsing to Programs: A Framework for Situated Question Answering. (In Preparation)
- Mrinmaya Sachan, Avinava Dubey, Eduard Hovy, Tom Mitchell, Dan Roth and Eric P. Xing. Discourse in Multimedia: A Case Study in Information Extraction. (To appear in the Computational Linguistics (CL) journal)
- Mrinmaya Sachan, Avinava Dubey, Tom Mitchell, Dan Roth and Eric P. Xing. Learning Pipelines with Limited Data and Domain Knowledge: A Study in Parsing Physics Problems. NeurIPS 2018
- Mrinmaya Sachan and Eric P. Xing. Parsing to Programs: A Framework for Situated QA. KDD 2018
- Mrinmaya Sachan, Eric P. Xing. Self-Training for Jointly Learning to Ask and Answer Questions. NAACL-HLT 2018
- Mrinmaya Sachan, Avinava Dubey and Eric P. Xing. From Textbooks to Knowledge: A Case Study in Harvesting Axiomatic Knowledge from Textbooks to Solve Geometry Problems. EMNLP 2017
- Mrinmaya Sachan and Eric P. Xing. Learning to Solve Geometry Problems from Natural Language Demonstrations in Textbooks. *SEM 2017
- Mrinmaya Sachan and Eric P. Xing. Easy Questions First? A Case Study on Curriculum Learning for Question Answering. ACL 2016

- Mrinmaya Sachan, Avinava Dubey and Eric P. Xing. Science Question Answering using Instructional Materials. ACL 2016
- Mrinmaya Sachan and Eric P. Xing. Machine Comprehension using Rich Semantic Representations. ACL 2016
- Mrinmaya Sachan, Avinava Dubey, Eric P. Xing and Matthew Richardson. Learning Answer-Entailing Structures for Machine Comprehension. (Best paper nomination, named one of 5 outstanding papers at the conference) ACL 2015

Chapter 2

Related Work

The idea of having a test for AI has a long history and dates back to the discussion by Rene Descartes in his famous treatise *Discourse on the Method* in which he writes that automata (machines) can be capable of responding to human interactions but also argues that such automata cannot respond appropriately to things said in their presence in the way that any human can. This idea of using the insufficiency of linguistic response as what separates the humans from automaton was later corroborated by Denis Diderot in his book *Pensees philosophiques* who famously said that “If one can find a parrot who could answer to everything, I would claim it to be an intelligent being without hesitation”. In his book, *Language, Truth and Logic*, Alfred Jules Ayer suggested a protocol to distinguish between a *conscious* man and an *unconscious* machine. A similar idea was famously proposed by Alan Turing in his famous Computing Machinery and Intelligence paper [276] where he proposed the Turing test. The Turing test is carried out as a sort of imitation game. On one side of a computer screen sits a human judge, whose job is to chat to some mysterious interlocutors on the other side. Most of those interlocutors will be humans; one will be a chatbot, created for the sole purpose of tricking the judge into thinking that it is the real human. Turing argued that if the judges could not distinguish the humans with the computer after the conversation, then it would be unreasonable not to call the computer intelligent, because we judge other people’s intelligence from external observation in just this way.

While the Turing test is powerful and appealing due to its simplicity. it does not directly test whether the computer behaves intelligently. It tests only whether the computer behaves like a human being. This requires that the machine be able to execute all human behaviors, regardless of whether they are intelligent. This led to objections famously being raised by The Economist, in an article entitled “artificial stupidity”. A number of chat bots have been developed which have come close to passing the Turing test. See ELIZA [289] and PARRY [55] as two early attempts at cracking the test. While the final nature of the results has also been debated, it has been seen that most of the systems competing to pass the test resort to tricks like masquerading as people with specific roles (e.g. ELIZA pretended to be a Rogerian therapist whereas PARRY pretended to be a paranoid schizophrenic) or deliberately adding errors into their output, so as to be better “players” of the game. Due to these criticisms, it has been argued that systems attempting to pass the Turing test are not necessarily intelligent and thus Turing test is not a tractable test for AI. Many researchers have since mooted several alternatives to the Turing test.

An interesting recent proposition is to use standardized tests as alternatives to the Turing

test. Standardized tests are large-scale tests administered to large populations of students, such as a multiple-choice science tests given to all the eighth-grade public-school students in the US. These tests are consistently administered to objectively measure the knowledge and skills students learned in school or to determine the academic progress they have made over a period of time. Thus it is natural to use these tests to measure the intelligence of our AI systems. The tests have an added advantage that they can be provide a graded measure of machine Intelligence with respect to humans. The tests are readily available and they reduce any potential for favoritism, bias, or subjective evaluation.

The proposition of using standardized tests as challenge problems also has a long history in AI. In 1972, Charniak, in his PhD thesis [43], proposed a *background* model to answer questions about children’s stories. Also, famously, Hirschmann et al. [119] showed that a bag of words pattern matching approach with some additional automated linguistic processing could achieve 40% accuracy for the task of picking the sentence that best matches the query for “who / what / when / where / why” questions, on the well-known reading comprehension dataset called Remedia. The results on this dataset have since been improved upon by [104, 109, 291]. Riloff et. al [228] also developed a rule-based system, Quarc, which used lexical and semantic clues in the question and the story to answer questions about it. On reading comprehension tests given to children in grades 3-6, Quarc also achieved an accuracy of around 40%. Breck et. al [25] collected 75 stories from the Canadian Broadcasting Corporation’s web site for children and generated 650 questions for them manually where each question was answered by a sentence in the text. Leidner et. al [162] used the CBC4kids data and added layers of annotation (such as semantic and POS tags), thus measuring QA performance as a function of question difficulty.

More recently, there has been a renewed interest in reading comprehensions (also known as machine comprehension in some works). The nomenclature of machine comprehension was popularized by [227] who crowd-sources dataset of 660 stories and multiple-choice questions. By restricting the vocabulary, it was ensured that the stories could be understood by the average 7 year old. Since the popularity of deep learning, a number of larger datasets have been built for machine comprehension. Popular examples include Children’s Book Test [118] from FAIR, CNN/Daily Mail [117] released by Google DeepMind, Stanford Question Answering Dataset or SQuAD [224], LAnuage Modeling Broadened to Account for Disclourse Aspects or LAMBADA dataset [211] from University of Trento and University of Amsterdam, QuizBowl questions [129] from University of Maryland and University of Colorado, NewsQA dataset [275] from Maluuba Research, and MS MARCO [203] from Microsoft. Infact, a number of companies and startups such as Allen Institute for AI ¹ and Microsoft Maluuba² focus on the idea of building solvers for various standardized tests. Various lexical approaches [227, 265], structured prediction approaches [202, 236, 239, 241, 285], and neural network approaches [45, 124, 182, 254, 259, 305] have been proposed for a number of these datasets and the state-of-the-art on these datasets continues to move at the time of writing this document.

Simultaneously, there has been interest in building QA systems for other standardized tests. For example, there has been significant work in science question answering [140, 141, 142, 256], solving algebra word problems [122, 123, 158, 189, 231, 232] and SAT style geometry problems

¹<http://allenai.org/>

²<http://www.maluuba.com/>

[253]. Infact, researchers at the National Institute of Informatics in Tokyo have undertaken the grand challenge of creating an AI system that can answer real questions on university entrance examinations of the University of Tokyo [8]. While, the project has not succeeded in doing so, their system is reportedly already competent enough to pass the entrance exams of 404 out of 744 private universities in Japan.

The latter part of this thesis focuses on generation of questions. Question generation, also referred to as question asking in this thesis, has been pursued before in a number of works such as [114] which focuses on generating questions based on given sources of knowledge. More recently, [70, 274] build on the sequence to sequence paradigm of deep learning to generate questions conditioned on an answer sentence. Finally, we build upon the relationship between question answering and question asking and propose a joint training framework for the two. Concurrently, some recent works [274, 288] also build upon this idea. We will distinguish our work with these later in chapter 8. This work was published as [234, 235].

In the final part of this thesis, we show how our works can be potentially useful as assistive tools for education. A large body of work already employs AI and automated methods for recommending, organizing and optimizing content modules [1, 69, 101, 125, 181, 207], to track student knowledge and recommend next steps using adaptive learning systems or game-based learning [26, 72, 188, 221, 269], grading systems that assess and score student responses to assessments and computer assignments at large scale, either automatically or via peer grading for scoring student responses or detecting plagiarism [2, 11, 18, 30], and predicting enrollment, placements and boosting retention [175, 210, 296]. In our work, we show how question answering and question generation can be used to assist student learning.

Chapter 3

Background

Statistical machine learning approaches are being widely used in NLP and computer vision today. In this chapter, we will cover a few basic machine learning concepts which will be used in the rest of the thesis. We will mainly focus on supervised learning and structured prediction, since these paradigms are most widely used in this thesis.

3.1 Supervised Machine Learning as Optimization

The dominant paradigm of machine learning is supervised learning. In supervised learning problems, we are given data in the form of input-output pairs $\{(x, y) \in \mathcal{X} \times \mathcal{Y}\}$. An example supervised learning problem is document classification where the inputs $x \in \mathcal{X}$ are documents and outputs $y \in \mathcal{Y}$ are a set of document classes and the goal is to learn a classification function $f : \mathcal{X} \rightarrow \mathcal{Y}$ which learns to predict the document class for a given document x .

In supervised learning, we typically define an appropriate loss function $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^{\geq 0}$ which models the loss $L(y, f(x))$ incurred when the predicted output $f(x)$ is not the same as the true output y . Thus, given a training dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$, supervised learning is formulated as an optimization problem (also known as the *empirical risk minimization* (ERM) framework where our goal is to learn the function \hat{f} which best optimizes the sum of losses over all samples in the training data:

$$\hat{f} = \arg \min_f \sum_{i=1}^N L(y_i, f(x_i))$$

A typical idea in machine learning is *regularization* where the idea is to regularize i.e. incorporate some priors on the weights \mathbf{w} . In this case the ERM framework is written as:

$$\hat{f} = \arg \min_f \sum_{i=1}^N L(y_i, f(x_i)) + \Omega(\mathbf{w})$$

Here, $\Omega(\mathbf{w})$ is the regularizer on weights \mathbf{w} .

3.2 Structured Prediction

Often in natural language processing and computer vision, the output space \mathcal{Y} is structured, where the goal is to predict complex structures instead of labels. The structures are typically semantic or linguistic structures or structured labels in a complex scene. For illustration purposes, let us take the canonical example of part-of-speech tagging. Given an input sentence to the problem denoted as \mathbf{x} , and the output structure denoted as \mathbf{y} , the output structure \mathbf{y} can be decomposed into a set of m variables y_1, y_2, \dots, y_m , where m is the number of words in the input sentence \mathbf{x} . Each output variable y_i denotes the part of speech tag for the i^{th} word of input sentence \mathbf{x} . Each y_i is assigned one of the 45 part-of-speech tags.

The structured prediction problem is to learn a mapping function $f(\mathbf{x}; \mathbf{w})$ from the input space \mathcal{X} to the output space \mathcal{Y} . This mapping function is parameterized with weights \mathbf{w} of the relevant features $\phi(\mathbf{x}, \mathbf{y})$. Note that the feature function is defined over both the entire input structure \mathbf{x} and the entire output structure \mathbf{y} . However, in practice the features are defined locally to express the dependency between the local inputs and outputs. The design of these feature templates typically requires a lot of domain knowledge. The weight vector \mathbf{w} and the feature function $\phi(\mathbf{x}, \mathbf{y})$ are typically used to define a scoring function for the output structures. Often the scoring function takes the following linear form:

$$S(\mathbf{y}; \mathbf{w}, \mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$$

Ideally, this scoring function would assign a higher score to the correct output structure and a lower score to all other structures.

3.2.1 Inference and Learning

In many machine learning approach, there are two key involved issues: *inference* and *learning*. Given the scoring function $S(\mathbf{y}; \mathbf{w}, \mathbf{x})$, the *Inference* procedure predicts the output structure for the input \mathbf{x} by searching for the best structure according to the scoring function. Formally, the inference procedure is:

$$\mathbf{y}^* = \arg \max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$$

Inference: Typically, the output space \mathcal{Y} is very large. For example, in the part of speech example above, $|\mathcal{Y}| = 45^m$. Thus, naively enumerating all output sequences and checking their score is not a computationally feasible solution. There are a number of well-known approaches used to perform this search more efficiently. These include dynamic programming, ILPs and beam search. In this thesis, we will often use beam search. The main idea behind beam search is to enumerate all possible assignments to the first output variable and score them, keep the top k assignments and discard the rest. Next, for each assignment of the first variable, enumerate all possible assignments for the second output variable, and again, keep the k highest scoring partial assignments, and discard the rest, and so on. Here k is often referred to as the beam size. Although beam search is an approximate procedure, in practice, it is very effective.

Learning: Given a set of N labeled training examples $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$, the learning problem is to find a weight vector \mathbf{w} such that it scores the target output structures \mathbf{y}_i is higher than other structure \mathbf{y}'_i . This means that we want the value of the score $S(\mathbf{y}_i; \mathbf{w}, \mathbf{x}_i)$ to be higher than $S(\mathbf{y}'_i; \mathbf{w}, \mathbf{x}_i)$ for all $i = 1 \dots N$. This can be readily defined as an optimization problem and a number of instances of the structured prediction paradigm are popular in literature. Let us dive down to a special case of structured prediction – structured prediction with latent variables which is more relevant to this thesis.

3.3 Structured Prediction with Latent Variables

Our structured prediction framework described above has only two types of variables – input variables \mathbf{x} and output variables \mathbf{y} . Typically, both these variables are observed and are given as part of the training data. However, often, we want to model some other information that is not provided as part of the training data. Formally, we additionally also model latent variables $h \in \mathcal{H}$, where \mathcal{H} is the set of feasible assignments to the hidden variables. We extend the structured prediction formulation above to include hidden variables.

We redefine the feature function to also include latent variables (written as $\phi(\mathbf{x}, \mathbf{h}, \mathbf{y})$). For inference, we search for the best output structure and the best assignment to the latent variables as below:

$$(\mathbf{h}^*, \mathbf{y}^*) = \underset{(\mathbf{h}, \mathbf{y}) \in \mathcal{H} \times \mathcal{Y}}{\operatorname{arg\,max}} \quad \mathbf{w}^T \phi(\mathbf{x}, \mathbf{h}, \mathbf{y})$$

Given a set of N labeled training examples $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$, the learning problem is to again find a weight vector \mathbf{w} such that it scores the target output structures \mathbf{y}_i is higher than other structure \mathbf{y}'_i . The learning problem is similar, however, with one key difference that the score of a structure \mathbf{y} is now given as: $S(\mathbf{y}; \mathbf{w}, \mathbf{x}) = \max_{\mathbf{h} \in \mathcal{H}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{h}, \mathbf{y})$.

3.4 Special Cases

In this thesis, we mostly use two key structured prediction models: structural SVM with latent variables [307] and Constraint or rule driven learning [42]. Let us cover them in more detail:

3.4.1 Latent Structure SVM

The structured SVM generalizes the support vector machine classifier for general structured output labels. Structured SVM defines the following objective:

$$\begin{aligned} \min_{\mathbf{w}, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \end{aligned}$$

The Latent Structure SVM further generalizes the structured SVM by incorporating latent variables \mathbf{h} . We again redefine the feature function to also include latent variables (written as $\phi(\mathbf{x}, \mathbf{h}, \mathbf{y})$). For inference, we again search for the best output structure and the best assignment to the latent variables as: $(\mathbf{h}^*, \mathbf{y}^*) = \arg \max_{(\mathbf{h}, \mathbf{y}) \in \mathcal{H} \times \mathcal{Y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{h}, \mathbf{y})$. The latent structure SVM is written down as the following:

$$\begin{aligned} \min_{\mathbf{w}, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & \max_{\mathbf{h}} \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{h}, \mathbf{y}_i) \geq \max_{\mathbf{h}} \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{h}, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \end{aligned}$$

A key property of the above objective is that it can be written as the difference of two convex functions. This allows us to solve the optimization problem using the Concave-Convex Procedure (CCCP) [309]. The CCCP algorithm is guaranteed to decrease the objective function at every iteration and to converge to a local minimum or a saddle point of the objective. The CCCP algorithm applied to Structural SVM with latent variables gives rise to a very intuitive algorithm that alternates between imputing the latent variables \mathbf{h}_i^* that best explain the training pair $(\mathbf{x}_i, \mathbf{y}_i)$ and solving the Structural SVM optimization problem while treating the latent variables as completely observed. For more details, we refer the reader to [307].

3.4.2 Constraint or Rule driven learning

A key issue in machine learning models for NLP is how we can incorporate the domain knowledge from domain experts into our models. There are many ways this problem is expressed [41, 56, 160]. A dominant family of models used in structured prediction are linear models, which can be represented as a weight vector \mathbf{w} , corresponding to a set of feature functions $\{\phi\}$. For an input instance \mathbf{x} and an output assignment \mathbf{y} , the “score” of the instance can be expressed as a weighted sum of feature functions: $f(\mathbf{x}, \mathbf{y}) = \sum_i w_i \phi_i(\mathbf{x}, \mathbf{y})$. Let us focus on *constrained conditional models* (CCM) [42], a classic technique for combining domain knowledge provided as constraints in (typically linear) structured prediction models.

Let us assume that we are presented with the domain knowledge as a set of constraints $\mathcal{C} = \{C_k(\cdot)\}_{k=1}^m, C_k : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ which encode predicates over a pair (\mathbf{x}, \mathbf{y}) . If $C_k(\mathbf{x}, \mathbf{y}) = 1$, it means that the pair (\mathbf{x}, \mathbf{y}) violates the constraint C_k . For each constraint, we are also provided a function $d_{C_k} : \mathcal{X} \times \mathcal{Y} \rightarrow R$ that measures the degree to which the constraint C_k is violated in a pair (\mathbf{x}, \mathbf{y}) . While there are different ways to estimate d_{C_k} , CCMs define what is called a “violation function”. Let $\mathbf{y}_{[1\dots i]} = (y_1, \dots, y_i)$ be a partial assignment of \mathbf{y} . Then, $d_{C_k} = \sum_{i=1}^{|\mathbf{y}|} \hat{C}_k(\mathbf{x}, \mathbf{y}_{[1\dots i]})$ where $C_k(\mathbf{x}, \mathbf{y}_{[1\dots i]})$ is a binary function which indicates whether y_i violates the constraint C_k with respect to the partial assignment. For some of these constraints, the violation cannot be calculated with partial assignments. In these cases, \hat{C}_k returns 0 to indicate that the constraint i is not violated according to the current partial assignment.

A CCM is represented using two weight vectors: the feature weight vector \mathbf{w} and the constraint penalty vector ρ . The score of an output assignment \mathbf{y} for an input \mathbf{x} is given by:

$$f(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n w_i \phi_i(\mathbf{x}, \mathbf{y}) - \sum_{k=1}^m \rho_k d_{c_k}(\mathbf{x}, \mathbf{y})$$

There has been a large body of work in inference methods for constraint driven models including dynamic programming, ILPs and beam search as well as in learning methods which are usually some variants of the EM algorithm. We will skip the details of inference and learning methods and refer them to [42] for more details.

Chapter 4

Answering Reading Comprehension Problems: Alignment as a proxy for Reasoning

Developing an ability to understand natural language is a long-standing goal in NLP and holds the promise of revolutionising the way in which people interact with machines and retrieve information (e.g., for scientific endeavour). To evaluate this ability, we tackle the task of *machine comprehension*. Machine comprehension evaluates a machine’s understanding by posing a series of reading comprehension questions and associated texts, where the answer to each question can be found only in its associated text. In contrast, historically, QA evaluations such as TREC have focused on short factoid questions where simple IR based approaches [198] that treat this as a problem of retrieval from existing knowledge sources followed by some shallow inference tend to do well. In figure 4.1, we show an example of a reading comprehension question from the MCTest dataset [227]. Our goal is to build a model that can answer questions based on the given piece of text.

Factoid question answering contains questions about some factual knowledge such as “Who is the president of the United States?” which can usually be answered by querying the web or existing knowledge tables. We point the interested reader to the TREC¹ and CLEF² evaluations for more details on the prior work on factoid QA. Recently, there has been a resurgence of non-factoid QA in the form of reading comprehensions (QA4MRE evaluations³, MCTest [227] and bAbI [293] datasets are notable examples). Non Factoid QA focuses on answering questions that are not fact based. They require solutions that can “understand” the content rather than using IR style solutions or solutions that use the redundancy of the web to answer questions. This is one of the main reasons for the growing interest in Non Factoid QA.

In this chapter, we present a strategy for learning answer-entailing structures that help us perform inference over much longer texts by treating this as a structured input-output problem. The approach of treating a problem as one of mapping structured inputs to structured outputs is common across many NLP applications. Examples include word or phrase alignment for bitexts

¹<http://trec.nist.gov/>

²<http://nlp.uned.es/clef-qa/>

³<http://nlp.uned.es/clef-qa/repository/qa4mre.php>

Once upon a time there a little girl named Ana. Ana was a smart girl. Everyone in Ana's school knew and liked her very much. She had a big dream of becoming spelling bee winner. Ana studied very hard to be the best she could be at spelling. Ana's best friend would help her study every day after school. By the time the spelling bee arrived Ana and her best friend were sure she would win. **There were ten students in the spelling bee. This made Ana very nervous**, but when she looked out and saw her dad cheering her on she knew she could do it. The spelling bee had five rounds and Ana made it through them all. She was now in the finals. During the final round James, the boy she was in the finals with, was given a really hard word and he spelled it wrong. All Ana had to do was spell this last word and she would be the winner. Ana stepped to the microphone, thought really hard and spelled the word. She waited and finally her teacher said "That is correct". Ana had won the spelling bee. Ana was so happy. She won a trophy. Ana also won a big yellow ribbon. The whole school was also happy, and everyone clapped for her. The whole school went outside. They had a picnic to celebrate Ana winning.

Q: What made Ana very nervous?

- A) The other ten students
- B) Her best friend
- C) The bright lights
- D) The big stage

Figure 4.1: An example reading comprehension question from the MCTest500 dataset.

in MT [21], text-hypothesis alignment in RTE [184, 247, 272, 300], question-answer alignment in QA [20, 303], etc. All of these approaches align local parts of the input to local parts of the output. In this work, we extended the word alignment formalism to align multiple sentences in the text to the hypothesis. We also incorporated the document structure (rhetorical structures [185]) and co-reference information to help us perform inference over longer documents.

QA has had a long history of using pipeline models that extract a limited number of high-level features from induced representations of question-answer pairs, and then build a classifier using some labelled corpora. On the other hand, we learn these structures and the model for answering standardized test questions jointly through a unified max-margin framework. We note that there exist some recent models such as Yih et. al. 2013 that do model QA by automatically defining some kind of alignment between the question and answer snippets and use a similar structured input-output model. However, they are limited to single sentence matching to determine answers.

Another advantage of our approach is its simple and elegant extension to multi-task settings as a way to combine the retrieval and alignment model. There has been a rich vein of work in multi-task learning for SVMs in the ML community. Evgeniou and Pontil 2004 proposed a multi-task SVM formulation assuming that the multi-task predictor \mathbf{w} factorizes as the sum of a shared and a task-specific component. We used the same idea to propose a multi-task variant of Latent Structured SVMs. This allows us to use the single task SVM in the multi-task setting with a different feature mapping. This is much simpler than other competing approaches such as Zhu et. al. 2011 proposed in the literature for multi-task LSSVM. We provide the details below:

4.1 Methodology

Let us first formalize the machine comprehension setup. For each question $q_i \in Q$, let $A_i = \{a_{i1}, \dots, a_{im}\}$ be the set of candidate answers to the question. Let a_i^* be the correct answer. The candidate answers may be pre-defined, as in multiple-choice QA, or may be undefined but easy to extract with a high degree of confidence (e.g., by using a pre-existing system). We want to learn a function $f : (q, \mathcal{K}) \rightarrow a$ that, given a question q_i and background knowledge \mathcal{K} (texts/resources required to answer the question), outputs an answer $\hat{a}_i \in A_i$. We consider a scoring function $S_{\mathbf{w}}(q, a; \mathcal{K})$ (with model parameters \mathbf{w}) and a prediction rule $f_{\mathbf{w}}(q_i) = \hat{a}_i = \arg \max_{a_{ij} \in A_i} S_{\mathbf{w}}(q_i, a_{ij}; \mathcal{K})$. Let $\Delta(\hat{a}_i, a_i^*)$ be the cost of giving a wrong answer. We consider the empirical risk minimization (ERM) framework given a loss function L and a regularizer Ω :

$$\min_{\mathbf{w}} \sum_{q_i \in Q} L_{\mathbf{w}}(a_i^*, f_{\mathbf{w}}(q_i); \mathcal{K}) + \Omega(\mathbf{w}) \quad (4.1)$$

4.1.1 Alignment based approach to Question Answering

In this work, we use alignment based models [303] and cast QA as a textual entailment problem by converting each question-answer candidate pair (q_i, a_{ij}) into a hypothesis statement h_{ij} . For example, the question “What are the important greenhouse gases?” and answer candidate “Carbon dioxide, Methane, Ozone and CFC” can be combined to achieve a hypothesis “The important greenhouse gases are Carbon dioxide , Methane, Ozone and CFC.”. A set of question

matching/rewriting rules are used to achieve this transformation. These rules match the question into one of a large set of pre-defined templates and apply a unique transformation to the question and answer candidate to achieve the hypothesis statement. For each question q_i , the QA task thereby reduces to picking the hypothesis \hat{h}_i that has the highest likelihood among the set of hypotheses $\mathbf{h}_i = \{h_{i1}, \dots, h_{im}\}$ generated for that question of being entailed by a body of relevant texts. The body of relevant texts can vary for each instance of the QA task. For example, it could be just the passage in a reading comprehension task, or a set of science textbooks in the science QA task. Let $h_i^* \in \mathbf{h}_i$ be the correct hypothesis. The model considers the quality of word alignment from a hypothesis h_{ij} (formed by combining question-answer candidates (q_i, a_{ij})) to snippets in the corresponding passage as a proxy for the evidence.

The latent alignment \mathbf{z}_{ij} for the question-answer candidate pair $(q_i, a_{i,j})$ depends on: (a) snippet from the relevant text chosen to be aligned to the hypothesis and (b) word alignment from the hypothesis to the snippet. The snippet from the text to be aligned to the hypothesis is determined by picking a subset of sentences in the text. Then each hypothesis word is aligned to a unique word in the snippet. Learning these alignment edges typically helps a model decompose the input and output structures into semantic constituents and determine which constituents should be compared to each other. These alignments can then be used to generate more effective features.

The choice of the snippet and the word alignment is latent. A natural solution is to treat QA as a problem of ranking the hypothesis set \mathbf{h}_i such that the correct hypothesis is at the top of this ranking. Hence, a scoring function $S_{\mathbf{w}}(h, \mathbf{z})$ is learnt such that the score given to the correct hypothesis h_i^* and the corresponding latent structure \mathbf{z}_i^* is higher than the score given to any other hypothesis and its corresponding latent structure. In fact, in a max-margin fashion, the model learns the scoring function such that $S_{\mathbf{w}}(h_i^*, \mathbf{z}_i^*) > S_{\mathbf{w}}(h_{ij}, \mathbf{z}_{ij}) + \Delta(h_i^*, h_{ij}) - \xi_i$ for all $h_j \in \mathbf{h} \setminus h_i^*$ for some slack ξ_i . This can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\|\mathbf{w}\|} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & S_{\mathbf{w}}(h_i^*, \mathbf{z}_i^*) \geq \max_{\mathbf{z}_{ij}} S_{\mathbf{w}}(h_{ij}, \mathbf{z}_{ij}) + \Delta(h_i^*, h_{ij}) - \xi_i \end{aligned} \quad (4.2)$$

It is intuitive to use the 0-1 cost, i.e. $\Delta(h_i^*, h_{ij}) = \mathbb{1}(h_i^* \neq h_{ij})$. If the scoring function is convex then this objective is in concave-convex form and can be minimized by the concave-convex programming procedure (CCCP) [309]. The scoring function is assumed to be linear: $S_{\mathbf{w}}(h, \mathbf{z}) = \mathbf{w}^T \psi(h, \mathbf{z})$. Here, $\psi(h, \mathbf{z})$ is a task-dependent feature map which will be described later.

4.1.2 Multi-task Learning

Machine comprehension is a complex task which often requires us to interpret questions, the kind of answers they seek as well as the kinds of inference required to solve them. Many approaches in QA [89, 199] solve this by having a top-level classifier that categorizes the complex task into a variety of sub-tasks. The sub-tasks can correspond to various categories of questions that can be asked or various facets of text understanding that are required to do well at machine

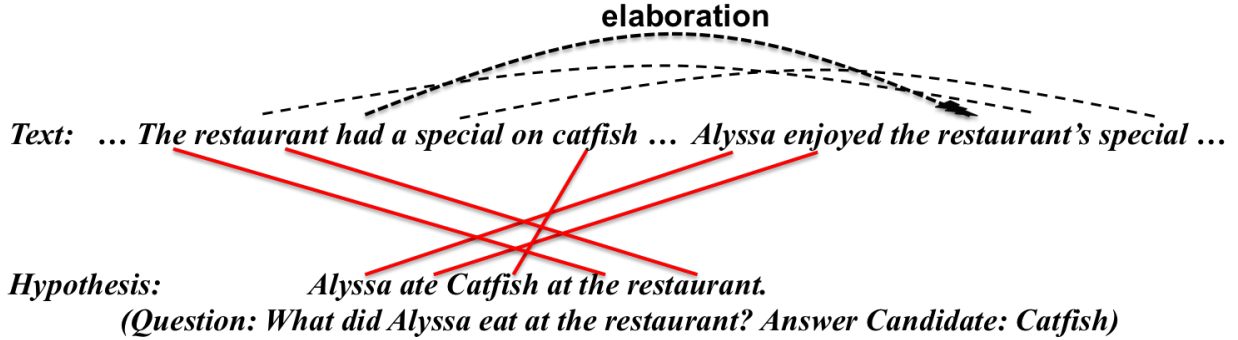


Figure 4.2: The *answer-entailing structure* for an example from MCTest500 dataset. The question and answer candidate are combined to generate a hypothesis sentence. Then latent alignments are found between the hypothesis and the appropriate snippets in the text. The solid red lines show the word alignments from the hypothesis words to the passage words, the dashed black lines show auxiliary co-reference links in the text and the labelled dotted black arrows show the RST relation (elaboration) between the two sentences. Note that the two sentences do not have to be contiguous sentences in the text.

comprehension in its entirety. It is well known that learning a sub-task together with other related sub-tasks leads to a better solution for each sub-task.

Hence, we consider learning classifications of the sub-tasks and then using multi-task learning. We simply extend our LSSVM formulation described above to this multi-task setting. Let S be the number of sub-tasks. We assume that the predictor \mathbf{w} for each subtask s is partitioned into two parts: a parameter \mathbf{w}_0 that is globally shared across each subtasks and a parameter \mathbf{v}_s that is locally used to provide for the variations within the particular subtask: $\mathbf{w} = \mathbf{w}_0 + \mathbf{v}_s$. Mathematically we define the scoring function for hypothesis \mathbf{h} and latent structure \mathbf{z} of the sub-task s to be $Score_{\mathbf{w}_0, \mathbf{v}_s}(h, \mathbf{z}) = (\mathbf{w}_0 + \mathbf{v}_s)^T \psi(h, \mathbf{z})$.

Now, we extend a trick that Evgeiou et. al [78] used for linear SVM to reformulate this problem into an objective that looks like (4.2). Such reformulation will help in using the CCCP algorithm again to solve the multi-task problem as well. Lets define a new feature map ψ_s , one for each sub-task s using the old feature map ψ as:

$$\psi_s(h, \mathbf{z}) = \left(\frac{\psi(h, \mathbf{z})}{\mu}, \underbrace{\mathbf{0}, \dots, \mathbf{0}}_{s-1}, \psi(h, \mathbf{z}), \underbrace{\mathbf{0}, \dots, \mathbf{0}}_{S-s} \right)$$

where $\mu = \frac{S\lambda_1}{\lambda_2}$ and the $\mathbf{0}$ denotes the zero vector of the same size as ψ . Also define our new predictor as $\mathbf{w} = (\sqrt{\mu}\mathbf{w}_0, \mathbf{v}_1, \dots, \mathbf{v}_S)$. Using this formulation we can show that $\mathbf{w}^T \psi_s(h, \mathbf{z}) = (\mathbf{w}_0 + \mathbf{v}_s)^T \psi(h, \mathbf{z})$ and $\|\mathbf{w}\|^2 = \sum_s \|\mathbf{v}_s\|^2 + \mu\|\mathbf{w}_0\|^2$. Hence, if we now define the objective (4.2) but use the new feature map and \mathbf{w} then we will get back our multi-task objective. Thus we can use the same setup as before for multi-task learning after appropriately changing the feature map. We will explore a few definitions of sub-tasks in our experiments.

4.2 Experiments

Answer-entailing structures: Reading comprehensions are very common tests given to students studying languages and form important component in the language portions of standardized tests like the SAT, GRE, GMAT, etc. As described above, our approach models machine comprehension as an extension to textual entailment, learning to output an answer that is best *entailed* by the passage. We use latent answer-entailing structures to estimate the degree of entailment. The answer-entailing structures are closely related to the inference procedure often used in various models for MT [21], RTE [184], paraphrase [301], QA [303], etc. and correspond to the best (latent) alignment between a hypothesis (formed from the question and a candidate answer) with appropriate snippets in the text that are required to answer the question. We consider the quality of a one-to-one word alignment from a hypothesis to snippets in the text as a proxy for the evidence. Hypothesis words are aligned to a unique text word in the text or an empty word. For example, in Figure 4.2, all words but “at” are aligned to a word in the text. The word “at” can be assumed to be aligned to an empty word and it has no effect on the model. Learning these alignment edges typically helps a model decompose the input and output structures into semantic constituents and determine which constituents should be compared to each other. These alignments can then be used to generate more effective features. The alignment depends on two things: (a) snippets in the text to be aligned to the hypothesis and (b) word alignment from the hypothesis to the snippets. We explore three variants of the snippets in the text to be aligned to the hypothesis. The choice of these snippets composed with the word alignment is the resulting hidden structure called an answer-entailing structure.

1. *Sentence Alignment:* The simplest variant is to find a single sentence in the text that best aligns to the hypothesis. This is the structure considered in a majority of previous works in RTE [184] and QA [303] as they only reason on single sentence length texts.

2. *Subset Alignment:* Here we find a subset of sentences from the text (instead of just one sentence) that best aligns with the hypothesis.

3. *Subset+ Alignment:* This is the same as above except that the best subset is an ordered set.

The key difference between the answer-entailing structures considered here and the alignment structures considered in previous works is that we can align multiple sentences in the text to the hypothesis. The sentences in the text considered for alignment are not restricted to occur contiguously in the text. To allow such a dis-contiguous alignment, we make use of the document structure; in particular, we take help from rhetorical structure theory [185] and event and entity coreference links across sentences. Modelling the inference procedure via answer-entailing structures is a crude yet effective and computationally inexpensive proxy to model the semantics needed for the problem. Learning these latent structures can also be beneficial as they can assist a human in verifying the correctness of the answer, eliminating the need to read a lengthy document.

Multi-task Learning: We also extend our LSSVM to multi-task settings using a top-level question-type classification. Many QA systems include a question classification component [166, 314], which typically divides the questions into semantic categories based on the type of the question or answers expected. This helps the system impose some constraints on the plausible answers. Machine comprehension can benefit from such a pre-classification step, not only to constrain plausible answers, but also to allow the system to use different processing strategies

for each category. Recently, [293] defined a set of 20 sub-tasks in the machine comprehension setting, each referring to a specific aspect of language understanding and reasoning required to build a machine comprehension system. They include fact chaining, negation, temporal and spatial reasoning, simple induction, deduction and many more. We use this set to learn to classify questions into the various machine comprehension sub-tasks, and show that this task classification further improves our performance on MCTest. By using the multi-task setting, our learner is able to exploit the commonality among tasks where possible, while having the flexibility to learn task-specific parameters where needed. To the best of our knowledge, this is the first use of multi-task learning in a structured prediction model for QA.

Features: Recall that our features had the form $\psi(\mathbf{t}, h, z)$ where the hypothesis h was itself formed from a question q and answer candidate a . Given an answer-entailing structure z , we induce the following features based on word level similarity of aligned words: (a) Limited word-level surface-form matching and (b) Semantic word form matching: Word similarity for synonymy using SENNA word vectors [57], “Antonymy” ‘Class-Inclusion’ or ‘Is-A’ relations using Wordnet [84]. We compute additional features of the aforementioned kinds to match named entities and events. We also add features for matching local neighborhood in the aligned structure: features for matching bigrams, trigrams, dependencies, semantic roles, predicate-argument structure as well as features for matching global structure: a tree kernel for matching syntactic representations of entire sentences using [268]. The local and global features can use the RST and coreference links enabling inference across sentences. For instance, in the example shown in Figure 4.2, the coreference link connecting the two “restaurant” words brings the snippets “Alyssa enjoyed the” and “had a special on catfish” closer making these features more effective. The answer-entailing structures should be intuitively similar to the question but also the answer. Hence, we add features that are the product of features for the text-question match and text-answer match.

String edit Features: In addition to looking for features on exact word/phrase match, we also add features using two paraphrase databases ParaPara [36] and DIRT [174]. The ParaPara database contains strings of the form $\text{string}_1 \rightarrow \text{string}_2$ like “total lack of” \rightarrow “lack of”, “is one of” \rightarrow “among”, etc. Similarly, the DIRT database contains paraphrases of the form “If \mathbf{X} decreases \mathbf{Y} then \mathbf{X} reduces \mathbf{Y} ”, “If \mathbf{X} causes \mathbf{Y} then \mathbf{X} affects \mathbf{Y} ”, etc. Whenever we have a substring in the text can be transformed into another using these two databases, we keep match features for the substring with a higher score (according to w) and ignore the other substring.

The sentences with discourse relations are related to each other by means of substitution, ellipsis, conjunction and lexical cohesion, etc [185] and can help us answer certain kinds of questions [132]. As an example, the “cause” relation between sentences in the text can often give cues that can help us answer “why” or “how” questions. Hence, we add additional features - conjunction of the RST label and the question word - to our feature vector. Similarly, the entity and event co-reference relations can allow the system to reason about repeating entities or events through all the sentences they get mentioned in. Thus, we add additional features of the aforementioned types by replacing entity mentions with their first mentions.

Subset+ Features: We add an additional set of features which match the first sentence in the ordered set to the question and the last sentence in the ordered set to the answer. This helps in the case when a certain portion of the text is targeted by the question but then it must be used in combination with another sentence to answer the question. For instance, in Figure 4.2, sentence

2 mentions the target of the question but the answer can only be given when in combination with sentence 1.

Negation We empirically found that one key limitation in our formulation is its inability to handle negation (both in questions and text). Negation is especially hurtful to our model as it not only results in poor performance on questions that require us to reason with negated facts, it provides our model with a wrong signal (facts usually align well with their negated versions). We use a simple heuristic to overcome the negation problem. We detect negation (either in the hypothesis or a sentence in the text snippet aligned to it) using a small set of manually defined rules that test for presence of words such as “not”, “n’t”, etc. Then, we flip the partial order - i.e. the correct hypothesis is now ranked below the other competing hypotheses. For inference at test time, we also invert the prediction rule i.e. we predict the hypothesis (answer) that has the least score under the model.

Datasets: We use two datasets for our evaluation.

(1) First is the MCTest-500 dataset⁴, a freely available set of 500 stories (split into 300 train, 50 dev and 150 test) and associated questions [227]. The stories are fictional so the answers can be found only in the story itself. The stories and questions are carefully limited, thereby minimizing the world knowledge required for this task. Yet, the task is challenging for most modern NLP systems. Each story in MCTest has four multiple choice questions, each with four answer choices. Each question has only one correct answer. Furthermore, questions are also annotated with ‘single’ and ‘multiple’ labels. The questions annotated ‘single’ only require one sentence in the story to answer them. For ‘multiple’ questions it should not be possible to find the answer to the question in any individual sentence of the passage. In a sense, the ‘multiple’ questions are harder than the ‘single’ questions as they typically require complex lexical analysis, some inference and some form of limited reasoning. Cucerzan-converted questions can also be downloaded from the MCTest website.

(2) The second dataset is a synthetic dataset released under the *bAbI project*⁵ [293]. The dataset presents a set of 20 ‘tasks’, each testing a different aspect of text understanding and reasoning in the QA setting, and hence can be used to test and compare capabilities of learning models in a fine-grained manner. For each ‘task’, 1000 questions are used for training and 1000 for testing. The ‘tasks’ refer to question categories such as questions requiring reasoning over single/two/three supporting facts or two/three arg. relations, yes/no questions, counting questions, etc. Candidate answers are not provided but the answers are typically constrained to a small set: either yes or no or entities already appearing in the text, etc. We write simple rules to convert the question and answer candidate pairs to hypotheses.⁶

Baselines: We have five baselines. (1) The first three baselines are inspired from [227]. The first baseline (called *SW*) uses a sliding window and matches a bag of words constructed from the question and hypothesized answer to the text. (2) Since this ignores long range dependencies,

⁴<http://research.microsoft.com/mct>

⁵<https://research.facebook.com/researchers/1543934539189348>

⁶Note that the *bAbI* dataset is artificial and not meant for open-domain machine comprehension. It is a toy dataset generated from a simulated world. Due to its restrictive nature, we do not use it directly in evaluating our method vs. other open-domain machine comprehension methods. However, it provides benefit in identifying interesting subtasks of machine comprehension. As will be seen, we are able to leverage the dataset both to improve our multi-task learning algorithm, as well as to analyze the strengths and weaknesses of our model.

the second baseline (called *SW+D*) accounts for intra-word distances as well. As far as we know, *SW+D* is the best previously published result on this task.⁷ (3) The third baseline (called *RTE*) uses textual entailment to answer MCTest questions. For this baseline, MCTest is again re-casted as an RTE task by converting each question-answer pair into a statement (using [59]) and then selecting the answer whose statement has the highest likelihood of being entailed by the story.⁸ (4) The fourth baseline (called *LSTM*) is taken from [293]. The baseline uses LSTMs [121] to accomplish the task. LSTMs have recently achieved state-of-the-art results in a variety of tasks due to their ability to model long-term context information as opposed to other neural networks based techniques. (5) The fifth baseline (called *QANTA*)⁹ is taken from [128]. *QANTA* too uses a recursive neural network for question answering.

Task Classifications: We consider three alternative task classifications for our experiments. First, we look at question classification. We use a simple question classification based on the question word (what, why, what, etc.). We call this QClassification. Next, we also use a question/answer classification¹⁰ from [166]. This classifies questions into different semantic classes based on the possible semantic types of the answers sought. We call this QAClassification. Finally, we also learn a classifier for the 20 tasks in the Machine Comprehension gamut described in [293]. The classification algorithm (called TaskClassification) was built on the *bAbI* training set. It is essentially a Naive-Bayes classifier and uses only simple unigram and bigram features for the question and answer. The tasks typically correspond to different strategies when looking for an answer in the machine comprehension setting. In our experiments we will see that learning these strategies is better than learning the question answer classification which is in turn better than learning the question classification.

Results: We compare multiple variants of our LSSVM¹¹ where we consider a variety of answer-entailing structures and our modification for negation and multi-task LSSVM, where we consider three kinds of task classification strategies against the baselines on the *MCTest* dataset. We consider two evaluation metrics: accuracy (proportion of questions correctly answered) and NDCG₄ [133]. Unlike classification accuracy which evaluates if the prediction is correct or not, NDCG₄, being a measure of ranking quality, evaluates the position of the correct answer in our predicted ranking.

Figure 4.3 describes the comparison on *MCTest*. We can observe that all the LSSVM models have a better performance than all the five baselines (including LSTMs and RNNs which are state-of-the-art for many other NLP tasks) on both metrics. Very interestingly, LSSVMs have a considerable improvement over the baselines for “multiple” questions. We posit that this is because of our answer-entailing structure alignment strategy which is a weak proxy to the deep semantic inference procedure required for machine comprehension. The RTE baseline achieves

⁷We also construct two additional baselines (*LSTM* and *QANTA*) for comparison in this paper both of which achieve superior performance to *SW+D*.

⁸The BIUTEE system [270] available under the Excitement Open Platform <http://hlfbk.github.io/Excitement-Open-Platform/> was used for recognizing textual entailment.

⁹<http://cs.umd.edu/miyyer/qblearn/>

¹⁰<http://cogcomp.cs.illinois.edu/Data/QA/QC/>

¹¹We tune the SVM regularization parameter *C* and the penalty factor on the subset size on the development set. We use a beam of size 5 in our experiments. We use Stanford CoreNLP and the HILDA parser [86] for linguistic preprocessing.

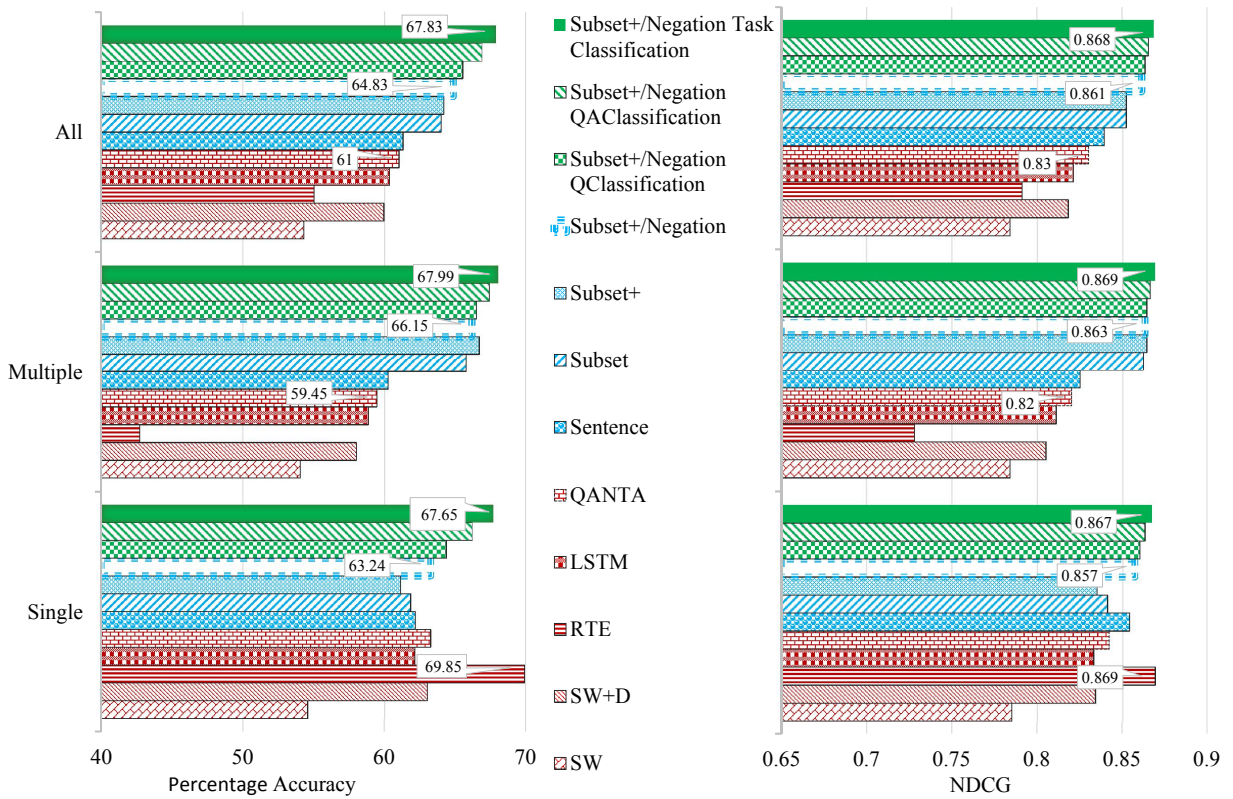


Figure 4.3: Comparison of variations of our method against several baselines on the MCTest-500 dataset. The figure shows two statistics, accuracy (on the left) and $NDCG_4$ (on the right) on the test set of MCTest-500. All differences between the baselines and LSSVMs, the improvement due to negation and the improvements due to multi-task learning are significant ($p < 0.01$) using the two-tailed paired T-test. The exact numbers are available in the supplementary.

the best performance on the “single” questions. This is perhaps because the RTE community has almost entirely focused on single sentence text hypothesis pairs for a long time. However, RTE fares pretty poorly on the “multiple” questions indicating that of-the-shelf RTE systems cannot perform inference across large texts.

Figure 4.3 also compares the performance of LSSVM variants when various answer-entailing structures are considered. Here we observe a clear benefit of using the alignment to the best subset structure over alignment to best sentence structure. We furthermore see improvements when the best subset alignment structure is augmented with the subset+ features. We can observe that the negation heuristic also helps, especially for “single” questions (majority of negation cases in the *MCTest* dataset are for the “single” questions).

It is also interesting to see that the multi-task learners show a substantial boost over the single task SSVM. Also, it can be observed that the multi-task learner greatly benefits if we can learn a separation between the various strategies needed to learn an overarching list of subtasks required to solve the machine comprehension task.¹² The multi-task method (TaskClassification) which uses the Weston style categorization does better than the multi-task method (QAClassification) that learns the question answer classification. QAClassification in turn performs better than multi-task method (QClassification) that learns the question classification only.

Analysis: A good question to be asked is how good is structure alignment as a proxy to the semantics of the problem? In this section, we attempt to tease out the strengths and limitations of such a structure alignment approach for machine comprehension. To do so, we evaluate our methods on various tasks in the *bAbI* dataset. For the *bAbI* dataset, we add additional features inspired from the “task” distinction to handle specific “tasks”.

Table 4.1 shows the results of various LSSVM models on the *bAbI* datasets for each sub-task. In our experiments, we observed a similar general pattern of improvement of LSSVM over the baselines as well as the improvement due to multi-task learning. Again task classification helped the multi-task learner the most and the QA classification helped more than the QClassification. It is interesting here to look at the performance within the sub-tasks. Negation improved the performance for three sub-tasks, namely, the tasks of modelling “yes/no questions”, “simple negations” and “indefinite knowledge” (the “Indefinite Knowledge” sub-task tests the ability to model statements that describe possibilities rather than certainties). Each of these sub-tasks contain a significant number of negation cases. Our models do especially well on questions requiring reasoning over one and two supporting facts, two arg. relations, indefinite knowledge, basic and compound coreference and conjunction. Our models achieve lower accuracy better than the baselines on two sub-tasks, namely “path finding” and “agent motivations”. Our model along with the baselines do not do too well on the “counting” sub-task, although we get slightly better scores. The “counting” sub-task (which asks about the number of objects with a certain property) requires the inference to have an ability to perform simple counting operations. The “path finding” sub-task requires the inference to reason about the spatial path between locations (e.g. Pittsburgh is located on the west of New York). The “agent’s motivations” sub-task asks questions such as ‘why an agent performs a certain action’. As inference is cheaply modelled

¹²Note that this is despite the fact that the classifier is not learned on the *MCTest* dataset but the *bAbI* dataset! This hints at the fact that the task classification proposed in [293] is more general and broadly also makes sense for other machine comprehension settings such as *MCTest*.

Tasks	Baselines				LSSVM						
	SW	RTE	LSTM	QANTA	Sentence	Subset	Subset+	Subset+/-Negation	MultiTask		
									Subset+/-Negation QClassification	Subset+/-Negation QAClassification	Subset+/-Negation TaskClassification
Single Supporting Fact	36	98	50	89	100	100	100	100	100	100	100
Two Supporting Facts	2	79	20	69	60	91	92	91	93	93	94
Three Supporting Facts	7	46	20	42	52	84	86	84	86	87	88
Two Arg. Relations	50	54	61	68	89	91	91	90	92	93	93
Three Arg. Relations	20	31	70	63	84	89	89	88	91	90	91
Yes/No Questions	49	48	48	54	58	58	58	78	81	84	85
Counting	52	11	49	55	61	59	63	61	65	64	64
Lists/Sets	42	34	45	47	55	72	73	71	77	80	82
Simple Negation	62	56	64	72	63	63	64	76	79	80	81
Indefinite Knowledge	45	43	44	68	74	74	78	87	88	91	92
Basic Coreference	25	31	72	80	91	93	96	96	97	97	98
Conjunction	9	59	74	86	94	91	91	90	95	96	97
Compound Coreference	26	72	94	95	86	89	89	88	93	93	94
Time Reasoning	19	68	27	43	65	68	70	68	71	74	76
Basic Deduction	20	49	21	72	76	74	78	76	80	81	82
Basic Induction	43	53	23	55	57	59	61	58	61	63	64
Positional Reasoning	46	66	51	55	81	85	88	88	90	91	90
Size Reasoning	52	77	52	63	78	82	84	83	85	87	89
Path Finding	0	11	8	45	9	9	9	9	11	11	11
Agent’s Motivations	76	91	91	93	66	69	70	68	69	69	70
Mean Performance	34	54	49	66	70	75	77	78	79	81	82

Table 4.1: Comparison of accuracies on the variations of our method against several baselines on 20 Tasks of the bAbI dataset. All integer differences are significant ($p < 0.01$) using the two-tailed paired T-test.

via alignment structure, we lack the ability to deeply reason about facts or numbers. This is an important challenge for future work.

4.3 Conclusion

In this chapter, we addressed the problem of machine comprehension which tests language understanding through multiple choice question answering tasks. We posed the task as an extension to RTE. Then, we proposed a solution by learning latent alignment structures between texts and the hypotheses in the equivalent RTE setting. The task requires solving a variety of sub-tasks so we extended our technique to a multi-task setting. Our technique showed empirical improvements over various IR and neural network baselines. The latent structures while effective are cheap proxies to the reasoning and language understanding required for this task and have their own limitations. We also discuss strengths and limitations of our model in a more fine-grained analysis. In the next chapters, we address some of these issues by exploring logic-like semantic representations of texts, questions and answers and explore approaches that perform structured inference over richer semantic representations.

Chapter 5

AMR as a Semantic Representation

Learning to efficiently represent and reason with natural language is a fundamental yet long-standing goal in NLP. In the previous section, we used a suite of linguistic features derived from POS taggers, parse trees, semantic role labellers, etc. This raises a fundamental question. Can we choose a better representation of language and improve our model. Recently, there have been a series of efforts in broad-coverage semantic representation (or “semlanking”). AMR, a new semantic representation in standard neo-Davidsonian [62, 213] framework has been proposed. AMRs are rooted, labeled graphs which incorporate PropBank style semantic roles, within-sentence coreference, named entities and the notion of types, modality, negation, quantification, etc. in one framework.

In this chapter, we ask the question if richer meaning representations such as the AMR representation could help us in the task of machine comprehension. Our approach again models machine comprehension as an extension to textual entailment, learning to output an answer that is best *entailed* by the passage. It works in two stages. First, we construct a meaning representation graph for the entire passage from the AMR graphs of comprising sentences. To do this, we account for cross-sentence linguistic phenomena such as entity and event coreference, and rhetorical structures. A similar meaning representation graph is also constructed for each question-answer pair. Once we have these graphs, the comprehension task henceforth can be reduced to a graph containment problem. We posit that there is a latent subgraph of the text meaning representation graph (called snippet graph) and a latent alignment of the question-answer graph onto this snippet graph that *entails* the answer (see Figure 5.1 for an example). Then, our unified max-margin model jointly learns the latent structure (subgraph selection and alignment) and the QA model.

5.1 The AMR Meaning Representation Graph

We construct the meaning representation graph using individual sentences’ AMR graphs and merging identical concepts (using entity and event coreference). First, for each sentence AMR, we merge nodes corresponding to multi-word expressions and nodes headed by a date entity (“date-entity”), or a named entity (“name”) or a person entity (“person”). For example, the hypothesis meaning representation graph in Figure 5.1 was achieved by merging the AMR parse

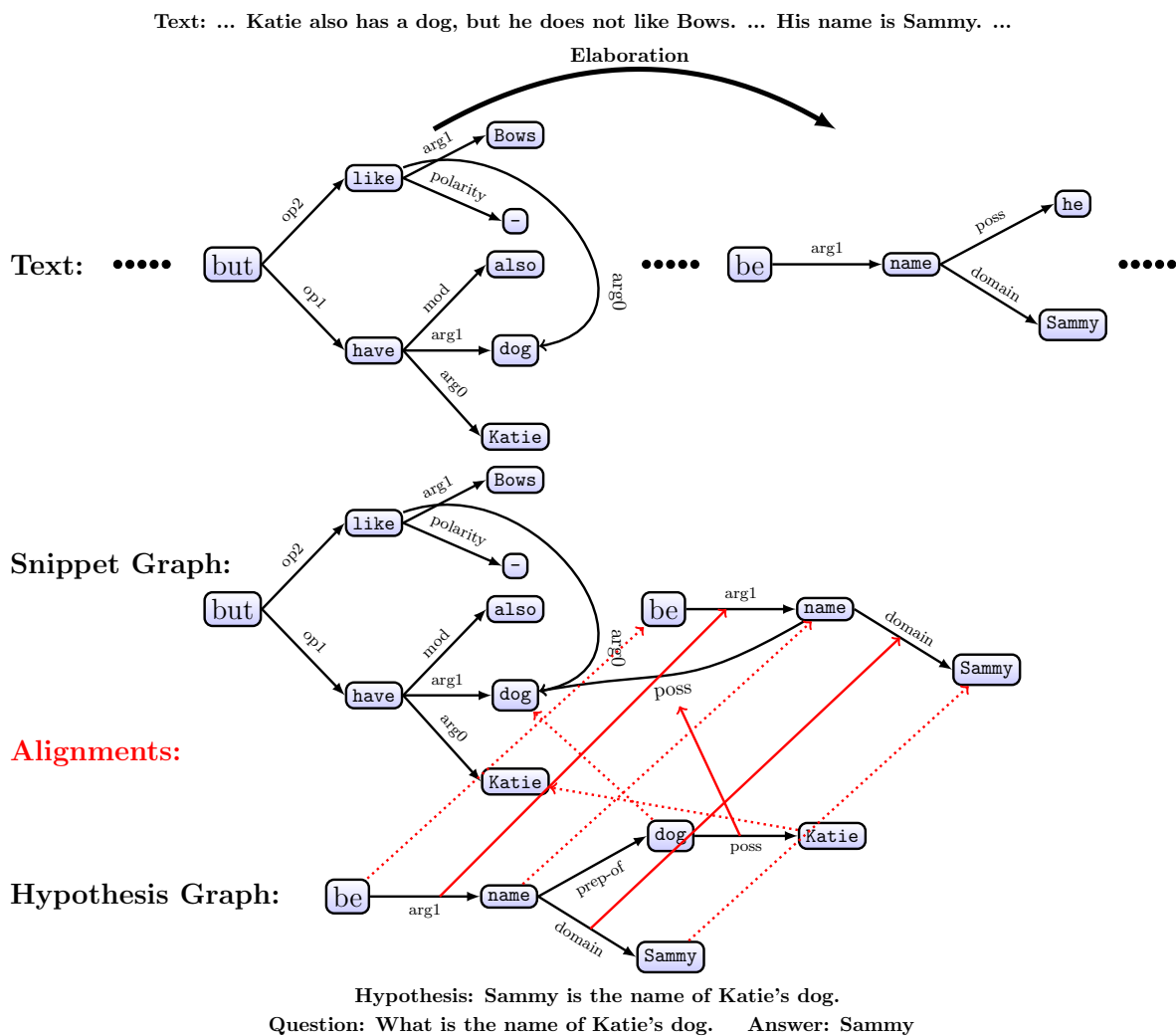


Figure 5.1: Example latent *answer-entailing* structure from the MCTest dataset. The question and answer candidate are combined to generate a hypothesis. This hypothesis is AMR parsed to construct a hypothesis meaning representation graph after some post-processing (section 5.1). Similar processing is done for each sentence in the passage as well. Then, a subset (not necessarily contiguous) of these sentence meaning representation graphs is found. These representation subgraphs are further merged using coreference information, resulting into a structure called the relevant text snippet graph. Finally, the hypothesis meaning representation graph is aligned to the snippet graph. The dashed red lines show node alignments, solid red lines show edge alignments, and thick solid black arrow shows the rhetorical structure label (elaboration).

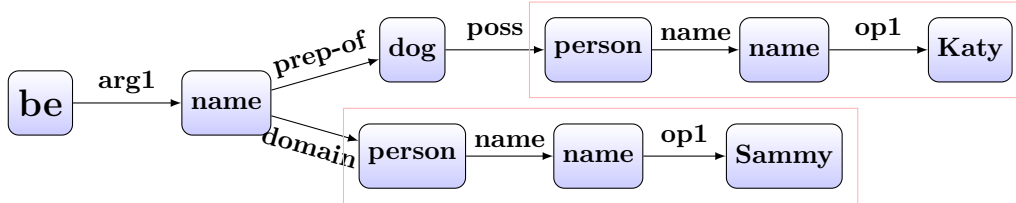


Figure 5.2: AMR parse for the hypothesis in Figure 5.1. The person nodes are merged to achieve the hypothesis meaning representation graph.

shown in Figure 5.2.

Next, we select the subset of sentence AMRs corresponding to sentences needed to answer the question. This step uses cross-sentential phenomena such as rhetorical structures¹ and entities/event coreference. The coreferent entities/event mentions are further merged into one node resulting in a graph called the relevant text snippet graph. A similar process is also performed with the hypothesis sentences (generated by combining the question and answer candidate) as shown in Figure 5.1.

5.2 Redefined Scoring Function and Inference

We use the same latent structural SVM as described before in our setup. The only difference is that in this case, we redefine the scoring function $S_{\mathbf{w}}(h, \mathbf{z})$. Let the hypothesis meaning representation graph be $G' = (V', E')$. Our latent structure \mathbf{z} decomposes into the selection (\mathbf{z}_s) of relevant sentences that lead to the text snippet graph G , and the mapping (\mathbf{z}_m) of every node and edge in G' onto G . We define the score such that it factorizes over the nodes and edges in G' . The weight vector \mathbf{w} also has three components \mathbf{w}_s , \mathbf{w}_v and \mathbf{w}_e corresponding to the relevant sentences selection, node matches and edge matches respectively. An edge in the graph is represented as a triple (v^1, r, v^2) consisting of the endpoint vertices and relation r .

$$S_{\mathbf{w}}(h, \mathbf{z}) = \mathbf{w}_s^T \mathbf{f}(G', G, \mathbf{t}, h, \mathbf{z}_s) + \sum_{v' \in V'} \mathbf{w}_v^T \mathbf{f}(v', z_m(v')) + \sum_{e' \in E'} \mathbf{w}_e^T \mathbf{f}(e', z_m(e'))$$

Here, \mathbf{t} is the text corresponding to the hypothesis h , and \mathbf{f} are parts of the feature map ψ to be described later. $z(v')$ maps a node $v' \in V'$ to a node in V . Similarly, $z(e')$ maps an edge $e' \in E'$ to an edge in E .

Next, we describe the inference procedure i.e. how to select the structure that gives the best score for a given hypothesis. The inference is performed in two steps: The first step selects the relevant sentences from the text. This is done by simply maximizing the first part of the score:

¹Rhetorical structure theory [185] tells us that sentences with discourse relations are related to each other. Previous works in QA [132] have shown that these relations can help us answer certain kinds of questions. As an example, the “cause” relation between sentences in the text can often give cues that can help us answer “why” or “how” questions. Hence, the passage meaning representation also remembers RST relations between sentences.

$\mathbf{z}_s = \arg \max_{\mathbf{z}_s} \mathbf{w}_s^T \mathbf{f}(G', G, \mathbf{t}, h, \mathbf{z}_s)$. Here, we only consider subsets of 1, 2 and 3 sentences as most questions can be answered by 3 sentences in the passage. The second step is formulated as an integer linear program by rewriting the scoring function. The ILP objective is:

$$\sum_{v' \in V'} \sum_{v \in V} z_{v',v} \mathbf{w}_v^T \mathbf{f}(v', v) + \sum_{e' \in E'} \sum_{e \in E} z_{e',e} \mathbf{w}_e^T \mathbf{f}(e', e)$$

Here, with some abuse of notation, $z_{v',v}$ and $z_{e',e}$ are binary integers such that $z_{v',v} = 1$ iff \mathbf{z} maps v' onto v else $z_{v',v} = 0$. Similarly, $z_{e',e} = 1$ iff \mathbf{z} maps e' onto e else $z_{e',e} = 0$. Additionally, we have the following constraints to our ILP:

- Each node $v' \in V'$ (or each edge $e' \in E'$) is mapped to exactly one node $v \in V$ (or one edge $e \in E$). Hence: $\sum_{v \in V} z_{v',v} = 1 \quad \forall v'$ and $\sum_{e \in E} z_{e',e} = 1 \quad \forall e'$
- If an edge $e' \in E'$ is mapped to an edge $e \in E$, then vertices $(v_{e'}^1, v_{e'}^2)$ that form the end points of e' must also be aligned to vertices (v_e^1, v_e^2) that form the end points of e . Here, we note that AMR parses also have inverse relations such as “arg0-of”. Hence, we resolve this with a slight modification. If neither or both relations (corresponding to edges e' and e) are inverse relations (case 1), we enforce that $v_{e'}^1$ align with v_e^1 and $v_{e'}^2$ align with v_e^2 . If exactly one of the relations is an inverse relation (case 2), we enforce that $v_{e'}^1$ align with v_e^2 and $v_{e'}^2$ align with v_e^1 . Hence, we introduce the following constraints:

$$\begin{aligned} z_{e'e} &\leq z_{v_e^1, v_{e'}^1} \text{ and } z_{e'e} \leq z_{v_e^2, v_{e'}^2} && \forall e'. e \text{ in case 1} \\ z_{e'e} &\leq z_{v_e^1, v_{e'}^2} \text{ and } z_{e'e} \leq z_{v_e^2, v_{e'}^1} && \forall e'. e \text{ in case 2} \end{aligned}$$

Features: Our feature function decomposes into three parts, each corresponding to a part of the latent structure.

The first part corresponds to relevant sentence selection. Here, we include features for matching local neighborhoods in the sentence subset and the hypothesis: features for matching bi-grams, trigrams, dependencies, semantic roles, predicate-argument structure as well as the global syntactic structure: a graph kernel for matching AMR graphs of entire sentences [268]. Before computing the graph kernel, we reverse all inverse relation edges in the AMR graph. Note that if a sentence subset contains the answer to the question, it should intuitively be similar to the question as well as to the answer. Hence, we add features that are the element-wise product of features for the subset-question match and subset-answer match. In addition to features for the exact word/phrase match of the snippet and the hypothesis, we also add features using two paraphrase databases: ParaPara [36] and DIRT [174]. These databases contain paraphrase rules of the form $\text{string}_1 \rightarrow \text{string}_2$. ParaPara rules were extracted through bilingual pivoting and DIRT rules were extracted using the distributional hypothesis. Whenever we have a substring in the text snippet that can be transformed into another using any of these two databases, we keep match features for the substring with a higher score (according to the current \mathbf{w}) and ignore the other substring. Finally, we also have features corresponding to the RST [185] links to enable inference across sentences. RST tells us that sentences with discourse relations are related to each other and can help us answer certain kinds of questions [132]. For example, the “cause” relation

between sentences in the text can often give cues that can help us answer “why” or “how” questions. Hence, we have additional features - conjunction of the rhetorical structure label from a RST parser and the question word as well.

The second part corresponds to node matches. Here, we have features for (a) Surface-form match (Edit-distance), and (b) Semantic word match (cosine similarity using SENNA word vectors [57] and “Antonymy” ‘Class-Inclusion’ or ‘Is-A’ relations using Wordnet).

The third part corresponds to edge matches. Let the edges be $e = (v^1, r, v^2)$ and $e' = (v'^1, r', v'^2)$ for notational convenience. Here, we introduce two features based on the relations - indicator that the two relations are the same or inverse of each other, indicator that the two relations are in the same relation category – categories as described in [14]. Then, we introduce a number of features based on distributional representation of the node pairs. We compute three vertex vector compositions (sum, difference and product) of the nodes for each edge proposed in recent representation learning literature in NLP [194, 196] i.e. $v^1 \odot v^2$ and $v'^1 \odot v'^2$ for $\odot = \{+, -, \times\}$. Then, we compute the cosine similarities of the resulting compositions producing three features. Finally we introduce features based on the structured distributional semantic representation [16, 75, 102] which takes the relations into account while performing the composition. Here, we use a large text corpora (in our experiments, the English Wikipedia) and construct a representation matrix $M^{(r)} \subset V \times V$ for every relation r (V is the vocabulary) where, the ij^{th} element $M_{ij}^{(r)}$ has the value $\log(1 + x)$ where x is the frequency for the i^{th} and j^{th} vocabulary items being in relation r in the corpora. This allows us to compose the node and relation representations and compare them. Here we compute the cosine similarity of the compositions $(v^1)^T M^{(r)}$ and $(v'^1)^T M^{(r')}$, the compositions $M^{(r)} v^2$ and $M^{(r')} v'^2$ and their respective sums $(v^1)^T M^{(r)} + M^{(r)} v^2$ and $(v'^1)^T M^{(r')} + M^{(r')} v'^2$ to get three more features.

5.3 Experiments

Datasets: We use MCTest-500 dataset [227], a freely available set of 500 stories (300 train, 50 dev and 150 test) and associated questions to evaluate our model. Each story in MCTest has four multiple-choice questions, each with four answer choices. Each question has exactly one correct answer. Each question is also annotated as ‘single’ or ‘multiple’. The questions annotated ‘single’ require just one sentence in the passage to answer them. For ‘multiple’ questions it should not be possible to find the answer to the question with just one sentence of the passage. In a sense, ‘multiple’ questions are harder than ‘single’ questions as they require more complex inference. We will present the results breakdown for ‘single’ or ‘multiple’ category questions as well.

Baselines: We compare our approach to the following baselines: (1-3) The first three baselines are taken from [227]. *SW* and *SW+D* use a sliding window and match a bag of words constructed from the question and the candidate answer to the text. *RTE* uses textual entailment by selecting the hypothesis that has the highest likelihood of being entailed by the passage. (4) *LEX++*, taken from [265] is another lexical matching method that takes into account multiple context windows, question types and coreference. (5) *JACANA* uses an off the shelf aligner and aligns the hypothesis statement with the passage. (6-7) *LSTM* and *QANTA*, taken from [239], use neural networks (LTSMs and Recursive NNs, respectively). (8) *ATTENTION*, taken from [306], uses

		Single	Multiple	All	
AMR +MTL	Subgraph	67.28	65.24	66.16	
	Subgraph+Negation	69.48	66.46	67.83	
	QClassification	70.59	67.99	69.17	
	QAClassification	71.32	68.29	69.67	
	TaskClassification	72.05	68.90	70.33	
Baselines	SW	54.56	54.04	54.28	
	SW+D	62.99	58.00	60.26	
	RTE	69.85	42.71	55.01	
	LEX++	69.12	63.34	65.96	
	JACANA Aligner	58.82	54.88	56.67	
	LSTM	62.13	58.84	60.33	
	QANTA	63.23	59.45	61.00	
	ATTENTION	54.20	51.70	52.90	
	DISCOURSE	68.38	59.90	63.75	
	LSSVM	61.12	66.67	64.15	
	LSSVM+Negation	63.24	66.15	64.83	
	+MTL	QClassification	64.34	66.46	65.50
		QAClassification	66.18	67.37	66.83
		TaskClassification	67.65	67.99	67.83
		SYN+FRM+SEM	72.05	67.94	69.94

Table 5.1: Comparison of variations of our method against several baselines on the MCTest-500 dataset. The table shows accuracy on the test set of MCTest-500. All differences between the baselines (except SYN+FRM+SEM) and our approaches, and the improvements due to negation and multi-task learning are significant ($p < 0.05$) using the two-tailed paired T-test.

an attention-based convolutional neural network. (9) *DISCOURSE*, taken from [202], proposes a discourse based model. (10-14) *LSSVM*, *LSSVM+Negation*, *LSSVM+Negation (MultiTask)*, taken from [239] are all discourse aware latent structural svm models. *LSSVM+Negation* accounts for negation. *LSSVM+Negation+MTL* further incorporates multi-task learning based on question types. Here, we have three variants of multitask learners based on the three question classification strategies. (15) Finally, *SYN+FRM+SEM*, taken from [284] proposes a framework with features based on syntax, frame semantics, coreference and word embeddings.

Results: We compare our AMR subgraph containment approach² where we consider our modifications for negation and multi-task learning as well in Table 5.1. We can observe that our models have a comparable performance to all the baselines including the neural network approaches and all previous approaches proposed for this task. Further, when we incorporate multi-task learning, our approach achieves the state of the art. Also, our approaches have a considerable improvement over the baselines for ‘multiple’ questions. This shows the benefit of our latent structure that allows us to combine evidence from multiple sentences. The negation heuristic helps signif-

²We tune the SVM parameter C on the dev set. We use Stanford CoreNLP, HILDA parser [86] and JAMR [92] for preprocessing.

icantly, especially for ‘single’ questions (majority of negation cases in the *MCTest* dataset are for the “single” questions). The multi-task method which performs a classification based on the sub-tasks for machine comprehension defined in [293] does better than QAClassification that learns the question answer classification. QAClassification in turn performs better than QClassification that learns the question classification only.

5.4 Conclusion

Our results, together, provide validation for our approach of subgraph matching over meaning representation graphs, and the incorporation of negation and multi-task learning. These results also provide evidence that good richer meaning representations of language such as AMR can indeed be useful in comprehension tasks such as machine comprehension. This motivates us to seek richer meaning representations and to find efficient and accurate ways to map language to these complex meaning representations.

Chapter 6

Science Question Answering from Instructional Materials: Incorporating External Knowledge in the Alignment Model

A key aspect in language comprehension is external knowledge. Accurately capturing background knowledge of the world and then using it in downstream language comprehension tasks is a key goal of natural language processing. As a way to push the community towards this goal, Elementary Science tests [50] have been proposed by the Allen Institute of AI¹ as a grand challenge to the research community under their Aristo project². Successfully solving these tests really requires building models that can understand language but also capture and use rich world knowledge. These tests comprise of English language questions from the domain of elementary school science that span several grade levels. Each question is a 4-way multiple choice structure and the goal is to build a system that acquires and stores a vast amount of knowledge in computable form, then applies this knowledge to answer these science questions. These tests are quite challenging because of a wide variety of knowledge and reasoning required to answer these questions. Despite significant interest in the recent years [53, 141, 167], the problem remains unsolved.

A typical elementary science question is shown in Figure 6.1. Answering the question “Which of the following gases cause the greenhouse effect?” and the (correct) answer candidate “ CO_2 , CH_4 , O_3 and CFC ” requires the model to seek evidence for answering the question in a large corpus of textbooks but also a number of external knowledge sources such as periodic tables, dictionaries, etc. which can provide us with useful facts such as CO_2 is the chemical symbol for “carbon dioxide” and that “greenhouse gases cause greenhouse effect”.

Thus, in order to handle retrieval from a large corpus of textbooks and external knowledge sources, we extended our *answer-entailing modelling* approach in the task of answering multiple-choice elementary science tests [50]. In this case, our approach learns latent *answer-entailing*

¹<https://allenai.org/>

²<https://allenai.org/aristo/>

structures that align question-answers with appropriate snippets in the curriculum or any external available knowledge sources. The student curriculum usually comprises of a set of textbooks. Each textbook, in-turn comprises of a set of chapters, each chapter is further divided into sections – each discussing a particular science concept. Hence, the answer-entailing structure consists of selecting a particular textbook from the curriculum, picking a chapter in the textbook, picking a section in the chapter, picking a few sentences in the section and then aligning words/multi-word expressions (mwe’s) in the hypothesis (formed by combining the question and an answer candidate) to words/mwe’s in the picked sentences. The answer-entailing structures are further refined using external domain-specific knowledge resources such as science dictionaries, study guides and semi-structured tables (see Figure 6.1). These domain-specific knowledge resources can be very useful forms of knowledge representation as shown in previous works [53].

In this case, we incorporate the curriculum hierarchy (i.e. the book, chapter, section bifurcation) into the latent structure. This helps us jointly learn the retrieval and answer selection modules of a QA system. Retrieval and answer selection are usually designed as isolated or loosely connected components in QA systems [89] leading to loss in performance – our approach mitigates this shortcoming. We also utilize domain-specific knowledge sources such as study guides, science dictionaries or semi-structured knowledge tables within our model.

The joint model is again trained in max-margin fashion using a latent structural SVM (LSSVM) where the answer-entailing structures are latent. We train and evaluate our models on a set of 8th grade science problems, science textbooks and multiple domain-specific knowledge resources. We achieve superior performance vs. a number of baselines.

6.1 The Answer-Entailing Structure

For the science question answering from instructional material, our latent answer-entailing structure depends on: (a) snippet from the curriculum hierarchy chosen to be aligned to the hypothesis, (b) external knowledge relevant for this entailment, and (c) the word/mwe alignment. The snippet from the curriculum to be aligned to the hypothesis is determined by walking down the curriculum hierarchy and then picking a set of sentences from the section chosen. Then, a subset of relevant external knowledge in the form of triples and equivalences (called knowledge bits) is selected from our reservoir of external knowledge (science dictionaries, cheat sheets, semi-structured tables, etc). Finally, words/mwe’s in the hypothesis are aligned to words/mwe’s in the snippet or knowledge bits. Learning these alignment edges helps the model determine which semantic constituents should be compared to each other. These alignments are also used to generate more effective features. The choice of snippets, choice of the relevant external knowledge and the alignments in conjunction form the latent answer-entailing structure.

6.2 Inference and knowledge selection

We use beam search with a fixed beam size (5) for inference. We infer the textbook, chapter, section, snippet and alignments one by one in this order. In each step, we only expand the five most promising (given by the current score) substructure candidates so far. During inference, we

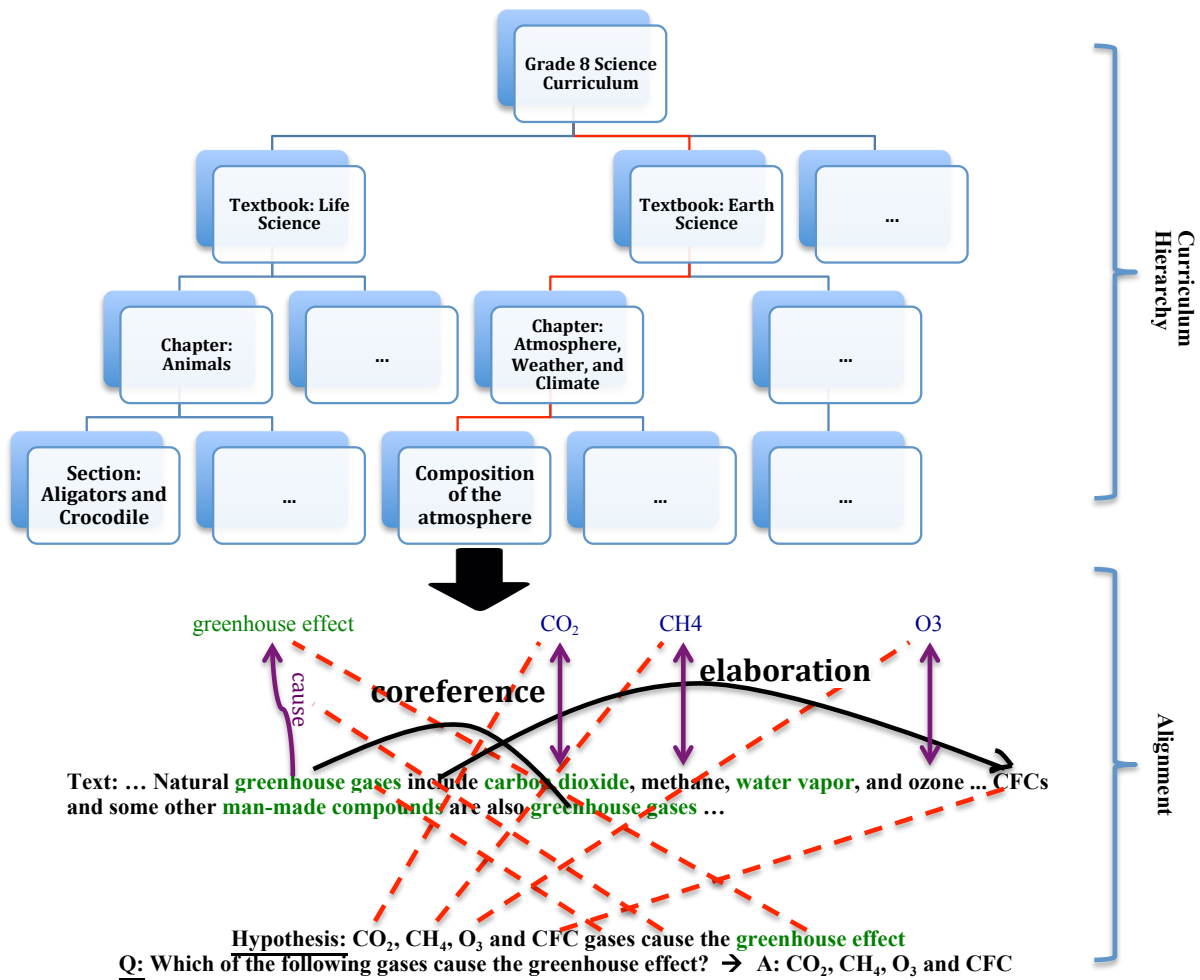


Figure 6.1: An example *answer-entailing structure* for science question answering. The answer-entailing structure consists of selecting a particular textbook from the curriculum, picking a chapter in the textbook, picking a section in the chapter, picking sentences in the section and then aligning words/mwe's in the hypothesis (formed by combining the question and an answer candidate) to words/mwe's in the picked sentences or some related “knowledge” appropriately chosen from additional knowledge stores. In this case, the relation (greenhouse gases, cause, greenhouse effect) and the equivalences (e.g. carbon dioxide = CO₂) – shown in violet – are hypothesized using external knowledge resources. The dashed red lines show the word/mwe alignments from the hypothesis to the sentences (some word/mwe are not aligned, in which case the alignments are not shown), the solid black lines show coreference links in the text and the RST relation (elaboration) between the two sentences. The picked sentences do not have to be contiguous sentences in the text. All mwe's are shown in green.

select top 5 knowledge bits (triples, equivalences, etc.) from the knowledge resources that could be relevant for this question-answer. This is done heuristically by picking knowledge bits that explain parts of the hypothesis not explained by the chosen snippets.

Incorporating partially known structures: As described earlier, modern textbooks often provide review problems at the end of each section. These review problems have value as part of the answer-entailing structure (textbook, chapter and section) is known for these problems. In this case, we use the formulation (equation 1) except that the max over \mathbf{z} for the review questions is only taken over the unknown part of the latent structure.

6.3 Features

Our feature vector $\psi(h, \mathbf{z})$ decomposes into five parts, where each part corresponds to a part of the answer-entailing structure. For the first part, we index all the textbooks and score the top retrieved textbook by querying the hypothesis statement. We use tf-idf and BM25 scorers resulting in two features. Then, we find the jaccard similarity of bigrams and trigrams in the hypothesis and the textbook to get two more features for the first part. Similarly, for the second part we index all the textbook chapters and compute the tf-idf, BM25 and bigram, trigram features. For the third part we index all the sections instead. The fourth part has features based on the text snippet part of the answer-entailing structure. Here we do a deeper linguistic analysis and include features for matching local neighborhoods in the snippet and the hypothesis: features for matching bigrams, trigrams, dependencies, semantic roles, predicate-argument structure as well as the global syntactic structure: a tree kernel for matching dependency parse trees of entire sentences [268]. If a text snippet contains the answer to the question, it should intuitively be similar to the question as well as to the answer. Hence, we add features that are the element-wise product of features for the text-question match and text-answer match. Finally, we also have features corresponding to the RST [185] and coreference links to enable inference across sentences. RST tells us that sentences with discourse relations are related to each other and can help us answer certain kinds of questions [131]. For example, the “cause” relation between sentences in the text can often give cues that can help us answer “why” or “how” questions. Hence, we add additional features - conjunction of the rhetorical structure label from a RST parser and the question word - to our feature vector. Similarly, the entity and event co-reference relations allow us to reason about repeating entities or events. Hence, we replace an entity/event mention with their first mentions if that results into a greater score. For the alignment part, we induce features based on word/mwe level similarity of aligned words: (a) Surface-form match (Edit-distance), and (b) Semantic word match (cosine similarity using SENNA word vectors [57] and “Antonymy” ‘Class-Inclusion’ or ‘Is-A’ relations using Wordnet). Distributional vectors for mwe’s are obtained by adding the vector representations of comprising words [196]. To account for the hypothesized knowledge bits, whenever we have the case that a word/mwe in the hypothesis can be aligned to a word/mwe in a hypothesized knowledge bit to produce a greater score, then we keep the features for the alignment with the knowledge bit instead.

Question Category	Example
Questions without context	Which example describes a learned behavior in a dog?
Questions with context	When athletes begin to exercise, their heart rates and respiration rates increase. At what level of organization does the human body coordinate these functions?
Negation Questions	A teacher builds a model of a hydrogen atom. A red golf ball is used for a proton, and a green golf ball is used for an electron. Which is not accurate concerning the model?

Table 6.1: Example questions for *Qtype* classification

6.4 Experiments

Dataset: We used a set of 8th grade science questions released as the training set in the Allen AI Science Challenge³ for training and evaluating our model. The dataset comprises of 2500 questions. Each question has 4 answer candidates, of which exactly one is correct. We used questions 1-1500 for training, questions 1500-2000 for development and questions 2000-2500 for testing. We also used publicly available 8th grade science textbooks available through *ck12.org*. The science curriculum consists of seven textbooks on Physics, Chemistry, Biology, Earth Science and Life Science. Each textbook on an average has 18 chapters, and each chapter in turn is divided into 12 sections on an average. Also, as described before, each section, on an average, is followed by 3-4 multiple choice review questions (total 1369 review questions). We collected a number of domain specific science dictionaries, study guides, flash cards and semi-structured tables (Simple English Wiktionary and Aristo Tablestore) available online and create triples and equivalences used as external knowledge.

Baselines: We compare our framework with ten baselines. The first two baselines (*Lucene* and *PMI*) are taken from [53]. The *Lucene* baseline scores each answer candidate a_i by searching for the combination of the question q and answer candidate a_i in a lucene-based search engine and returns the highest scoring answer candidate. The *PMI* baseline similarly scores each answer candidate a_i by computing the point-wise mutual information to measure the strength of the association between parts of the question-answer candidate combine and parts of the CK12 curriculum. The next three baselines, inspired from [227], retrieve the top two CK12 sections querying $q + a_i$ in *Lucene* and score the answer candidates using these documents. The *SW* and *SW+D* baselines match bag of words constructed from the question and the answer answer candidate to the retrieved document. The *RTE* baseline uses textual entailment [270] to score answer candidates as the likelihood of being entailed by the retrieved document. Then we also tried other approaches such as the *RNN* approach described in [53], *Jacana aligner* [300] and two neural network approaches, *LSTM* [121] and *QANTA* [128] They form our next four baselines. To test if our approach indeed benefits from jointly learning the retrieval and the answer selection modules, our final baseline *Lucene+LSSVM Alignment* retrieves the top section by querying $q + a_i$ in *Lucene* and then learns the remaining answer-entailment structure (i.e. the alignment part of the

³<https://www.kaggle.com/c/the-allen-ai-science-challenge/>

answer-entailing structure) using a LSSVM.

Task Classification for Multitask Learning: We explore two simple question classification schemes. The first classification scheme classifies questions based on the question word (what, why, etc.). We call this *Qword* classification. The second scheme is based on the type of the question asked and classifies questions into three coarser categories: (a) questions without context, (b) questions with context and (c) negation questions. This classification is based on the observation that many questions lay down some context and then ask a science concept based on this context. However, other questions are framed without any context and directly ask for the science concept itself. Then there is a smaller, yet, important subset of questions that involve negation that also needs to be handled separately. Table 6.1 gives examples of this classification. We call this classification *Qtype* classification⁴.

Results: We compare variants of our method⁵ where we consider our modification for negation or not and multi-task LSSVMs. We consider both kinds of task classification strategies and joint training (JT). Finally, we compare our methods against the baselines described above. We report accuracy (proportion of questions correctly answered) in our results. Figure 6.2 shows the results. First, we can immediately observe that all the LSSVM models have a better performance than all the baselines. We also found an improvement when we handle negation using the heuristic described above⁶. MTLSSVMs showed a boost over single task LSSVM. *Qtype* classification scheme was found to work better than *Qword* classification which simply classifies questions based on the question word. The multi-task learner could benefit even more if we can learn a better separation between the various strategies needed to answer science questions. We found that joint training with review questions helped improve accuracy as well.

Feature Ablation: As described before, our feature set comprises of five parts, where each part corresponds to a part of the answer-entailing structure – textbook (\mathbf{z}_1), chapter (\mathbf{z}_2), section (\mathbf{z}_3), snippets (\mathbf{z}_4), and alignment (\mathbf{z}_5). It is interesting to know the relative importance of these parts in our model. Hence, we perform feature ablation on our best performing model - *MTLSSVM(QWord, JT)* where we remove the five feature parts one by one and measure the loss in accuracy. Figure 6.3 shows that the choice of section and alignment are important components of our model. Yet, all components are important and removing any of them will result in a loss of accuracy. Finally, in order to understand the value of external knowledge resources (K), we removed the component that induces and aligns the hypothesis with knowledge bits. This results in significant loss in performance, establishing the efficacy of adding in external knowledge via our approach.

⁴We wrote a set of question matching rules (similar to the rules used to convert question answer pairs to hypotheses) to achieve this classification

⁵We tune the SVM regularization parameter C on the development set. We use Stanford CoreNLP, the HILDA parser [86], and jMWE [156] for linguistic preprocessing

⁶We found that the accuracy over test questions tagged by our heuristic as negation questions went up from 33.64 percent to 42.52 percent and the accuracy over test questions not tagged as negation did not decrease significantly

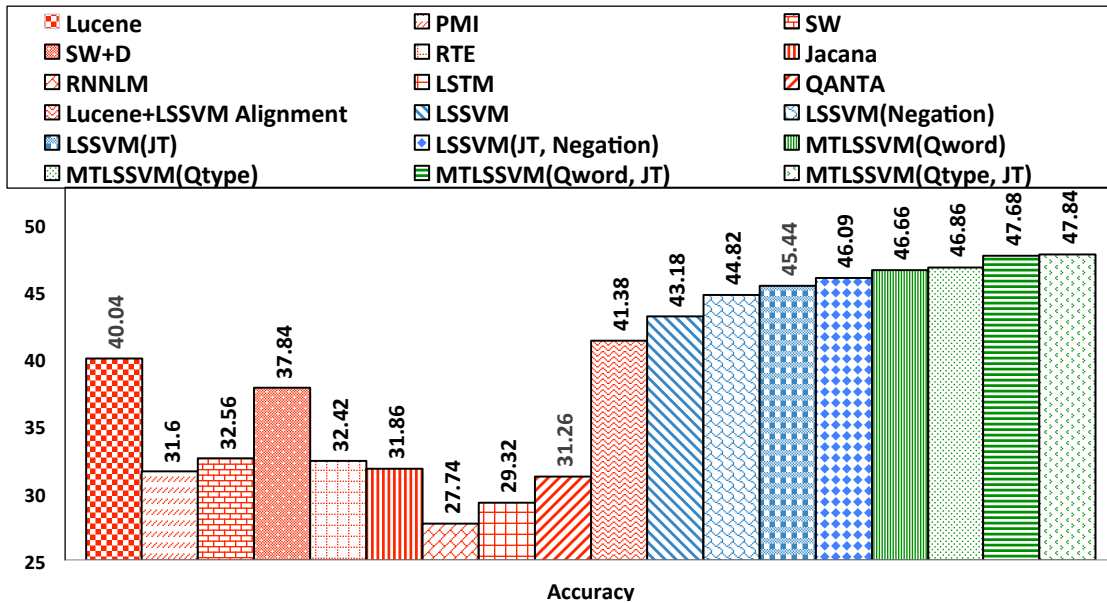


Figure 6.2: Variations of our method vs several baselines on the Science QA dataset. Differences between the baselines and LSSVMs, the improvement due to negation, the improvements due to multi-task learning and joint-learning are significant ($p < 0.05$) using the two-tailed paired T-test.

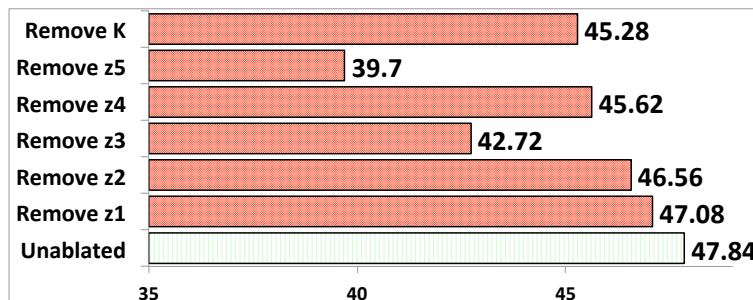


Figure 6.3: Ablation on *MTLSSVM(Qword, JT)* model

6.5 Conclusion

The science question answering task requires the ability to understand unstructured information in the form of textbooks in the student curriculum as well as external structured knowledge in the form of various knowledge tables, periodic tables, etc. Our results provide validation for our latent structure learning approach of jointly modelling retrieval and alignment and incorporating external knowledge in the *answer-entailing structure*. Furthermore, the multi-task learning approach which models different question types further improves performance of the model.

Chapter 7

Mathematical Question Answering: A Case for Explicit Reasoning

Human beings are rational – a view often attributed to Aristotle – and a major component of rationality is the ability to reason. Humans have a unique ability to reason based on their past interactions with other humans and their experiences with other devices inside their prescribed environment. Reasoning comes in many forms – including, but not restricted to logical reasoning, deductive reasoning, inductive reasoning, abductive reasoning; and other forms of reasoning considered more informal such as intuitive and verbal reasoning.

In the previous chapters, we looked at question answering tasks that test natural language comprehension such as reading comprehensions or science question answering. Even though reading comprehensions or science questions were targeted to students of particular age or grade level, the questions involved natural language in all its syntactic or semantic ambiguity. While these tasks clearly required a lot of reasoning, it is not clear how the various models to solve these natural language comprehension tasks model reasoning.

On the other hand, a number of standardised test problems in the domains of math and science require explicit forms of reasoning. This provides us with a number of rich problems that force us to think how we can model reasoning interacts with natural language understanding. Let us take the example of the math SAT exam – a popular mathematics examination given to pre-university students in the US. A large part of the math SAT exam comprises of problems in geometry. SAT geometry tests the student’s knowledge of Euclidean geometry in its classical sense, including an understanding of points, lines, planes, angles, triangles, congruence, similarity, solid figures, circles, and analytical geometry. A typical geometry problem is provided in Table 7.1 (left). The geometry question typically includes a textual description, and is often accompanied by a diagram. Various levels of understanding are required to solve geometry problems. An important challenge is understanding both the diagram (which consists of identifying visual elements in the diagram, their locations, their geometric properties, etc) and the text simultaneously, and then reasoning about the geometrical concepts using well-known axioms of Euclidean geometry.

Yet another similar task is of solving university level Newtonian Physics questions such as the one in Table 7.1 (right). These questions typically consist of a paragraph size piece of text describing the question and (optionally) an associated diagram. An AI system that can answer these questions must again be able to interpret both the question text as well as the associated

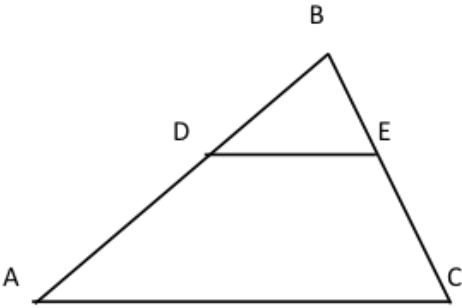
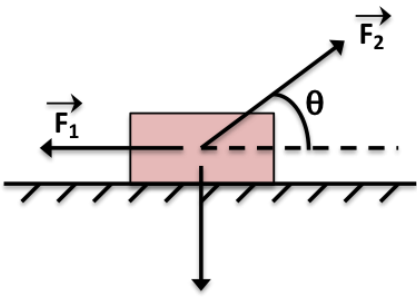
 <p>In triangle ABC, line DE is parallel with line AC, DB equals 4, AD is 8, and DE is 5. Find AC.</p>	 <p>Figure above shows three forces applied to a trunk that moves leftward by 3.00 m over a frictionless floor. The force magnitudes are $F_1 = 5.00\text{N}$, $F_2 = 9.00\text{N}$, and $F_3 = 3.00\text{N}$, and the indicated angle is $\theta = 60.0^\circ$. During the displacement, what is the net work done on the trunk by the three forces?</p>
---	---

Table 7.1: On the left: An example SAT geometry problem. Right: An example AP Newtonian physics problem.

diagram. These questions are quite diverse, offering a number of challenges. The questions text is usually ambiguous, posing a number of NLP challenges. The diagrams associated with these questions represent complex physical and mathematical concepts; not often represented in natural images. Hence, traditional computer vision technology cannot easily extract and represent the semantics of these diagrams. Finally, solving these questions requires the system to reason with domain knowledge of Physics (axioms, theorems, etc.).

These math and science problems explore a rich interplay of text and visual understanding, knowledge representation and extraction, and reasoning, posing a formidable challenge for AI systems. The task of answering these problems poses a number of key challenges for our AI systems:

1. How to interpret the question (i.e. interpret the question text together with the associated diagram)?
2. How to gather the domain knowledge of geometry and physics required to answer these questions?
3. How to use the question interpretation along with the domain knowledge to determine the answer?

In order to tackle these challenges, we propose a framework called parsing to programs (P2P). P2P assumes a formal language for the domain (geometry or physics) and some domain knowledge (representing rules, axioms and theorems in the domain) needed to solve these questions. The formal language and the domain knowledge can be manually provided by a domain expert, or, as we show in our work, can be extracted from textbooks in an automated way. We extract axiomatic knowledge from textbooks [242, 243] by (a) leveraging the *redundancy* and *shared ordering* of axiom mentions across multiple textbooks, and (b) utilizing rich *contextual* and *typographical* features¹ in textbooks to extract and parse axioms. When presented with a

¹A major contribution of this line of work is in linguistics. Linguistic theories of discourse only consider written

question, *P2P* learns a representation of the question in the formal language. This is challenging as:

1. *P2P* needs to interpret the question text in concert with the diagram.
2. Our datasets are small – these questions are curated by human experts for testing purposes; moreover, annotating formal representations of questions is expensive.
3. These problems are diverse, representing complex concepts which are not often present in natural text and images. For instance, in Figure 7.1b, the visual concept of `FLOOR` is represented as a solid horizontal line with many small, same sized, parallel lines with their end points on the horizontal line. This concept can be concisely expressed as a rule but is hard to learn. Hence, traditional NLP and computer vision methods cannot directly be ported to extract and represent the semantics of these problems as a rich integration of domain knowledge is needed.

To tackle these issues, we proposed `Nuts&Bolts` [244], an approach to learn a pipeline process that incorporates existing code, pre-learned machine learning models and human engineered rules. It jointly trains the interpretation pipeline to prevent propagation of errors, using labeled as well as unlabeled data.

In the final step, *P2P* uses the formal question interpretation to solve the question using relevant domain knowledge. *P2P* can be seen as a **v2.0 expert system**. Expert systems [130] were popular in early stages of AI but went out of favor as (a) they could not support linguistic and visual input and (b) they needed extensive knowledge engineering which was expensive. Our research demonstrates that we can use machine learning to mitigate these two key issues in expert systems and successfully deploy them. We can indeed build a system that can read textbooks and answer challenging reasoning problems in the domains of geometry [245] and mechanics [238].

7.1 Related Work

The *P2P* framework bring together multiple ideas from semantic parsing of text, diagram understanding, multimodal learning, knowledge representation, information extraction and reasoning in order to solve situated questions. We review related work thematically under the aforementioned heading. As described earlier, *P2P* can be viewed as a natural language interface to expert systems. We begin by reviewing some early work (for example, by building expert systems) for solving geometry and physics problems.

Historical work on solving geometry and physics Problems: The problem of using computers to solve geometry questions is old [64, 82, 251]. However, these approaches have mostly been rule-based or have employed very limited amount of machine learning. Expert system type approaches that use logical inference for geometry theorem proving such as the Wus method [292], Grobner basis method [137], and angle method [49] have been proposed for tutoring systems such as *Geometry Expert* [95] and *Geometry Explorer* [294]. There has also been research in

text without much formatting. However, in this multimedia age, text is often richly formatted. Especially, textbooks contain rich formatting features with the intent of making the subject material easy to grasp and remember for students. We provided the first corpus analysis of multimedia text and used it to show that the formatting features can be used for information extraction.

synthesizing geometry constructions given logical constraints [106, 127] or generating geometric proof problems [5] for applications in tutoring systems. However, these approaches were pretty brittle and can only handle limited types of questions. Similarly for the physics domain, prior work has focused on expert systems that can solve a very narrow domain of problems. For example, [40] solves simple vector addition, tension, and gravitation ranking problems and [148] solves physical reasoning problems by analyzing sketches. [144, 145, 146, 147] further use the expert systems to explore the idea of domain transfer via analogies for solving physics problems, however, with limited success. On the other hand, P2P uses machine learning for majority of the pipeline. Thus, it is more robust. Moreover, the domain knowledge necessary for these questions is automatically harvested from textbooks in P2P. In all the mentioned expert systems, the domain knowledge was provided by domain experts which is very time-consuming.

Semantic Parsing: Semantic parsing is an important area of NLP research [19, 73, 91, 96, 97, 138, 159, 220, 226, 313]. However, semantic parsers do not tackle diagrams—a critical element of the situated question answering. In addition, we assume that the overall number of available situated questions is quite small compared to the size of typical NLP corpora. This makes it challenging to learn semantic parsers directly from situated questions. Relation extraction is another area of NLP that is related to our task [58, 60]. Again, these works do not handle diagrams and small corpora size as in our setting brings in unique challenges for this body of work.

Our work is part of grounded language acquisition research [6, 10, 22, 24, 46, 108, 120, 143, 151, 169, 281] that involves mapping text to a restricted formalism (instead of a full, domain independent representation). In our work, we recover the entities (e.g., circles) from diagrams, derive relations compatible with both text and diagram, and re-score relations derived from text parsing using diagram information. Our approach of casting the interpretation problem as selecting the most likely subset of literals can be generalized to grounded semantic parsing domains such as navigational instructions.

Multimodal learning for diagram and text understanding: Coupling images and the corresponding text has attracted attention in both vision and NLP [79, 80, 99, 107, 155]. We build on this powerful paradigm, but instead of generating captions we show how processing multimodal information help improve textual or visual interpretations for solving situated questions. Most previous approaches differ from our method because they address the twin problems of diagram understanding and text understanding in isolation. Often, previous work relies on manual identification of visual primitives, or on rule-based system for text analysis.

Diagram understanding has been explored since early days in AI (e.g., [87, 88, 113, 176, 205, 267]). We refer interested readers to [208]. Most previous work differ from our method because they address two problems of diagram understanding and text understanding in isolation. Our work is related to early work on coupling over textual and visual data [28, 206, 267], however these methods assume that the visual primitives of diagrams are manually identified. P2P revisits the problem of diagram understanding by coupling two tasks of visual understanding of diagrams and detecting alignments between text and diagrams.

The most common approach to diagram understanding is a bottom up method where primitives can be linked together [173] to form larger elements such as rectangles [165] or general shapes [200]. Using Hough transform is another popular alternative in detecting visual elements [136, 315]. What is common among almost all conventional methods of visual element

identification is thresholding of a scoring function that determines the existence of visual elements. Although being considered as a well studied subject, our experiments reveal that the thresholding step hinders applications of such techniques on real-world situated questions. Our data suggests that there is no single threshold that results in a reliable discovery of visual elements across different diagrams. In this paper, we propose a method that initially overestimates the visual elements, but then benefits from submodular optimization coupled with textual information to home in on the correct elements.

Coupling visual and textual information has recently attracted attention in both vision and NLP [80, 107, 155]. We build on this powerful paradigm, but utilize it for the more manageable task of understanding diagrams in situated questions. Understanding these diagrams is more manageable because diagrams are less ambiguous, expose less visual variance, have smaller vocabulary of elements than images typically studied in machine vision. This easier task allows us to have more reliable estimates of visual elements and focus on interactions between textual mentions and visual elements.

Knowledge representation and Reasoning: Knowledge representation (KR) is a large sub-field of AI dedicated to representing knowledge about the world in a form that a computer system can utilize to solve complex tasks. Some common knowledge representation paradigms that have been used in literature include rules, relations of sets and subsets, ontologies, knowledge graphs and logic. A thorough review of KR is beyond the scope of this paper so we point the interested reader to [23, 63] – two comprehensive surveys on knowledge representation. In our work, we represent knowledge as logic rules which is common in a lot of AI approaches [94, 171]. The language of logic is well-suited to capture reasoning, due to its expressivity, its model-theoretic semantics, and its inferential power.

One of the main purposes of explicitly representing knowledge is to be able to reason about that knowledge, to make inferences and assert new knowledge. Thus, knowledge representation goes hand in hand with automated reasoning methods. Indeed, most knowledge representation languages have a reasoning or inference engine as part of the system. There has been a large body of work on reasoning and a thorough review of reasoning approaches is beyond the scope of this paper. There are two main methods of reasoning inference engines in literature: forward chaining (or forward reasoning) and backward chaining (or backward reasoning) [81]. In theory, we can use both forward as well as backward chaining in our implementation. However, in our work, we use forward chaining, which is a popular inference strategy in expert systems. Forward chaining uses modus ponens (if A then B, A, therefore B) to connect data to conclusions, starting with the data and searching for provable conclusions until we arrive at the solution.

Information Extraction from Textbooks: Our model for extracting structured domain rules from textbooks builds upon ideas from Information extraction (IE), which is the task of automatically extracting structured information from unstructured and/or semi-structured documents. While there has been a lot of work in IE on domains such as web documents [15, 32, 38, 39, 76, 77, 197] and scientific publication data [215, 246, 255], work on IE from educational material is much more sparse. Most of the research in IE from educational material deals with extracting simple educational concepts [33, 168, 181, 255, 286, 287, 295, 299] or binary relational tuples [13, 52, 61] using existing IE techniques. On the other hand, our approach extracts axioms and parses them to structured rules. This is much more challenging. Raw application of rule mining or sequence labeling techniques used to extract information from web documents and scientific

publications to educational material usually leads to poor results as the amount of redundancy in educational material is lower and the amount of labeled data is sparse. Our approach tackles these issues by making judicious use of typographical information, the redundancy of information and ordering constraints to improve the harvesting and parsing of axioms. This has not been attempted in previous work.

Language to Programs: After harvesting axioms from textbooks, we also parse the axiom mentions to structured rules. This work is again related to semantic parsing [139, 311, 312, 313, inter alia]. Semantic parsers typically map natural language to formal programs such as database queries [20, 170, 297, inter alia], commands to robots [47, 190, 261, inter alia], or even general purpose programs [161, 178, 179, 304]. More close to our work, Liu et al. [180] and Quirk et al. [223] learn “If-Then” and “If-This-Then-That” rules, respectively. In theory, these works can be adapted to parse axiom mentions to horn-clause rules. However, this would require a large amount of supervision which would be expensive to obtain. We mitigated this issue by using redundant axiom mention extractions from multiple textbooks and then combining the parses obtained from various textbooks to achieve a better final parse for each axiom.

7.2 Methodology

We build a semi-automated solver that solves the two kinds of problems. First, we define a typed logical language: a subset of typed first-order logic comprising of *constants*, *variables*, and a hand-picked set of predicates for the specific domain. Then, we also define the structured domain knowledge representing the axioms and theorems in the domain required to solve these questions. These could be horn clause rules or more complex general purpose programs. The domain knowledge can be manually provided by a domain expert or may be extracted from the instructional material in an automated way.

Given this domain knowledge, our approach answers the questions in two stages: (1) *Question Parsing* – which parses the question text and any associated diagram and represents it as a (weighted) logical expression, and (2) *Programmatic Solving* – which applies the provided domain knowledge to answer the question given the logical expression representing the question. The *Question Parsing* stage maps the question (question text as well as any associated diagram) to a formal representation. This is achieved by generating weighted first-order logic formulas (a set of literals) that correspond to the question text and associating a confidence score with each literal. The *Programmatic Solving* stage takes this formal representation of the question and solves it by performing (probabilistic) reasoning using the provided domain knowledge. A flowchart of the procedure is shown in Figure 7.1. This can be perceived as a natural language interface to expert systems [111] which were popular in early stages of AI for solving these kinds of problems.

Problem Formulation: A situated question is a tuple (t, d, c) consisting of a text t in natural language, optionally a diagram d in raster graphics, and optionally choice answers $c = \{c_1, \dots, c_M\}$. The task of answering the situated question is to find the correct answer for the question. For ease of evaluation, we assume that all the questions are either multiple choice questions, or subjective questions where the answer is a numeric quantity (e.g. 3 cm, 50° , 5 m/s^2 , etc.) such that the predicted answer can easily be compared to the correct answer.

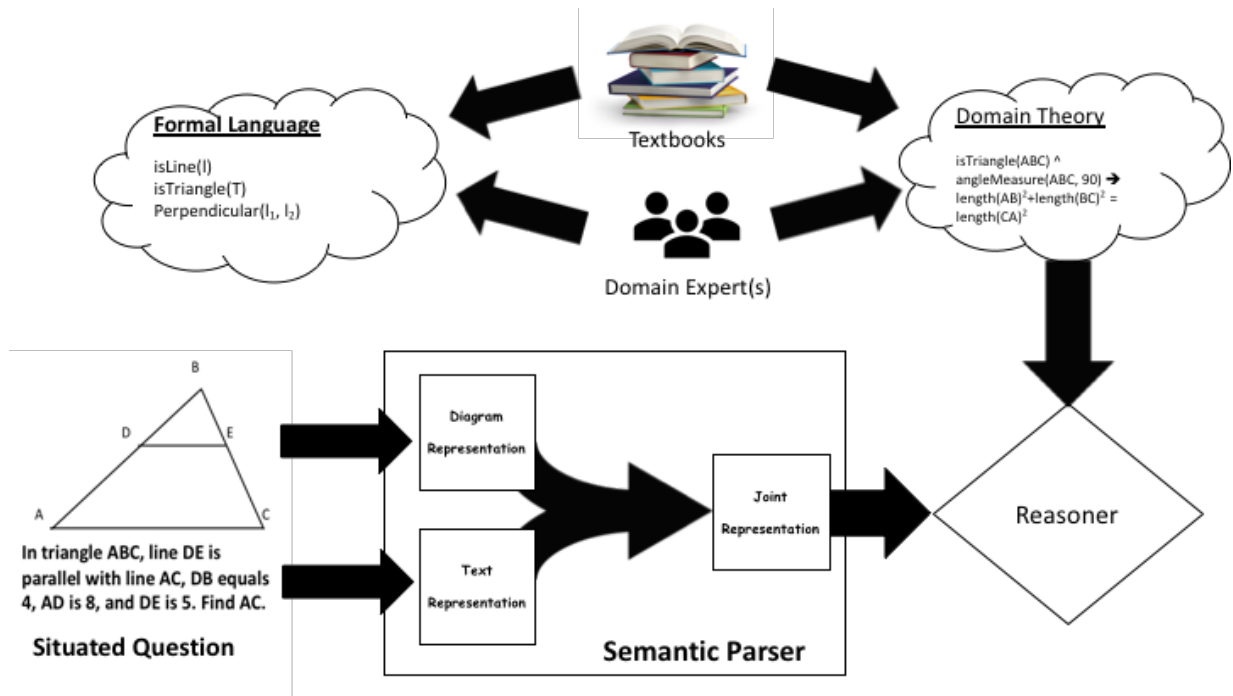


Figure 7.1: The framework of our Parsing to Programs approach. The approach solves the question in two stages. The first stage, *Question Parsing*, parses the question text and any associated diagram into an equivalent (weighted) logical expression in a typed first-order logic language. The second stage, *Programmatic Solving* takes this formal representation of the question and solves it using the domain specific theory provided to the system.

Formal Language: We define our formal language Ω as a subset of typed first-order logic and represent questions as logical expressions in the language. The language includes:

- *constants*, corresponding to known numbers (e.g., 4, 8 and 5 in Table 7.1a and 3.00 m, 5.00 N, 60° in Table 7.1b) or known geometrical or physical entities.
- *variables*, corresponding to unknown numbers or entities in the question (e.g., A, DE, ABC, etc. in Table 7.1a and F_1 , θ , etc. in Table 7.1b).
- *predicates*, corresponding to relations (e.g., Equals, IsDiameter, IsTangent, etc. in in geometry domain and Equals, isAtRest, etc. in the physics domain).
- *functions*, corresponding to properties of entities (e.g., LengthOf, AreaOf in the geometry domain and *mass, distance, force, speed, velocity, work*, etc. in the physics domain) or arithmetic operations (e.g., SumOf, RatioOf, etc.).

For both applications, element in the language have either boolean (e.g., true), numeric (e.g., 4), or entity (e.g., line, circle, object, force, mass, velocity) type. We refer to all symbols in the language Ω as *concepts*. We use the term *literal* to refer to the application of a predicate to a sequence of arguments (e.g., IsTriangle(ABC)). Literals are possibly negated atomic formulas in the language Ω . Logical formulas contain constants, variables, functions, existential quantifiers and conjunctions over literals (e.g., $\exists x, \text{IsTriangle}(x) \wedge \text{IsIsosceles}(x)$).

Given this formal language, our *Parsing to Programs* method consists of three steps: (1) *interpreting* a question by deriving a logical expression that represents the meaning of the text and the diagram, (2) acquiring *domain knowledge*, and (3) *solving* the question by using the parsed interpretations and the domain knowledge.

Interpretation is the task of mapping a new question (with each answer choice, if available), (t, d, c_m) , into a logical formula γ in Ω . Hereafter, we will omit the argument c_m for ease of notation. More formally, the goal is to find $\gamma^* = \arg \max_{\gamma \in \Gamma} \text{score}(\gamma; t, d)$ where Γ is the set of all logical formulas in Ω and *score* measures the interpretation score of the formula according to both text and diagram. We show that the problem of deriving the best formula γ^* can be modeled as a combinatorial search in the space of literals L (note that each logical formula γ is represented as a conjunction over literals l_i). When reasoning under uncertainty (for example in a larger framework such as our P2P framework), we instead represent questions as weighted logic formulas where each literal is also associated with a weight corresponding to the confidence of our method that the literal is correct.

Answering situated questions requires a method that interprets question text and diagrams in concert. These questions have several distinctive characteristics.

1. First, diagrams provide essential information absent from question text. In Table 7.1 problem (a), the unstated facts that point D lies on line AB and point E lies on line BC are necessary to solve the problem. Similarly, in Table 7.1 problem (b), we need to read off the direction of the three forces and that θ is the angle between the horizontal and the direction of force \vec{F}_2 .
2. Second, the text often includes references to diagram elements. For example, in the sentence “In the diagram, the longer line is tangent to the circle”, resolving the referent of the phrase “longer line” is challenging. Similarly, in Table 7.1 problem (b), resolving that the phrase “three forces” refers to forces \vec{F}_1 , \vec{F}_2 and \vec{F}_3 is challenging.

3. Third, the text often contains implicit relations. For example, in the sentence “AB is 5”, the relations $\text{IsLine}(AB)$ and $\text{length}(AB)=5$ are implicit. In the sentence “ $F_1 = 5.00\text{ N}$ ”, the relations $\text{IsForce}(F_1)$ and $\text{magnitude}(F_1)=5.00\text{ N}$ are implicit (because of the type/dimensionality constraint imposed by N (Newtons), F_1 can only resolve to force).
4. Fourth, some terms can be ambiguous as well. For instance, radius can be a type identifier in “the length of radius AO is 5”, or a predicate in “AO is the radius of circle O”.
5. Fifth, identifying the correct arguments for each relation is challenging. For example, in sentence “Lines AB and CD are perpendicular to EF”, the parser has to determine *what* is perpendicular to EF—line AB? line CD? Or both AB and CD?
6. Finally, it is hard to obtain large situated question answering datasets; Learning from a few examples makes this a particularly challenging NLP problem.

We describe the P2P framework for answering geometry questions below:

7.3 Answering Geometry Questions

We first apply the *Parsing to Programs* approach described above to solve SAT style geometry problems. First, we describe our approach on diagram and question text parsing. We write axioms of geometry in the form of horn-clause rules and show that we can use probabilistic logic to solve these problems (see section 7.3.3). We also show that we can harvest this subject knowledge from textbooks in the form of structured programs from textbooks using the typographical and lexical information (see section 7.3.4). Finally, we show that we can learn to solve the SAT geometry problems using demonstrative solutions to these problems. Such demonstrations are common in textbooks as they help students learn how to solve geometry problems effectively. See section 7.4.1 for details.

7.3.1 Diagram and Question Parsing

P2P parses geometry problems via a multi-stage approach. It first learns to parse the problem text and the diagram to a formal problem description compatible with both of them. The problem description is a first-order logic expression that includes known numbers or geometrical entities (e.g. 4 cm) as constants, unknown numbers or geometrical entities (e.g. O) as variables, geometric or arithmetic relations (e.g. *isLine*, *isTriangle*) as predicates and properties of geometrical entities (e.g. *measure*, *liesOn*) as functions. The parser first learns a set of relations that potentially correspond to the problem text (or diagram) along with confidence scores. Then, a subset of relations that maximize the joint text and diagram score are picked as the problem description.

For diagram parsing, *P2P* obtains the set of all visual elements, their coordinates, their relationships in the diagram, and their alignment with entity references in the question text. The parser also provides confidence scores for each literal to be true in the diagram.

Text parsing is performed in three stages. The parser first maps words or phrases in the text to their corresponding concepts. Then, it identifies relations between identified concepts. Finally, it performs *relation completion* which handles implications and coordinating conjunctions.

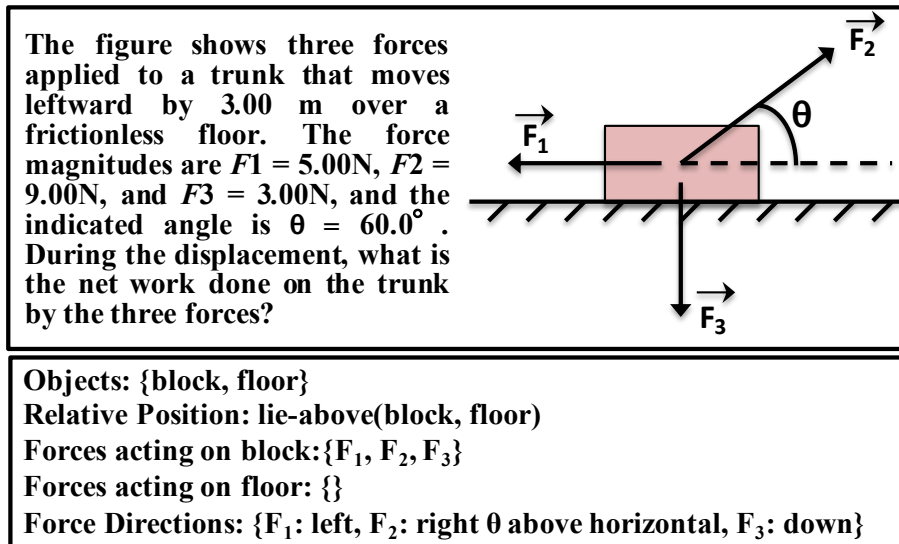


Figure 7.2: Above: An example Newtonian physics problem. Below: Diagram parsed in formal language.

A key challenge for us in building the diagram and text parsers is supervision. Most modern machine learning approaches are very data hungry. However, we cannot have large datasets for these kinds of domains. One possible remedy to the issue of data-hungriness is to incorporate domain knowledge.

Thus, we rely on the classical approach of building pipelines. Pipelines decompose a complex problem into a series of stages, where the local predictor at a particular stage depends on predictions from previous stages. Pipelining is popular as it breaks down the task into easier-to-handle sub-tasks which can be individually implemented. In contrast, pipelines [283] decompose a complex task into a series of easier-to-handle sub-tasks (stages), where the local predictor at a particular stage depends on predictions from previous stages. Pipelines can be tuned with small amount of labeled data and it is easier to incorporate domain knowledge expressed as rules, existing software and pre-learnt components. However, pipelining suffers from propagation of local errors [90].

Thus, we propose `Nuts&Bolts`: an approach for learning pipelines with labeled data, unlabeled data, existing software and domain knowledge expressed as rules. By jointly learning the pipeline, `Nuts&Bolts` retains the advantages of end-to-end learning (i.e. doesn't suffer from error propagation). Furthermore, it allows for easy incorporation of domain knowledge and reduces the amount of supervision required, removing the two key shortcomings of end-to-end learning.

We use this approach to parse our Geometry and Newtonian physics problems into the formal language (see Figure 7.2 for an example). This is useful as it builds a computer ingestible rich semantic representation of these problems. These problems are quite diverse, representing complex physical and mathematical concepts which are not often present in natural text and images²

²For instance, in Figure 7.2, the visual concept of `FLOOR` is represented as a solid horizontal line with many small, same sized, parallel lines with their end points on the horizontal line (see Figure 7.4c). This concept can be concisely expressed as a rule but is hard to learn.

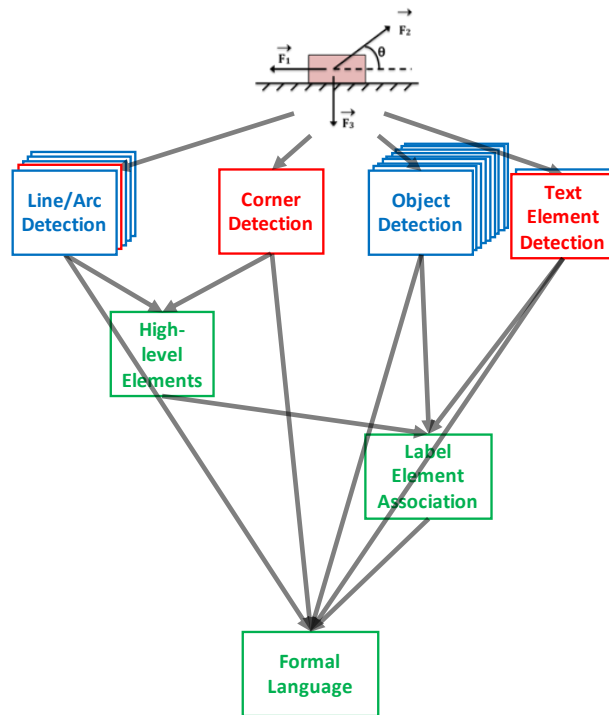


Figure 7.3: A pipeline for diagram parsing with various (possibly multiple) pre-trained functions, existing software and rules. The pre-learnt components are shown in blue, the existing software are shown in red and rule-based components are shown in green.

– understanding them itself requires substantial domain knowledge. Hence, traditional NLP and Computer Vision methods cannot be directly ported to extract and represent the semantics of these problems, and a richer integration of domain knowledge is needed.

The pipeline allows us to think of the task in a modular way, and to integrate stagewise supervision and domain knowledge of physics into the model. It also allows us to supervise the various sub-components to aid rapid learning. We begin by describing the parsing pipeline at a high level. We break the parsing task into three phases. In the first phase, we parse the diagram recognizing the various diagram elements and relationships between them, leading to a diagram parse in formal language. In the second phase, we parse the problem text into the same formal language. In the third and final phase, we reconcile the diagram and the text parse and achieve the final parse of the problem

Diagram parsing is performed in the following stages: (a) We first identify low-level diagram elements such as the lines and arcs, corners i.e intersecting points of various low-level diagram elements, objects (e.g. block in Figure 7.2) and text elements, i.e. labels such as $\vec{F}_1, \vec{F}_2, \vec{F}_3$ and θ in Figure 7.2. (b) Then, we assemble the various low-level diagram elements (such as lines and arcs) to higher level diagram elements (such as axes, blocks, wedges, pulleys, etc.) by a set of human engineered grouping rules. (c) Then, we map the various text elements to corresponding diagram elements detected in the previous stages. For example, the text element \vec{F}_1 in Figure 7.2 refers to the leftward arrow in the diagram. (d) In the final step, we use a set of human-engineered rules to maps the diagram to formal language. We show

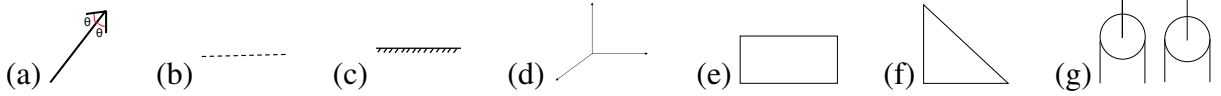


Figure 7.4: Some example high-level diagram elements: (a) Arrow, (b) Dotted line, (c) Ground, (d) Coordinate System, (e) Block, (f) Wedge, and (g) Pulley. We describe rules to form these elements in Table 7.2.

the various stages in the diagram parsing pipeline and their corresponding inputs and outputs in Figure 7.3.

Next, we describe the various components of the diagram parsing pipeline, pointing out the pre-learnt functions, software and rules in each stage of the pipeline (see Table 7.2). Note that every pre-learnt function can be treated as a software. We present this difference merely due to philosophical reasons.

In the first stage, we detect low-level diagram elements (lines and arcs) using a number of pre-learnt functions and software. For corner detection, we use Harris corner detectors [110]. Then we further assemble these low-level diagram elements to high level elements. High level elements can be easily expressed by humans as rules given their knowledge of Physics (see Figure 7.4). However, it is difficult to learn the input-output mapping for high-level elements directly as this will require a very large amount of labelled data for each high-level element. We introduce a set of manually curated grouping rules for grouping low-level diagram elements to form high-level diagram elements. For example, the rule to form an arrow tests if there are three detected lines which share an end-point which can be combined to form an arrow. The three lines must also satisfy some additional conditions for the high-level element to be an arrow. The central line (stem line) is the longest of the three lines, the two arrowhead lines are roughly of the same length and the two angles subtended by the arrowhead lines with the arrow stem line must be roughly equal.

This rule is incorporated as shown below:

$$\begin{aligned}
 C_1 &= isLine(line1) \wedge isLine(line2) \wedge isLine(line3) \\
 C_2 &= length(line1) > length(line2) \wedge length(line1) > length(line3) \quad i.e. \text{ line1 is stem} \\
 C_3 &= roughly_equal(angle(line1, line2), angle(line1, line3)) \\
 C_1 \wedge C_2 \wedge C_3 &\rightarrow H_{arrow}
 \end{aligned}$$

All our rules take the form of {IF:THEN} expressions, e.g. {IF condition THEN result}, where the condition tests if a set of detected low-level elements satisfy the requirements to form the high-level element. In general, we write down a rule for high-level element detection as $r_i : AND(l_{i_1}, l_{i_2}, \dots, l_{i_\alpha}, c_{i_1}, c_{i_2}, \dots, c_{i_\beta}) \rightarrow h_i$ s.t. the rule preconditions $P_{i_1}, P_{i_2}, \dots, P_{i_\gamma}$ are all satisfied. Here, $l_{i_1}, l_{i_2}, \dots, l_{i_\alpha}$ denote pre-requisite low-level elements and $c_{i_1}, c_{i_2}, \dots, c_{i_\beta}$ denote pre-requisite corner elements required for the application of rule r_i leading to the formation of high-level element h_i . Then, we map textual element labels with diagram elements. Let M_{te} represent a variable that takes values 1 if the detected element (high-level element or object) e is matched with the detected text label t , and 0 otherwise. Here, we have a matching constraint that $\sum_e M_{te} = 1$ which states that every text label must be matched to exactly one high-level element

or object. Next, we build a set of candidate matching rules. These rules essentially capture features accounting for type, shape, orientation, distance, etc. In the final stage, we again use a set of rules to map diagrams to formal language. These rules, one for each predicate, decide if the predicate holds for a set of diagram elements which are type consistent with the arguments of the predicate. We have rules for listing objects, relative position of objects, forces acting on objects, force directions, etc.

The text parsing pipeline works in two stages: (a) identifying concepts in the formal language and then (b) relations between the detected concepts. We will describe this later in more detail.

Formulation: For each stage of the pipeline, we have choices for various pre-existing software (such as various line or corner detectors), rules or pre-learned functions that we additionally wish to integrate. We also wish to minimize propagation of errors by learning the pipeline jointly. Next, we formalize the problem of learning such a pipeline.

Let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ represent members of the input and output domains of a data mining task, respectively where we wish to learn a prediction function $f : \mathcal{X} \rightarrow \mathcal{Y}$ such that $\hat{y} \approx f(x)$.

Pipeline. We formally define a pipeline \mathcal{P} as a directed acyclic graph (DAG) $G = (V, E)$ where nodes V represent various computation modules in the pipeline and edges E represent input/output relationships between the various modules. Given G , we can always derive a topological ordering of the computation modules, thus decomposing the prediction problem into S stages. At each stage s , a predictor $f^{(s)}$ takes in as input the data instance x and predictions from all previous stages $f^{(s)} : z^{(s)} \rightarrow y^{(s)}$ where $z^{(s)} = (x, \hat{y}^{(0)}, \dots, \hat{y}^{(s-1)})$. Given a model for each stage of the pipeline, predictions are made locally and sequentially with the expressed goal of maximizing performance on all the various stages, $\hat{y} = f(x) = \{f^{(s)}(z^{(s)})\}_{s=1}^S$.

Extended Pipeline. Usually, a pipeline has a single predictor at each stage. However, system engineers are often faced with many choices for every stage of the pipeline. For example, they might have to choose between many different object detectors or many different part-of-speech taggers. It will be useful to not have to make that choice but have an ensemble of these choices. Hence, we extend our definition of the pipeline and assume that we are given multiple function approximators $\{f_i^{(s)}\}_{i=1}^{K_s}$ for the pipeline stage s and we wish to use them to estimate the true underlying function $f^{(s)}$. $f_i^{(s)}$ could use a pre-existing software, encode a domain-specific rule or a pre-learned function.

Problem Definition: Given the pipeline \mathcal{P} , multiple function approximators for each stage $\{\{f_i^{(s)}\}_{i=1}^{K_s}\}_{s=1}^S$ and partial supervision $\mathcal{S} = \{(x_n, \{y_n^{(t)}\}_{t \in \Omega_n})\}_{n=1}^N$, we want to learn the global prediction function $f(x)$. Here, N denotes the total number of data instances and Ω_n is a set of stages for which supervision is available for the n^{th} data instance x_n . In general, at each stage, the predictor $f^{(s)}$ may output a binary prediction $y^{(s)} \in \{0, 1\}$ or a regression $y^{(s)} \in (0, 1)$. We desire a framework which can handle partial supervision, existing software and domain knowledge expressed as rules in a feasible manner. To this end, we describe `Nuts&Bolts`, an approach that integrates these inputs while minimizing a global objective defined over all stages.

Nuts&Bolts – Learning with Constraints: We use CCMs [42] to model the `Nuts&Bolts` framework. Our original goal is to minimize the overall loss function \mathbf{L} defined on the final function approximation \hat{f} . We assume that the pipeline factorizes as a Bayesian network and the overall loss function factorizes as the following:

Table 7.2: Various components of the diagram parsing pipeline. We denote the pre-learnt functions by ●, software by ● and rules by ● in each stage of the pipeline.

Line	<p>Apply a weak Gaussian blur on the raw image and then binarized it using a threshold selection method proposed in [209]. Then using it, apply:</p> <ul style="list-style-type: none"> ● Boundary detection and grouping method [150] ● Hough transforms [71] ● Detect parallel curved edge segments in a canny edge map. ● Recursively merge proposals that exhibit a low residual when fit to a 1st or a 2nd degree polynomial. ● A 2 class CNN resembling VGG-16 [263] with a fourth channel (which specifies the location of the diagram element smoothed with a Gaussian kernel of width 5) appended to the standard 3 channel RGB input.
Corner	<ul style="list-style-type: none"> ● Harris corner detector [110]
High Level	<p>{IF:THEN} expressions, i.e. IF 'condition' THEN 'result' rules as described below:</p> <ul style="list-style-type: none"> ● Arrow: The central line (stem line) is the longest of the three lines, the two arrowhead lines are roughly of the same length, and the two angles subtended by the arrowhead lines with the arrow stem line must be roughly equal ● Dotted line: The various lines should be in a straight line, roughly the same sized lines and equi-spaced ● Ground: The solid line is in contact with a number of smaller parallel lines which subtend roughly the same angle with it, their end-point lies on the solid line and the smaller lines are on the same side with respect to the solid line ● Coordinate System: Three arrows where the arrow tails are incident on the same point. Two lines are mutually perpendicular (i.e. angle=90°) and the third roughly bisects the complementary (270°) angle ● Block: Four lines which form a rectangle ● Wedge: Three lines where each two share a distinct end-point ● Pulley: A circle with two lines tangent to it. An end-point of the two lines lies on the circle
Text	<ul style="list-style-type: none"> ● An off-the-shelf OCR system – Tesseract³. ● Since many textual elements are heavily structured (these include elements in vector notation (e.g. \vec{F}), greek alphabets (e.g. θ), physical quantities (e.g. 2 m/s)) and are usually longer than a single character, we trained a text localizer using a CNN having the same architecture as AlexNet [153]. We used the Chars74K dataset [65], a dataset obtained from vector PDFs of a number of physics textbooks and a set of synthetic renderings of structured textual elements generated by us as training data.
Object	<ul style="list-style-type: none"> ● Window classification [280] ● Perceptual grouping [35, 105] ● Cascaded ranking svm [316] ● Objectness [3] ● Selective search [277] ● Global and local search [225] ● Edge boxes [319] ● A classifier with features capturing location, size, central and Hu moments, etc. ● A discriminatively trained part-based model [85] trained to focus on the detection of a manually selected list of objects commonly seen in physics diagrams (blocks, pulleys, etc.).
Label Association	<ul style="list-style-type: none"> ● Type Matching Rules: Type matching rules note that if the element is of type t_1 and the text label is of type t_2, then the element should be matched to the text label. Thus, the rule can be written down as $type(e, t_1) \wedge type(t, t_2) \rightarrow M_{t_1}$. We have type matching experts for the following element-object types: (a) element is an arrow and the text label is one of F, v, a, g, x, d indicating physical vector quantities such as forces, velocity, acceleration and displacement, (b) element is the coordinate system and the text label is one of x, y or z indicating one of coordinate system axes, (c) element is a block or a wedge and the text labels it as a block' or 'wedge' (or one of their synonyms) respectively. ● Proximity Rules: The proximity rule notes that if the element and the text label are close to each other (i.e. the closest pixels of the element and the text label are closer than a threshold) then the element should be matched to the text label i.e. $proximal(t, o) \rightarrow M_{t_1}$ ● Orientation Rule: The orientation rule notes that if the element and the text label are in the same orientation, they should be matched i.e. $orientation_match(t, o) \rightarrow M_{t_1}$. The orientations are computed using the first principal component of the grey scale pixels labeled as the element/text.
Formal Language	<ul style="list-style-type: none"> ● Rules (one for each predicate) decide if the predicate holds for a set of diagram elements which are type consistent with the arguments of the predicate.

$$\min_f \mathbf{L}(\hat{f}, y) = \min_{\{f_v\}} \sum_v L(\hat{f}_v, y_v)$$

We further regularize the above objective to incorporate multiple function approximators and any given prior domain knowledge. We describe this below:

A. Integrating multiple function approximators: We introduce a notion of ‘trustworthiness’ model to integrate multiple function approximators. Let $T_i^{(s)} \in (0, 1)$ denote how much we can trust the function approximator $f_i^{(s)}$. In probabilistic logic, we introduce the following rules which specify the relationship between various function approximators, the unknown true underlying function and our trusts on the various function approximators. Thus, we have:

$$f_i^{(s)}(z^{(s)}) \wedge T_i^{(s)} \rightarrow f^{(s)}(z^{(s)}), \neg f_i^{(s)}(z^{(s)}) \wedge T_i^{(s)} \rightarrow \neg f^{(s)}(z^{(s)}) \quad (7.1)$$

$$f_i^{(s)}(z^{(s)}) \wedge \neg T_i^{(s)} \rightarrow \neg f^{(s)}(z^{(s)}), \neg f_i^{(s)}(z^{(s)}) \wedge \neg T_i^{(s)} \rightarrow f^{(s)}(z^{(s)}) \quad (7.2)$$

Intuitively, the first set of rules state that if a function approximator is trustworthy, its output should match the output of the true function. The second set of rules state that if a function approximator is not trustworthy, its output should not match the output of the true function. The trust values are implicitly learnt based on the agreement between various function approximators [218].

We make an additional assumption that most of the function approximators are better than chance. With this assumption, we additionally add the following two rules: $f_i^{(s)}(z^{(s)}) \rightarrow f^{(s)}(z^{(s)})$, $\neg f_i^{(s)}(z^{(s)}) \rightarrow \neg f^{(s)}(z^{(s)})$. This helps alleviate the identifiability issues introduced by the above rules (eq. 7.1 and 7.2). Note that flipping the values of trusts (i.e. setting them to one minus the trust values) and the true functions leads to the rules evaluating to the same set of rules as before. In probabilistic frameworks where all the rules are weighted with a real value in $[0, 1]$, we can think of the weight of these prior belief rules as regularization weights which can be learnt from data. Note that we estimate a single trust variable for every function approximator in the pipeline stage – trust is shared across data instances. Thus, the trust variables implicitly couple various function approximators by relating them to the true underlying function, aiding semi-supervised learning.

B. Integrating domain knowledge: Pre-existing software or pre-learned functions can be incorporated as function approximators in our probabilistic logic framework. Next, we will describe how we incorporate domain knowledge in the form of rules. We assume that the rules are provided to us as conditional statements (or implications) which can be read as “if *Precondition* then *Postcondition*”. Note that *Precondition* and *Postcondition* can be arbitrary logical formulas (i.e. conjunctions of possibly negated predicates). In our case, the rules relate the input at a stage $z^{(s)}$ to the output $y^{(s)}$. To incorporate these rules, we introduce a function approximator for the stage $f_j^{(s)}$ and a rule $Precondition(z^{(s)}) \rightarrow f^{(s)}(z^{(s)})$. Introducing rules as function approximators allows us to combine domain knowledge expressed as rules with arbitrary function approximators.

C. Multiple functions and domain knowledge as regularizers: We incorporate the above multiple function approximator and domain knowledge constraints described above in our CCM

model as regularizers. We re-write the constraints using Lukasiewicz soft logic [183] and introduce the extent of violation of these constraints as additional regularizers. Lukasiewicz logic [183]: $A \wedge B = \max\{A + B - 1, 0\}$, $A \vee B = \min\{A + B, 1\}$, $\neg A = 1 - A$, and $A \rightarrow B = \min\{1 - A + B, 1\}$. We add these terms as additional regularizers and solve these using CCMs [].

D. Inference and Learning: As in CCMs, we pose inference as an optimization problem and we use the structured perceptron algorithm for learning.

7.3.2 Combining Text and Diagram representations

Given the text and diagram representations, we combine the two to form a joint representation of the entire question. This is similar to the previous work in diagram parsing by [253] when they built GEOS, a solver for geometry problems. We model this as a combinatorial search in the space of literals. P2P efficiently searches this combinatorial space taking advantage of a sub-modular set function that scores a subset of literals using both text and diagram. We define the best subset of literals is the one that has a high *affinity* with both text and diagram and is *coherent* i.e., does not suffer from redundancy. More formally,

$$L^* = \arg \max_{L' \subset L} \lambda \underbrace{\mathcal{A}(L', t, d)}_{\text{Affinity}} + \underbrace{\mathcal{H}(L', t, d)}_{\text{Coherence}}, \quad (7.3)$$

where $\mathcal{A}(L', t, d)$ measures the affinity of the literals in L' with both the text and the diagram, $\mathcal{H}(L', t, d)$ measures the coverage of the literals in L' compared to the text and discourages redundancies, and λ is a trade-off parameter between \mathcal{A} and \mathcal{H} .

The affinity \mathcal{A} is decomposed into text-based affinity, $\mathcal{A}_{\text{text}}$, and diagram-based affinity, $\mathcal{A}_{\text{diagram}}$. The text-based affinity closely mirrors the linguistic structure of the sentences as well as type matches in the language Ω . For modeling the text score for each literal, we learn a log-linear model. The diagram-based affinity $\mathcal{A}_{\text{diagram}}$ grounds literals into the diagram, and scores literals according to the diagram parse. $\mathcal{A}_{\text{text}}$ and $\mathcal{A}_{\text{diagram}}$ are the “true” function values of the trained final text and diagram parse pipelines.

Here, we describe the details of the objective function (Equation 7.3) and how to efficiently maximize it. The integrated affinity score of a set of literals L' (the first term in Equation 7.3) is defined as:

$$\mathcal{A}(L', t, d) = \sum_{l'_j \in L'} [\mathcal{A}_{\text{text}}(l'_j, t) + \mathcal{A}_{\text{diagram}}(l'_j, d)]$$

where $\mathcal{A}_{\text{text}}$ and $\mathcal{A}_{\text{diagram}}$ are the text and diagram affinities of l'_j , respectively.

To encourage P2P to pick a subset of literals that cover the concepts in the question text and, at the same time, avoid redundancies, we define the coherence function as:

$$\mathcal{H}(L', t, d) = N_{\text{covered}}(L') - R_{\text{redundant}}(L')$$

where N_{covered} is the number of the concept nodes used by the literals in L' , and $N_{\text{redundant}}$ is the number of redundancies among the concept nodes of the literals. To account for the different scales between \mathcal{A} and \mathcal{H} , we use the trade-off parameter λ in Equation 7.3 learned on the validation dataset.

Axiom	Premise	Conclusion
Midpoint Definition	midpoint(M, AB)	length(AM) = length(MB)
Angle Addition	interior(D, ABC)	angle(ABC) = angle(ABD) + angle(DBC)
Supplementary Angles	perpendicular(AB,CD) \wedge liesOn(C,AB)	angle(ACD) + angle(DCB) = 180°
Vertically Opp. Angles	intersectAt(AB, CD, M)	angle(AMC) = angle(BMD)

Table 7.3: Examples of geometry theorems as horn clause rules.

Maximizing the objective function in Equation 7.3 is an NP-hard combinatorial optimization problem. However, we show that our objective function is submodular (see appendix for the proof of submodularity). This means that there exists a greedy method that can provide a reliable approximation. P2P greedily maximizes Equation 7.3 by starting from an empty set of literals and adding the next literal l_j that maximizes the gain of the objective function until the gain becomes negative.

7.3.3 Probabilistic Logic for Geometry Question Answering

We build an axiomatic solver that performs logical inference with the domain knowledge of geometry and the formal problem description obtained from our parser output. We represent theorems as horn clause rules that map a premise in the logical language to a conclusion in the same language. Table 7.3 gives some examples of geometry theorems written as horn clause rules. The free variables in the theorems are universally quantified. The variables are also typed. For example, ABC can be of type *triangle* or *angle* but not *line*. Let \mathcal{T} be the set of theorems. Formally, each theorem $t \in \mathcal{T}$ maps a logical formula $l_t^{(pr)}$ corresponding to the premise to a logical formula $l_t^{(co)}$ corresponding to the conclusion.

A sample logical program (in prolog notation) that solves the problem in Figure 7.2 is given in Figure 7.5. The logical program has a set of declarations from the *GEOS* text and diagram parsers which describe the problem specification and the parsed horn clause rules describe the underlying theory. Normalized confidence scores from question text, diagram and axiom extraction models are used as probabilities in the program. Next, we describe how we harvest structured axiomatic knowledge from textbooks.

7.3.4 Harvesting Axiomatic Knowledge from Textbooks

We present an automatic approach that can (a) harvest such subject knowledge from textbooks, and (b) parse the extracted knowledge to structured programs that the solvers can use. Unlike information extraction systems trained on domains such as web documents [38, 76, inter alia], learning an information extraction system that can extract axiomatic knowledge from textbooks is challenging because of the small amount of in-domain labeled data available for these tasks. We tackle this challenge by (a) leveraging the *redundancy* and *shared ordering* of axiom mentions

Datastructures	↑↓	<pre> sort point = {A, B, C, D, O, M} sort line = {AB, BC, CA, BD, DA, OA, OM} //Symmetrically define BA, CB, ... sort angle = {ABC, BCA, CAB, ABD, BDA, DAB, AMO, MOA, OAM, BMO} //Symmetrically define CBA, ACB, ... sort triangle = {ABC, ABD, AMO} //Symmetrically define CBA, ACB, ... sort circle = {O} </pre>
Diagram Parse	↑↓	<pre> 0.4 perpendicular(OM, AB) 0.8 measure(ADB, x) 0.9 liesOn(A, O) 0.9 liesOn(B, O) 0.9 liesOn(C, O) 0.9 liesOn(D, O) 0.9 liesOn(M, AB) 0.9 liesInInterior(M, AOB) </pre>
Text Parse	↑↓	<pre> 0.9 measure(OAM, 30) 0.9 measure(radius(O), 4 cm) 0.9 query(x, _) </pre>
Axiomatic Rules	↑↓	<pre> 1 0.8 measure(ABC, 90.0) :- perpendicular(AB, CD), liesOn(B, CD) 2 0.8 measure(XAC, 180-t) :- liesOn(A, BC), measure(XAB, t) 3 0.7 equals(length(AX), length(XB)) :- liesOn(A, O), liesOn(B, O), perpendicular(OX, AB), liesOn(X, AB) 4 0.7 similar(ABC, DEF) :- equals(length(BC), length(EF)), equals(measure(ABC), measure(DEF)), equals(measure(BCA), measure(EFD)) // ASA rule. Similar rules for SAS, SSS, RHS rules of similarity 5 0.7 equals(measure(CAB), measure(FED)) :- similar(ABC, DEF) // Similar rules for other corresponding angles 6 0.7 equals(measure(ABC), u+v) :- equals(measure(ABD), u), equals(measure(DBC), v), liesInInterior(D, ABC) 7 0.6 equals(measure(ADB), t/2) :- equals(measure(AOB), t), liesOn(A, O), liesOn(B, O) </pre>

Figure 7.5: A sample logical program (in prolog style) that solves the problem in Figure 7.2. The program consists of a set of data structure declarations that correspond to types in the prolog program, a set of declarations from the diagram and text parse and a subset of the geometry axioms written as horn clause rules. The axioms are used as the underlying theory with the aforementioned declarations to yield the solution upon logical inference. Normalized confidence weights from the diagram, text and axiom parses are used as probabilities. For readers understanding, we list the axioms in the order (1 to 7) they are used to solve the problem. However, this ordering is not required. Other (less probable) declarations and axiom rules are not shown here for clarity but they can be assumed to be present.

Theorem 8.4 Pythagorean Theorem

In a right triangle, the sum of the squares of the measures of the legs equals the square of the measure of the hypotenuse.

Symbols: $a^2 + b^2 = c^2$

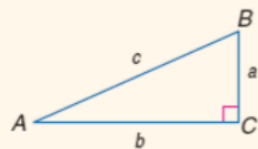


Figure 7.6: An excerpt of a textbook from our dataset that introduces the Pythagoras theorem. The textbook has a lot of typographical features that can be used to harvest this theorem: The textbook explicitly labels it as a “theorem”; there is a colored bounding box around it; an equation writes down the rule and there is a supporting figure. Our models leverages such rich contextual and typographical information (when available) to accurately harvest axioms and then parses them to horn-clause rules. The horn-clause rule derived by our approach for the Pythagoras theorem is: $isTriangle(ABC) \wedge perpendicular(AC, BC) \implies BC^2 + AC^2 = AB^2$.

across multiple textbooks⁴, and (b) utilizing rich contextual and typographical features⁵ from textbooks to effectively extract and parse axioms. Finally, we also provide an approach to parse the extracted axiom mentions from various textbooks and reconcile them to achieve the best program for each axiom.

While *GEOS* has its basis in coordinate geometry and indeed works, it has some key issues: *GEOS* requires an explicit mapping of each predicate into a set of constraints over point coordinates⁶. These constraints can be non-trivial to write, requiring significant manual engineering. As a result, *GEOS*'s constraint set is incomplete and it cannot solve a number of SAT style geometry questions. Furthermore, this solver is not interpretable. As our user studies show, it is not natural for a student to understand the solution of these geometry questions in terms of satisfiability of constraints over coordinates. A more natural way for students to understand and reason about these questions is through deductive reasoning using axioms of geometry⁷.

We use our model to extract and parse axiomatic knowledge from a novel dataset of 20 publicly available math textbooks. We use this structured axiomatic knowledge to build a new axiomatic solver that performs logical inference to solve geometry problems. Our axiomatic solver outperforms *GEOS* on all existing test sets introduced in [253] as well as a new test set of geometry questions collected from these textbooks. We also performed user studies on a number of school students studying geometry who found that our axiomatic solver is more *interpretable*

⁴The same axiom can be potentially mentioned in a number of textbooks in different ways. All textbooks typically introduce axioms in roughly the same order – for example, pythagorous theorem would typically be introduced after introducing the notion of a right angled triangle.

⁵Textbooks contain rich context and typographical information (see Figure 7.6 for an illustrative example). We use this rich information as features in our model.

⁶For example, the predicate $isPerpendicular(AB, CD)$ is mapped to the constraint $\frac{y_B - y_A}{x_B - x_A} \times \frac{y_D - y_C}{x_D - x_C} = -1$.

⁷For example, the deductive reasoning required to solve the question in Figure 7.2 is: (1) Use the axiom that the sum of interior angles of a triangle is 180° and the fact that $\angle AMO$ is 90° to conclude that $\angle MOA$ is 60° . (2) $\triangle MOA \sim \triangle MOB$ (using a similar triangle axiom) and then, $\angle MOB = \angle MOA = 60^\circ$ (using the axiom that corresponding angles of similar triangles are equal). (3) Use angle sum rule to conclude that $\angle AOB = \angle MOB + \angle MOA = 120^\circ$. (4) Use the axiom that the angle subtended by an arc of a circle at the centre is double the angle subtended by it at any point on the circle to conclude that $\angle ADB = 0.5 \times \angle AOB = 60^\circ$.

and *useful* compared to *GEOS*.

We present a structured prediction model that identifies axioms in textbooks and then parses them. Since harvesting axioms from a single textbook is a very hard problem, we use multiple textbooks and leverage the redundancy of information to accurately extract and parse axioms. We first define a joint model that identifies axiom mentions in each textbook and aligns repeated mentions of the same axiom across textbooks. Then, given a set of axioms (with possibly, multiple mentions of each axiom), we define a parsing model that maps each axiom to a horn clause rule by utilizing the various mentions of the axiom.

Given a set of textbooks \mathcal{B} in machine readable form (XML in our experiments), we extract chapters relevant for geometry in each of them to obtain a sequence of sentences (with associated typographical information) from each textbook. Let $\mathbf{S}_b = \{s_0^{(b)}, s_1^{(b)}, \dots, s_{|\mathbf{S}_b|}^{(b)}\}$ denote the sequence of sentences in textbook b . $|\mathbf{S}_b|$ denotes the number of sentences in textbook b .

Axiom Identification and Alignment

We decompose the problem of extracting axioms from textbooks into two tractable sub-problems: (a) identification of axiom mentions in each textbook using a sequence labeling approach, and (b) aligning repeated mentions of the same axiom across textbooks. Then, we combine the learned models for these sub-problems into a joint optimization framework that simultaneously learns to identify and align axiom mentions. Joint modeling of the axiom identification and alignment is necessary as both sub-problems can help each other.

Axiom Identification: Linear-chain CRF formulation [160] can be used for the subproblem of axiom identification. Given $\{\mathbf{S}_b | b \in \mathcal{B}\}$, the model labels each sentence $s_i^{(b)}$ as **Before**, **Inside** or **Outside** an axiom. Hereon, a contiguous block of sentences labeled **B** or **I** will be considered as an axiom mention. Let $\mathcal{T} = \{\mathbf{B}, \mathbf{I}, \mathbf{O}\}$ denote the tag set. Let $y_i^{(b)}$ be the tag assigned to $s_i^{(b)}$ and \mathbf{Y}_b be the tag sequence assigned to \mathbf{S}_b . The CRF defines:

$$p(\mathbf{Y}_b | \mathbf{S}_b; \boldsymbol{\theta}) \propto \prod_{k=1}^{|\mathbf{S}_b|} \exp \left(\sum_{i,j \in \mathcal{T}} \boldsymbol{\theta}_{ij}^T \mathbf{f}_{ij}(y_{k-1}^{(b)}, y_k^{(b)}, \mathcal{S}_b) \right)$$

We find the parameters $\boldsymbol{\theta}$ using maximum-likelihood estimation with L2 regularization:

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta}} \sum_{b \in \mathcal{B}} \log p(\mathbf{Y}_b | \mathbf{S}_b; \boldsymbol{\theta}) - \lambda \|\boldsymbol{\theta}\|_2^2$$

We use L-BFGS to optimize the objective and Viterbi decoding for inference.

Features: Features f look at a pair of adjacent tags $y_{k-1}^{(b)}, y_k^{(b)}$, the input sequence \mathbf{S}_b , and where we are in the sequence. The features (listed in Table 7.4) include various content based features encoding various notions of similarity between pairs of sentences as well as various typographical features such as whether the sentences are annotated as an axiom (or theorem or corollary) in the textbook, contain equations, diagrams, text that is bold or italicized, are in the same node of the xml hierarchy, are contained in a bounding box, etc. Some extracted axiom

Content	Sentence Overlap	Semantic Textual Similarity between the current and next sentence. We include features that compute the proportion of common unigrams and geometry entities (constants, predicates and functions) across the two sentences. This feature is conjoined with the tag assigned to the current and next sentence.
	Geometry entities	No. of geometry entities (normalized by the number of tokens) in this sentence. This feature is conjoined with the tag assigned to the current sentence.
	Intra-sentence semantics	Indicator that the current sentence contains any one of the following words: <i>hence, if, equal, twice, proportion, ratio, product</i> . This feature is conjoined with the tag assigned to the current sentence.
Typography	Axiom, Theorem, Corollary Mention	(a) The current (or previous) sentence is mentioned as an Axiom, Theorem or Corollary e.g. <i>Similar Triangle Theorem</i> or <i>Corollary 2.1</i> . (b) The section or subsection in the textbook containing the current (or previous) sentence mentions an Axiom, Theorem or Corollary. This feature is conjoined with the tag assigned to the current (and previous) sentence.
	Eqn. Template	The current (or next) sentence contains an equation eg. $PA \times PB = PT^2$. This feature is conjoined with the tag assigned to the current (and next) sentence.
	Assoc. Diagram	The current sentence contains a pointer to a figure eg. "Figure 2.1". This feature is conjoined with the tag assigned to the current sentence.
	RST edge	Indicator for the RST relation between the current and next sentence. This feature is conjoined with the tag assigned to the current and next sentence.
	Bold/Underline	The sentence (or previous) sentence contains text that is in bold font or underlined. Conjoined with the tag assigned to the current (and previous) sentence.
	XML structure	Indicator that the current and previous sentence are in the same node of the XML hierarchy. Conjoined with the tag assigned to the current and previous sentence.
	Bounding box	Indicator that the current and previous sentence are bounded by a bounding box in the textbook. Conjoined with the tag assigned to the current and previous sentence.

Table 7.4: Feature set for our axiom identification model. The features are based on content and typography.

mentions contain pointers to a diagram eg. “Figure 2.1”. We consider the diagram to be a part of the axiom mention.

Axiom Alignment: Next, we leverage the redundancy of information and the relatively fixed ordering of axioms in various textbooks by aligning various mentions of the same axiom across textbooks and introducing structural constraints on the alignment.

Let $\mathbf{A}_b = (A_1^{(b)}, A_2^{(b)}, \dots, A_{|\mathbf{A}_b|}^{(b)})$ be the axiom mentions extracted from textbook b . Let \mathbf{A} denote the collection of axiom mentions extracted from all textbooks. We assume a global ordering of axioms $\mathbf{A}^* = (A_1^*, A_2^*, \dots, A_U^*)$ where U is some pre-defined upper bound on the total number of axioms in geometry. Then, we emphasize that the axiom mentions extracted from each textbooks (roughly) follow this ordering. Let $Z_{ij}^{(b)}$ be a random variable that denotes if axiom $A_i^{(b)}$ extracted from book b refers to the global axiom A_j^* . We introduce a log-linear model that factorizes over alignment pairs:

$$P(\mathbf{Z}|\mathbf{A};\phi) = \frac{1}{Z(\mathbf{A};\phi)} \times \exp \left(\sum_{\substack{b_1, b_2 \in \mathcal{B} \\ b_1 \neq b_2}} \sum_{1 \leq k \leq U} \sum_{\substack{1 \leq i \leq |\mathbf{A}_{b_1}| \\ 1 \leq j \leq |\mathbf{A}_{b_2}|}} Z_{ik}^{(b_1)} Z_{jk}^{(b_2)} \phi^T \mathbf{g}(A_i^{(b_1)}, A_j^{(b_2)}) \right)$$

Here, $Z(\mathbf{A};\phi)$ is the partition function of the log-linear model. \mathbf{g} denotes the feature function described later. We introduce the following constraints on the alignment structure:

C1: An axiom appears in one book at-most once

C2: An axiom refers to exactly one theorem in the global ordering

C3: Ordering Constraint: If i^{th} axiom in a book refers to the j^{th} axiom in the global ordering then no axiom succeeding the i^{th} axiom can refer to a global axiom preceding j .

Learning with Hard Constraints: We find the optimal parameters ϕ using maximum-likelihood estimation with L2 regularization:

$$\phi^* = \arg \max_{\phi} \log P(\mathbf{Z}|\mathbf{A};\phi) - \mu \|\phi\|_2^2$$

We use L-BFGS to optimize the objective. To compute feature expectations appearing in the gradient of the objective, we use a Gibbs sampler. The sampling equations for Z_{ik}^b are:

$$P(Z_{ik}^{(b)} | rest) \propto \exp(T_b(i, k)) \quad (1)$$

$$T_b(i, k) = Z_{ik}^{(b)} \sum_{\substack{b' \in \mathcal{B} \\ b' \neq b}} \sum_{1 \leq j \leq |\mathbf{A}_{b'}|} Z_{jk}^{(b')} \phi^T \mathbf{g}(A_i^{(b)}, A_j^{(b')})$$

Note that the constraints $C1 \dots C3$ define the feasible space of alignments. Our sampler always samples the next $Z_{ik}^{(b)}$ in this feasible space.

Learning with Soft Constraints: We might want to treat some constraints, in particular, the ordering constraints C3 as soft constraints. We can write down the constraint C3 using the alignment variables:

$$\begin{aligned} Z_{ij}^{(b)} &\leq 1 - Z_{kl}^{(b)} \\ \forall 1 \leq i < k \leq |A_b|, 1 \leq l < j \leq U \\ \forall b \in \mathcal{B} \end{aligned}$$

To model these constraints as soft constraints, we penalize the model for violating these constraints. Let the penalty for violating the above constraint be $\exp\left(v \max\left(0, 1 - Z_{ij}^{(b)} - Z_{kl}^{(b)}\right)\right)$. We introduce a new regularization term: $\mathbf{R}(\mathbf{Z}) = \sum_{\substack{1 \leq i < k \leq |A_b| \\ 1 \leq l < j \leq U \\ b \in \mathcal{B}}} \exp\left(v \max\left(0, 1 - Z_{ij}^{(b)} - Z_{kl}^{(b)}\right)\right)$. Here v is a hyper-parameter to tune the cost of violating a constraint. We write down the following regularized objective:

$$\phi^* = \arg \max_{\phi} \log P(\mathbf{Z}|\mathbf{A}; \phi) - \mathbf{R}(\mathbf{Z}) - \mu \|\phi\|_2^2$$

We use L-BFGS to find the optimal parameters ϕ^* . We perform Gibbs sampling to compute feature expectations. The sampling equation for $Z_{ik}^{(b)}$ is similar (eq 1), but:

$$\begin{aligned} T_b(i, k) &= \sum_{\substack{b' \in \mathcal{B} \\ b' \neq b}} \sum_{1 \leq j \leq |A_{b'}|} Z_{ik}^{(b)} Z_{jk}^{(b')} \phi^T \mathbf{g}(A_i^{(b)}, A_j^{(b')}) \\ &+ v \sum_{\substack{b' \in \mathcal{B} \\ b' \neq b}} \sum_{i < j \leq |A_{b'}|} \sum_{1 \leq l < k} \left(1 - Z_{ik}^{(b)} - Z_{jl}^{(b')}\right) \\ &+ v \sum_{\substack{b' \in \mathcal{B} \\ b' \neq b}} \sum_{1 \leq j < i} \sum_{k < l \leq U} \left(1 - Z_{ik}^{(b)} - Z_{jl}^{(b')}\right) \end{aligned}$$

Features: Now, we describe the features g . These too include content based features encoding various notions of similarity between pairs of axiom mentions as well as various typographical features. The features are listed in Table 7.5.

Joint Identification and Alignment: Joint modeling of axiom identification and alignment components is useful as both problems potentially help each other. Let $Y_{ij}^{(b)}$ denote that the sentence $s_i^{(b)}$ from book b has tag j . We reuse the definitions of the alignment variables $Z_{ij}^{(b)}$ as before. We further define $Z_{i0}^{(b)}$ such that it denotes that the i^{th} axiom in textbook b is not aligned to any global axiom. We again define a log-linear model with factors that score axiom identification and axiom alignments.

$$p(\mathbf{Y}, \mathbf{Z}|\{\mathbf{S}_b\}; \theta, \phi) \propto f_{AI}(\mathbf{Y}|\{\mathbf{S}_b\}; \theta) \times f_{AA}(\mathbf{Z}|\mathbf{Y}, \{\mathbf{S}_b\}; \phi)$$

Here, the factors:

Unigram, Bigram, Dependency and Entity Overlap	Real valued features that compute the proportion of common unigrams, bigrams, dependencies and geometry entities (constants, predicates and functions) across the two axioms. When comparing geometric entities, we include geometric entities derived from the associated diagrams when available.
Longest Common Subsequence	Real valued feature that computes the length of longest common subsequence of words between two axiom mentions normalized by the total number of words in the two mentions.
Number of sentences	Real valued feature that computes the absolute difference in the number of sentences in the two mentions.
Alignment Scores	We use an off-the-shelf monolingual word aligner – <i>JACANA</i> [300] pre-trained on PPDB – and compute alignment score between axiom mentions as the feature.
MT Metrics	We use two common MT evaluation metrics <i>METEOR</i> [66] and <i>MAXSIM</i> [37], and use the evaluation scores as features. While <i>METEOR</i> computes n-gram overlaps controlling on precision and recall, <i>MAXSIM</i> performs bipartite graph matching and maps each word in one axiom to at most one word in the other.
Summarization Metrics	We also use <i>Rouge-S</i> [172], a text summarization metric, and use the evaluation score as a feature. <i>Rouge-S</i> is based on skip-grams.
Equation Template	Indicator feature that matches templates of equations detected in the axiom mentions.
Image Caption	Proportion of common unigrams in the image captions of the diagrams associated with the axiom mentions. If both mentions do not have associated diagrams, this feature doesn't fire.
XML structure	Indicator matching the current (and parent) node of axiom mentions in respective XML hierarchies.

Table 7.5: Feature set for our axiom alignment model. The features are based on content, structure and typography.

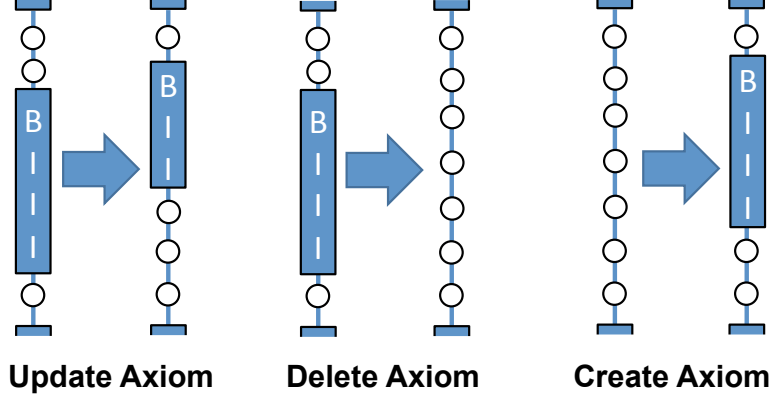


Figure 7.7: An illustration of the three operations to sample axiom blocks.

$$f_{AI} = \exp\left(\sum_{b \in \mathcal{B}} \sum_{k=1}^{|\mathcal{S}_b|} \sum_{i,j \in \mathcal{T}} Y_{k-1i}^{(b)} Y_{kj}^{(b)} \boldsymbol{\theta}_{ij}^T \mathbf{f}_{ij}(i, j, \mathcal{S}_b)\right)$$

$$f_{AA} = \exp\left(\sum_{\substack{b_1, b_2 \in \mathcal{B} \\ b_1 \neq b_2}} \sum_{1 \leq k \leq U} \sum_{\substack{1 \leq i \leq |\mathbf{A}_{b_1}| \\ 1 \leq j \leq |\mathbf{A}_{b_2}|}} Z_{ik}^{(b_1)} Z_{jk}^{(b_2)} \boldsymbol{\phi}^T \mathbf{g}(A_i^{(b_1)}, A_j^{(b_2)}))\right)$$

We write down the model constraints below:

C1': Every sentence has a unique label

C2': Tag O cannot be followed by tag I

C3': Consistency between Y 's and Z 's i.e. axiom boundaries defined by Y 's and Z 's must agree.

C4' = C3.

We use L-BFGS for learning. To compute feature expectations, we use a Metropolis Hastings sampler that samples \mathbf{Y} 's and \mathbf{Z} 's alternatively. Sampling for \mathbf{Z} 's reduces to Gibbs sampling and the sampling equations are as same as before (Section 7.3.4). For better mixing, we sample \mathbf{Y} in blocks. Consider blocks of \mathbf{Y} 's which denote axiom boundaries at time stamp t , we define three operations to sample axiom blocks at the next time stamp. The operations (shown in Figure 7.7) are:

1. **Update axiom:** The axiom boundary can be shrunk, expanded or moved. The new axiom, however, cannot overlap with other axioms.
2. **Delete axiom:** The axiom can be deleted by labeling all its sentences as O .
3. **Introduce axiom:** Given a contiguous sequence of sentences labeled O , a new axiom can be introduced.

Note that these three operations define an ergodic Markov chain. We use the axiom identification part of the model as the proposal:

$$Q(\bar{\mathbf{Y}}|\mathbf{Y}) \propto \exp\left(\sum_{b \in \mathcal{B}} \sum_{k=1}^{|\mathcal{S}_b|} \sum_{i,j \in \mathcal{T}} \bar{Y}_{k-1i}^{(b)} \bar{Y}_{kj}^{(b)} \boldsymbol{\theta}_{ij}^T \mathbf{f}_{ij}(i, j, \mathcal{S}_b)\right)$$

Hence, the acceptance ratio only depends on the alignment part of the model: $R(\bar{\mathbf{Y}}|\mathbf{Y}) = \min\left(1, \frac{U(\bar{\mathbf{Y}})}{U(\mathbf{Y})}\right)$ where $U(\mathbf{Y}) = f_{AA}$. We again have two variants, where we model the ordering constraints (C4') as soft or hard constraints.

Axiom Parsing

After harvesting axioms, we build a parser for these axioms that maps raw axioms to horn clause rules. The axiom harvesting step provides us a multi-set of axiom extractions. Let $\mathcal{A} = \{\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_{|\mathcal{A}|}\}$ represent the multi-set where each axiom \mathbf{A}_i is mentioned at least once.

First, we describe a base parser that parses axiom mentions to horn clause rules. Then, we utilize the redundancy of axiom extractions from various sources (textbooks) to improve our parser.

Base Axiomatic Parser: Our base parser identifies the *premise* and *conclusion* portions of each axiom and then uses *GEOS*'s text parser to parse the two portions into a logical formula. Then, the two logical formulas are put together to form horn clause rules.

Axiom mentions (for example, the Pythagoras theorem mention in Figure 7.6) are often accompanied by equations or diagrams. When the mention has an equation, we simply treat the equation as the *conclusion* and the rest of the mention as the *premise*. When the axiom has an associated diagram, we always include the diagram in the *premise*. We learn a model to predict the split of the axiom text into two parts forming the *premise* and the *conclusion* spans. Then, the *GEOS* parser maps the *premise* and *conclusion* spans to *premise* and *conclusion* logical formulas, respectively.

Let Z_s represent the split that demarcates the *premise* and *conclusion* spans. We score the axiom split as a log-linear model: $p(Z_s|a; \mathbf{w}) \propto \exp(\mathbf{w}^T \mathbf{h}(a, Z_s))$. Here, \mathbf{h} are feature functions described later. We found that in most cases (>95%), the premise and conclusion are contiguous spans in the axiom mention where the left span corresponds to the *premise* and the right span corresponds to the *conclusion*. Hence, we search over the space of contiguous spans to infer Z_s . We use L-BGFGS for learning.

Features: We list the features \mathbf{h} in Table 7.6. The features are defined over candidate spans forming the text split, are strongly inspired from rhetorical structure theory [185] and previous works on discourse parsing [187, 266]. Given a beam of *Premise* and *Conclusion* splits, we use the *GEOS* parser to get *Premise* and *Conclusion* logical formulas for each split in the beam and obtain a beam of axiom parses for each axiom in each textbook.

Multi-source Axiomatic Parser: Now, we describe a multi-source parser that utilizes the redundancy of axiom extractions from various sources (textbooks). Given a beam of 10-best parses for each axiom from each source, we use a number of heuristics to determine the best parse for the axiom:

1. **Majority Voting:** For each axiom, pick the parse that occurs most frequently across beams.
2. **Average Score:** Pick the parse that has the highest average parse score (only counting top 5 parses for each source), for each axiom.
3. **Learn Source Confidence:** Learn a set of weights $\{\mu_1, \mu_2, \dots, \mu_S\}$, one for each source and then picks the parse that has the highest average weighted parse score for each axiom.
4. **Predicate Score:** Instead of selecting from one of the top parses across various sources,

Discourse Markers	Discourse markers (connectives, cue-words or cue-phrases, etc) have been shown to give good indications on discourse structure [187]. We build a list of discourse markers using the training set, considering the first and last tokens of each span, culled to top 100 by frequency. We use these 100 discourse markers as features. We repeat the same procedure by using part-of-speech (POS) instead of words and use them as features.
Punctuation	Punctuation at the segment border is an excellent cue. We include indicator features whether there is a punctuation at the segment border.
Text Organization	Indicator that the two text spans are part of the same (a) sentence, (b) paragraph.
XML Structure	Indicator that the two spans are in the same node in the XML hierarchy. Conjoined with the indicator feature that the two spans are part of the same paragraph.
RST Parse	We use an off-the-shelf RST parser [86] and include an indicator feature that the segmentation matches the parse segmentation. We also include the RST label as a feature.
Span Lengths	The distribution of the two text spans is typically dependent on their lengths. We use the ratio of the length of the two spans as an additional feature.
Soricut and Marcu Segmenter	[266] (section 3.1) presented a statistical model for deciding elementary discourse unit boundaries. We use the probability given by this model retrained on our training set as feature. This feature uses both lexical and syntactic information.
Head/Common Ancestor/Attachment Node	Head node is the word with the highest occurrence as a lexical head in the lexicalized tree among all the words in the text span. The attachment node is the parent of the head node. We have features for the head words of the left and right spans, the common ancestor (if any), the attachment node and the conjunction of the two head node words. We repeat these features with part-of-speech (POS) instead of words.
Syntax	Distance to (a) root (b) common ancestor for the nodes spanning the respective spans. We use these distances, and the difference in the distances as features.
Dominance	<i>Dominance</i> [266] is a key idea in discourse which looks at syntax trees and studies sub-trees for each span to infer a logical nesting order between the two. We use the dominance relationship as a feature. See [266] for details.
Span Similarity	Proportion of (a) words (b) geometry relations (c) relation-arguments shared by the two spans.
No. of Relations	Number of geometry relations represented in the two spans. We use the Lexicon Map from GEOS to compute the number of expressed geometry relations.
Relative Position	Relative position of the two lexical heads and the text split in sentence.

Table 7.6: Feature set for our axiom parsing model.

	Strict Comp.			Relaxed Comp.		
	P	R	F	P	R	F
Identification	64.3	69.3	66.7	84.3	87.9	86.1
Joint-Hard	68.0	68.1	68.0	85.4	87.1	86.2
Joint-Soft	69.7	71.1	70.4	86.9	88.4	87.6

Table 7.7: Test set Precision, Recall and F-measure scores for axiom identification when performed alone and when performed jointly with axiom alignment. We show results for both strict as well as relaxed comparison modes. For the joint model, we show results when we model ordering constraints as hard or soft constraints.

treat each axiom parse as a bag of premise predicates and a bag of conclusion predicates. Then, pick a subset of premise and conclusion predicates for the final parse using average scoring with thresholding.

7.4 Experiments

Datasets: We use a collection of grade 6-10 Indian high school math textbooks by four publishers/authors – *NCERT*, *R S Aggarwal*, *R D Sharma* and *M L Aggarwal* – a total of $5 \times 4 = 20$ textbooks to validate our model. Millions of students in India study geometry from these books every year and these books are readily available online. We manually marked chapters relevant for geometry in these books and then parsed them using Adobe Acrobat’s *pdf2xml* parser and AllenAI’s *Science Parse* project⁸. Then, we annotated geometry axioms, alignments and parses for grade 6, 7 and 8 textbooks by the four publishers/authors. We use grade 6, 7 and 8 textbook annotations for development, training, and testing, respectively. All the hyper-parameters in all the models are tuned on the development set using grid search.

GEOS used 13 types of entities and 94 functions and predicates. We add some more entities, functions and predicates to cover other more complex concepts in geometry not covered in *GEOS*. Thus, we obtain a final set of 19 entity types and 115 functions and predicates for our parsing model. We use Stanford CoreNLP [186] for feature generation. We use two datasets for evaluating our system: (a) practice and official SAT style geometry questions used in *GEOS*, and (b) an additional dataset of geometry questions collected from the aforementioned textbooks. This dataset consists of a total of 1406 SAT style questions across grades 6-10, and is approximately 7.5 times the size of the dataset used in *GEOS*. We split the dataset into training (350 questions), development (150 questions) and test (906 questions) with equal proportion of grade 6-10 questions. We annotated the 500 training and development questions with ground-truth logical forms. We use the training set to train another version of *GEOS* with expanded set of entity types, functions and predicates. We call this system *GEOS++*.

Results: We first evaluate the axiom identification, alignment and parsing models individually. For axiom identification, we compare the results of automatic identification with gold axiom identifications and compute the precision, recall and F-measure on the test set. We use strict as

⁸<https://github.com/allenai/science-parse>

	P	R	F	NMI
Alignment	71.8	74.8	73.3	0.60
Joint-Hard	75.0	76.4	75.7	0.65
Joint-Soft	79.3	81.4	80.3	0.69

Table 7.8: Test set Precision, Recall, F-measure and NMI scores for axiom alignment when performed alone and when performed jointly with axiom identification. For the joint model, we show results when we model ordering constraints as hard or soft constraints.

		Literals			Full Parse		
		P	R	F	P	R	F
GEOS		86.7	70.9	78.0	64.2	56.6	60.2
Single Src.		91.6	75.3	82.6	68.8	60.4	64.3
GEOS++	Maj. Voting	90.2	78.5	83.9	70.0	63.3	66.5
	Avg. Score	90.8	79.6	84.9	71.7	66.4	69.0
	Src. Confid.	91.0	79.9	85.1	73.3	68.1	70.6
	Pred. Score	92.8	82.8	87.5	76.6	70.1	73.2

Table 7.9: Test set Precision, Recall and F-measure scores for axiom parsing. These scores are computed over literals derived in axiom parses or full axiom parses. We show results for the old *GEOS* system, for the improved *GEOS++* system with expanded entity types, functions and predicates, and for the multi-source parsers presented in this paper.

well as relaxed comparison. In strict comparison mode the automatically identified mentions and gold mentions must match exactly to get credit, whereas, in the relaxed comparison mode only a majority (>50%) of sentences in the automatically identified mentions and gold mentions must match to get credit. Table 7.7 shows the results of axiom identification where we clearly see improvements in performance when we jointly model axiom identification and alignment. This is due to the fact that both the components reinforce each other. We also observe that modeling the ordering constraints as soft constraints leads to better performance than modeling them as hard constraints. This is because the ordering of presentation of axioms is generally (yet not always) consistent across textbooks.

To evaluate axiom alignment, we first view it as a series of decisions, one for each pair of axiom mentions and compute precision, recall and F-score by comparing automatic decisions with gold decisions. Then, we also use a standard clustering metric, Normalized Mutual Information (NMI) [271] to measure the quality of axiom mention clustering. Table 7.8 shows the results on the test set when gold axiom identifications are used. We observe improvements in axiom alignment performance too when we jointly model axiom identification and alignment jointly both in terms of F-score as well as NMI. Modeling ordering constraints as soft constraints again leads to better performance than modeling them as hard constraints in terms of both metrics.

To evaluate axiom parsing, we compute precision, recall and F-score in (a) deriving literals in axiom parses, as well as for (b) the final axiom parses on our test set. Table 7.9 shows the results of axiom parsing for *GEOS* (trained on the training set) as well as various versions of our best performing system (*GEOS++* with our axiomatic solver) with various heuristics for multi-

	Practice	Official	Textbook
<i>GEOS</i>	61	49	32
Our System	64	55	51
Oracle	80	78	72

Table 7.10: Scores for solving geometry questions on the SAT practice and official datasets and a dataset of questions from the 20 textbooks. We use SAT’s grading scheme that rewards a correct answer with a score of 1.0 and penalizes a wrong answer with a negative score of 0.25. *Oracle* uses gold axioms but automatic text and diagram interpretation in our logical solver. All differences between *GEOS* and our system are significant ($p < 0.05$ using the two-tailed paired t-test).

source parsing. The results show that our system (single source) performs better than *GEOS* as it is trained with the expanded set of entity types, functions and predicates. The results also show that the choice of heuristic is important for the multi-source parser – though all the heuristics lead to improvements over the single source parser. The average score heuristic that chooses the parse with the highest average score across sources performs better than majority voting which chooses the best parse based on a voting heuristic. Learning the confidence of every source and using a weighted average is an even better heuristic. Finally, predicate scoring which chooses the parse by scoring predicates on the premise and conclusion sides performs the best leading to 87.5 F1 score (when computed over parse literals) and 73.2 F1 score (when computed on the full parse). The high F1 score for axiom parsing on the test set shows that our approach works well and we can accurately harvest axiomatic knowledge from textbooks.

Finally, we use the extracted horn clause rules in our axiomatic solver for solving geometry problems. For this, we over-generate a set of horn clause rules by generating 3 horn clause parses for each axiom and use them as the underlying theory in prolog programs such as the one shown in Figure 7.5. We use weighted logical expressions for the question description and the diagram derived from *GEOS++* as declarations, and the (normalized) score of the parsing model multiplied by the score of the joint axiom identification and alignment model as weights for the rules. Table 7.10 shows the results for our best end-to-end system and compares it to *GEOS* on the practice and official SAT dataset from [253] as well as questions from the 20 textbooks. On all the three datasets, our system outperforms *GEOS*. Especially on the dataset from the 20 textbooks (which is indeed a harder dataset and includes more problems which require complex reasoning based on geometry), *GEOS* doesn’t perform very well whereas our system still achieves a good score. *Oracle* shows the performance of our system when gold axioms (written down by an expert) are used along with automatic text and diagram interpretations in *GEOS++*. This shows that there is scope for further improvement in our approach.

Interpretability: Students around the world solve geometry problems through rigorous deduction whereas the numerical solver in *GEOS* does not provide such interpretability. One of the key benefits of our axiomatic solver is that it provides an easy-to-understand student-friendly deductive solution to geometry problems.

To test the interpretability of our axiomatic solver, we asked 50 grade 6-10 students (10 students in each grade) to use *GEOS* and our system (*GEOS++* with our axiomatic solver) as a web-based assistive tool while learning geometry. They were each asked to rate how ‘inter-

	Interpretability		Usefulness	
	<i>GEOS</i>	<i>O.S.</i>	<i>GEOS</i>	<i>O.S.</i>
Grade 6	2.7	2.9	2.9	3.2
Grade 7	3.0	3.7	3.3	3.6
Grade 8	2.7	3.5	3.1	3.5
Grade 9	2.4	3.3	3.0	3.7
Grade 10	2.8	3.1	3.2	3.8
Overall	2.7	3.3	3.1	3.6

Table 7.11: User study ratings for *GEOS* and our system (*O.S.*) by students in grade 6-10. Ten students in each grade were asked to rate the two systems on a scale of 1-5 on two facets: ‘interpretability’ and ‘usefulness’. Each cell shows the mean rating computed over ten students in that grade for that facet.

pretable’ and ‘useful’ the two systems were on a scale of 1-5. Table 7.16 shows the mean rating by students in each grade on the two facets. We can observe that students of each grade found our system to be more interpretable as well as more useful to them than *GEOS*. This study lends support to our claims about the need of an interpretable deductive solver for geometry problems.

7.4.1 A Learning from Demonstrations Approach

Cognitive science emphasizes the importance of *imitation* or *learning by example* [191, 192] in human learning. When a teacher signals a pedagogical intention, children tend to imitate the teacher’s actions [27, 31]. Inspired by this phenomenon, the *learning by demonstration* view of machine learning [9, 98, 250] assumes training data in the form of example demonstrations. A task is demonstrated by a teacher and the learner generalizes from these demonstrations in order to execute the task.

Research in question answering has traditionally focused on learning from question-answer pairs [29]. However, it is well-established in the educational psychology literature [4, 83] that children tend to learn better and faster from concrete illustrations and demonstrations. We show that we can leverage demonstrative solutions for questions as provided by a teacher to improve our question answering systems.

We solve the task of learning to solve SAT geometry problems (such as the one in Figure 7.2) using demonstrative solutions to these problems (such as the one in Figure 7.8). Such demonstrations are common in textbooks as they help students learn how to solve geometry problems effectively. We build a new dataset of demonstrative solutions of geometry problems and show that it can be used to improve *GEOS* [253], the state-of-the-art in solving geometry problems.

We also present a technique inspired from recent work in situated question answering [152] that jointly learns how to interpret the demonstration and use this interpretation to solve geometry problems. We model the interpretation task (the task of recognizing various states in the demonstration) as a semantic parsing task. We model state transitions in the demonstration via a deduction model that treats each application of a theorem of geometry as a state transition. We describe techniques to learn the two models separately as well as jointly from various kinds

1. Sum of interior angles of triangle is 180°

$$\Rightarrow \angle OAM + \angle AMO + \angle MOA = 180^\circ$$

$$\Rightarrow \angle MOA = 60^\circ$$

2. Similar triangle theorem

$$\Rightarrow \triangle MOB \sim \triangle MOA$$

$$\Rightarrow \angle MOB = \angle MOA = 60^\circ$$

3. $\angle AOB = \angle MOB + \angle MOA$

$$\Rightarrow \angle AOB = 120^\circ$$

4. Angle subtended by a chord at the center is twice the angle subtended at the circumference

$$\begin{aligned} \Rightarrow \angle ADB &= 0.5 \times \angle AOB \\ &= 60^\circ \end{aligned}$$

Figure 7.8: An example demonstration on how to solve the problem in Figure 1: (1) Use the theorem that the sum of interior angles of a triangle is 180° and additionally the fact that $\angle AMO$ is 90° to conclude that $\angle MOA$ is 60° . (2) Conclude that $\triangle MOA \sim \triangle MOB$ (using a similar triangle theorem) and then, conclude that $\angle MOB = \angle MOA = 60^\circ$ (using the theorem that corresponding angles of similar triangles are equal). (3) Use angle sum rule to conclude that $\angle AOB = \angle MOB + \angle MOA = 120^\circ$. (4) Use the theorem that the angle subtended by an arc of a circle at the centre is double the angle subtended by it at any point on the circle to conclude that $\angle ADB = 0.5 \times \angle AOB = 60^\circ$.

of supervision: (a) when we only have a set of question-answer pairs as supervision, (b) when we have a set of questions and demonstrative solutions for them, and (c) when we have a set of question-answer pairs and a set of demonstrations.

An important benefit of our approach is ‘interpretability’. While *GEOS* is uninterpretable, our approach utilizes known theorems of geometry to deductively solve geometry problems. Our approach also generates demonstrative solutions (like Figure 7.8) as a by-product which can be provided to students on educational platforms such as MOOCs to assist in their learning.

We present an experimental evaluation of our approach on the two datasets previously introduced in [253] and a new dataset collected by us from a number of math textbooks in India. Our experiments show that our approach of leveraging demonstrations improves *GEOS*. We also performed user studies with a number of school students studying geometry, who found that our approach is more *interpretable* as well as more *useful* in comparison to *GEOS*.

In contrast to *GEOS* which uses supervised learning, our approach learns to solve geometry problems by interpreting natural language demonstrations of the solution. These demonstrations illustrate the process of solving the geometry problem via step-wise application of geometry theorems. This part of the thesis is taken from our past work published as [237].

Demonstration as Sequence of Program Applications

The demonstration can be seen as a program – a sequence of horn clause rule applications that lead to the solution of the geometry problem. Given a current state, theorem t can be applied to the state if there exists an assignment to free variables in $l_t^{(pr)}$ that is true in the state. Each theorem application also has a probability associated with it; in our case, these probabilities are learned by a trained model. The state diagram for the demonstration in Figure 7.8 is shown in Figure 7.9. Now, we describe the various components of our *learning from demonstrations* approach which involves the use of the *semantic parser* (described earlier) to interpret the demonstration and a *deductive solver* that learns to chain theorems.

State and Axiom Identification

Given a demonstrative solution of a geometry problem in natural language such as the one shown in Figure 7.8, we identify theorem applications by two simple heuristics. Often, theorem mentions in demonstrations collected from textbooks are labeled as references to theorems previously introduced in the textbook (for example, “Theorem 3.1”). In this case, we simply label the theorem application as the referenced theorem. Sometimes, the theorems are mentioned verbosely in the demonstration. To identify these mentions, we collect a set of theorem mentions from textbooks. Each theorem is also represented as a set of theorem mentions. Then, we use an off-the-shelf semantic text similarity system [282] and check if a contiguous sequence of sentences in the demonstration is a paraphrase of any of the gold theorem mentions. If the degree of similarity of a contiguous sequence of sentences in the demonstration with any of the gold theorem mentions is above a threshold, our system labels the sequence of sentences as the theorem. The text similarity system is tuned on the training dataset and the threshold is tuned on the development set. This heuristic works well and has a small error ($< 10\%$) on our development set.

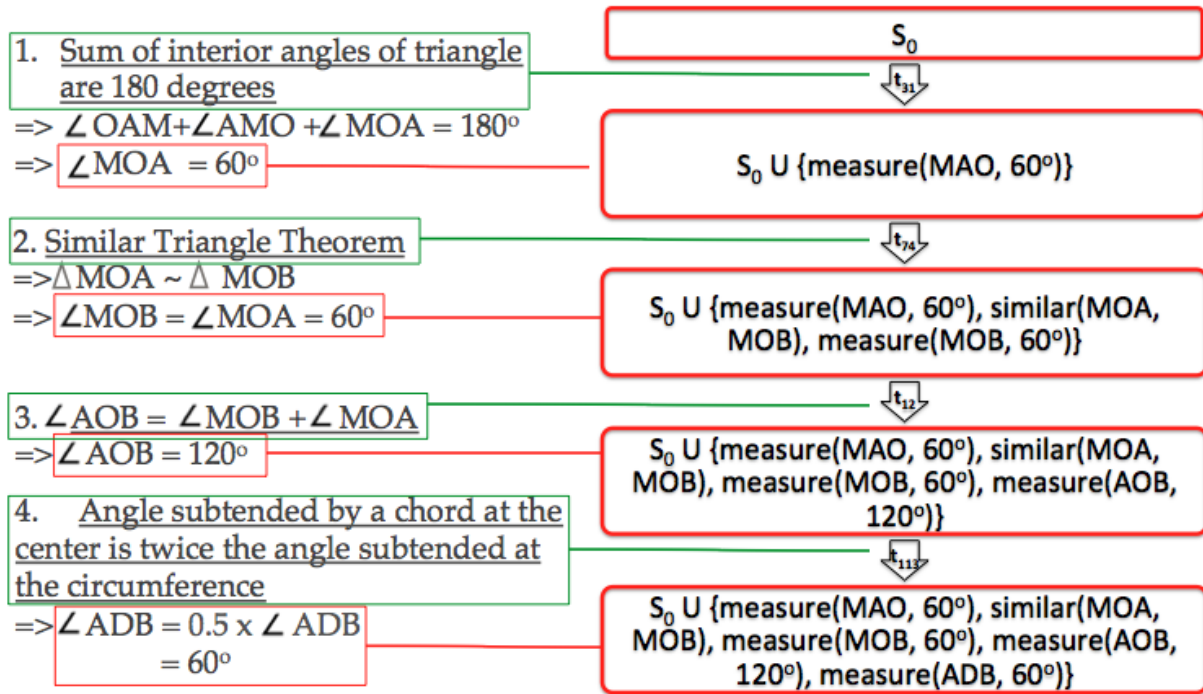


Figure 7.9: State sequence corresponding to the demonstration in Figure 2. Theorems applied are marked in green and the state information is marked in red. Here S_0 corresponds to the state derived from question interpretation and each theorem application subsequently adds new predicates to the logical formula corresponding to S_0 . The final state contains the answer: $\text{measure}(\text{ADB}, 60^\circ)$. This annotation of states and theorem applications is provided only for illustrative purposes. It is not required by our model.

For state identification, we use our semantic parser. The initial state corresponds to the logical expression corresponding to the question. Subsequent states are derived by parsing sentences in the demonstration. The identified state sequences are used to train our deductive solver.

Deductive Solver

Our deductive solver, inspired from [152], uses the parsed state and axiom information (when provided) and learns to score the sequence of axiom applications which can lead to the solution of the problem. Our solver uses a log-linear model over the space of possible axiom applications. Given a set of theorems \mathcal{T} and optionally demonstration d , we assume $\mathbf{T} = [t_1, t_2, \dots, t_k]$ to be a sequence of theorem applications. Each theorem application leads to a change in state. Let s_0 be the initial state determined by the logical formula derived from the question text and the diagram. Let $\mathbf{s} = [s_1, s_2, \dots, s_k]$ be the sequence of program states after corresponding theorem applications. The final state s_k contains the answer to the question. We define the model score of the deduction as:

$$P(\mathbf{s}|\mathbf{T}, d; \boldsymbol{\theta}_{ex}) = \frac{1}{Z(\mathbf{T}, d; \boldsymbol{\theta}_{ex})} \prod_{i=1}^k \exp(\boldsymbol{\theta}_{ex}^T \boldsymbol{\psi}(s_{i-1}, s_i, t_i, d))$$

Here, $\boldsymbol{\theta}_{ex}$ represents the model parameters and $\boldsymbol{\psi}$ represents the feature vector that depends on the successive states s_{i-1} and s_i , the demonstration d and the corresponding theorem application t_i . We find optimal parameters $\boldsymbol{\theta}_{ex}$ using maximum-likelihood estimation with L2 regularization:

$$\boldsymbol{\theta}_{ex}^* = \arg \max_{\boldsymbol{\theta}_{ex}} \sum_{\mathbf{s} \in \mathcal{T}_{train}} \log P(\mathbf{s}|\mathbf{T}, d; \boldsymbol{\theta}_{ex}) - \mu \|\boldsymbol{\theta}_{ex}\|_2^2$$

We use beam search for inference and L-BFGS to optimize the objective.

Joint Semantic Parsing and Deduction

Finally, we describe a joint model for semantic parsing and problem solving that parses the geometry problem text, the demonstration when available, and learns a sequence of theorem applications that can solve the problem.

In this case, we use a joint log-linear model for semantic parsing and deduction. The model comprises of factors that scores semantic parses of the question and the demonstration (when provided) and the other that scores various possible theorem applications. The model predicts the answer a given the question q (and possibly demonstration d) using two latent variables: \mathbf{p} represents the latent semantic parse of the question and the demonstration which involves identifying the logical formula for the question (and for every state in the demonstration when provided) and \mathbf{s} represents the (possibly latent) program.

$$P(\mathbf{p}, \mathbf{s} | q, a, d; \boldsymbol{\theta}) \propto f_p(p | \{q, a, d\}; \boldsymbol{\theta}_p) \times f_s(\mathbf{s} | \mathbf{T}, d, ; \boldsymbol{\theta}_s)$$

Here, $\boldsymbol{\theta} = \{\boldsymbol{\theta}_p, \boldsymbol{\theta}_{ex}\}$. f_p and f_s represent the factors for semantic parsing and deduction. $f_p(p | \{q, a, d\}; \boldsymbol{\theta}_p) \propto \exp(\boldsymbol{\theta}_p^T \boldsymbol{\phi}(p, \{q, a, d\}))$ was the semantic parsing model defined earlier and $f_s(\mathbf{s} | \mathbf{T}, d, ; \boldsymbol{\theta}_s) \propto \prod_{i=1}^k \exp(\boldsymbol{\theta}_{ex}^T \boldsymbol{\psi}(s_{i-1}, s_i, t_i, d))$ was defined in section 7.4.1. Next, we describe approaches to learn the joint model with various kinds of supervision.

Learning from Types of Supervision

Our joint model for parsing and deduction can be learned using various kinds of supervision. We provide a learning algorithm when (a) we only have geometry question-answer pairs as supervision, (b) when we have geometry questions and demonstrations for solving them, and (c) mixed supervision: when we have a set of geometry question-answer pairs in addition to some geometry questions and demonstrations. To do this, we implement two supervision schemes [152]. The first supervision scheme only verifies the answer and treats other states in the supervision as latent. The second scheme verifies every state in the program. We combine both kinds of supervision when provided. Given supervision $\{q_i, a_i\}_{i=1}^n$ and $\{q_i, a_i, d_i\}_{i=1}^m$, we define the following L2 regularized objective:

$$\begin{aligned} \mathcal{J}(\boldsymbol{\theta}) = & \nu \sum_{i=1}^n \log \sum_{\mathbf{p}, \mathbf{s}} P(\mathbf{p}, \mathbf{s} | q_i, a_i; \boldsymbol{\theta}) \times \mathbb{1}_{exec(\mathbf{s})=a_i} \\ & + (1 - \nu) \sum_{i=1}^m \log \sum_{\mathbf{p}, \mathbf{s}} P(\mathbf{p}, \mathbf{s} | q_i, a_i, d_i; \boldsymbol{\theta}) \times \mathbb{1}_{\mathbf{s}(\mathbf{d}_i)=\mathbf{s}} \\ & - \lambda \|\boldsymbol{\theta}_p\|_2^2 - \mu \|\boldsymbol{\theta}_{ex}\|_2^2 \end{aligned}$$

For learning from answers, we set $\nu = 1$. For learning from demonstrations, we set $\nu = 0$. We tune hyperparameters λ , μ and ν on a held out dev set. We use L-BFGS, using beam search for inference for training all our models. To avoid repeated usage of unnecessary theorems in the solution, we constrain the next theorem application to be distinct from previous theorem applications during beam search.

Features

Next, we define our feature set: ϕ_1, ϕ_2 for learning the semantic parser and ψ for learning the deduction model. Semantic parser features ϕ_1 and ϕ_2 are inspired from *GEOS*. The deduction model features ψ score consecutive states in the deduction s_{i-1}, s_i and the theorem t_i which when applied to s_{i-1} leads to s_i . ψ comprises of features that score if theorem t_i is applicable on state s_{i-1} and if the application of t_i on state s_{i-1} leads to state s_i . Table 7.12 lists the feature set.

Experiments

We use three geometry question datasets for evaluating our system: practice and official SAT style geometry questions used in *GEOS*, and an additional dataset of geometry questions collected from the aforementioned textbooks. We selected a total of 1406 SAT style questions across grades 6-10. This dataset is approximately 7.5 times the size of the datasets used in [253]. We split the dataset into training (350 questions), development (150 questions) and test (906 questions) with equal proportion of grade 6-10 questions. We also annotated the training and development set questions with ground-truth logical forms. *GEOS* used 13 types of entities, 94 functions and predicates. We added some more entities, functions and predicates to cover other more complex concepts in geometry not covered in *GEOS*. Thus, we obtained a final set of 19 entity types and 115 functions and predicates. We use the training set to train our semantic parser

ϕ_1	Lexicon Map	Indicator that the word or phrase maps to a predicate in a lexicon created in <i>GEOS</i> . <i>GEOS</i> derives correspondences between words/phrases and geometry keywords and concepts in the geometry language using manual annotations in its training data. For instance, the lexicon contains (“square”, <i>square</i> , <i>IsSquare</i>) including all possible concepts for the phrase “square”.
	Regex for numbers and explicit variables	Indicator that the word or phrase satisfies a regular expression to detect numbers or explicit variables (e.g. “5”, “AB”, “O”). These regular expressions were built as a part of <i>GEOS</i> .
ϕ_2	Dependency tree distance	Shortest distance between the words of the concept nodes in the dependency tree. We use indicator features for distances of -3 to 3. Positive distance shows if the child word is at the right of the parent’s in the sentence, and negative otherwise.
	Word distance	Distance between the words of the concept nodes in the sentence.
	Dependency edge	Indicator functions for outgoing edges of the parent and child for the shortest path between them.
	Part of speech tag	Indicator functions for the POS tags of the parent and the child
	Relation type	Indicator functions for unary / binary parent and child nodes.
	Return type	Indicator functions for the return types of the parent and the child nodes. For example, return type of <i>Equals</i> is boolean, and that of <i>LengthOf</i> is numeric.
ψ	State and theorem premise predicates	Treat the state s_{i-1} and theorem premise $l_i^{(pr)}$ as multi-sets of predicates. The feature is given by $div(s_{i-1} l_i^{(pr)})$, the divergence between the two multi-sets. $div(A, B)$, the divergence between multi-sets A and B is given by $\sum_k \frac{\min(A_k, B_k)}{B_k}$ which measures the degree to which the elements in A satisfy the pre-condition in B .
	State and theorem premise predicate-arguments	Now treat the state s_{i-1} and theorem premise $l_i^{(pr)}$ as two multi-sets over predicate-arguments. The feature is given by $div(s_{i-1} l_i^{(pr)})$, the divergence between the two multi-sets.
	State and theorem conclusion predicates	Now treat the state s_i and theorem conclusion $l_i^{(co)}$ as two multi-sets over predicate-arguments. The feature is given by $div(s_i l_i^{(co)})$, the divergence between the two multi-sets.
	State and theorem conclusion predicate-arguments	Now treat the state s_i and theorem conclusion $l_i^{(co)}$ as two multi-sets over predicate-arguments. The feature is given by $div(s_i l_i^{(co)})$, the divergence between the two multi-sets.
	State and theorem conclusion predicates	Treat the state s_i and theorem conclusion $l_i^{(co)}$ as two distributions over predicates. The feature is the total variation distance between the two distributions.
	State and theorem conclusion predicate-arguments	Now treat the state e_i and theorem conclusion $l_i^{(co)}$ as two distributions over predicate-arguments. The feature is the total variation distance between the two distributions.
	Product Features	We additionally use three product features: $\psi_1\psi_3\psi_5$, $\psi_2\psi_4\psi_6$ and $\psi_1\psi_2\psi_3\psi_4\psi_5\psi_6$

Table 7.12: The feature set for our joint semantic-parsing and deduction model. Features ϕ_1 and ϕ_2 are motivated from *GEOS*

	P	O	T
GEOS	61	49	32
GEOS++	62	49	44
O.S. (QA Pairs)	63	52	47
O.S. (Demonstrations)	66	55	56
O.S. (QA + Demonstrations)	67	57	58

Table 7.13: Scores of various approaches on the SAT practice (P) and official (O) datasets and a dataset of questions from the 20 textbooks (T). We use SAT’s grading scheme that rewards a correct answer with a score of 1.0 and penalizes a wrong answer with a negative score of 0.25. O.S. represents our system trained on question-answer (QA) pairs, demonstrations, or a combination of QA pairs and demonstrations.

with expanded set of entity types, functions and predicates. We used Stanford CoreNLP [186] for linguistic pre-processing. We also adapted the *GEOS* solver to the expanded set of entities, functions and predicates for comparison purposes. We call this system *GEOS++*.

Quantitative Results

We evaluated our joint model of semantic parsing and deduction with various settings for training: training on question-answer pairs or demonstrations alone, or with a combination of question-answer pairs and demonstrations. We compare our joint semantic parsing and deduction models against *GEOS* and *GEOS++*.

In the first setting, we only use question-answer pairs as supervision. We compare our semantic parsing and deduction model to *GEOS* and *GEOS++* on practice and official SAT style geometry questions from [253] as well as the dataset of geometry questions collected from the 20 textbooks (see Table 7.24). On all the three datasets, our system outperforms *GEOS* and *GEOS++*. Especially on the dataset from the 20 textbooks (which is a harder dataset and includes more problems which require complex reasoning supported by our deduction model), *GEOS* and *GEOS++* do not perform very well whereas our system achieves a very good score.

Next, we only use demonstrations to train our joint model (see Table 7.24). We test this model on the aforementioned datasets and compare it to *GEOS* and *GEOS++* trained on respective datasets. Again, our system outperforms *GEOS* and *GEOS++* on all three datasets. Especially on the textbook dataset, this model trained on demonstrations has significant improvements as our semantic parsing and deduction model trains the deduction model as well and learns to reason about geometry using axiomatic knowledge.

Finally, we train our semantic parsing and deduction model on a combination of question answer-pairs and demonstrations. This model trained on question-answer pairs and demonstrations leads to further improvements over models trained only question-answer pairs or only on demonstrations. These results (shown in Table 7.24) hold on all the three datasets.

We tested the correctness of the parses and the deductive programs induced by our models. First, we compared the parses induced by our models with gold parses on the development set. Table 7.23 reports the Precision, Recall and F1 scores of the parses induced by our models when only the parsing model or when the joint model is used and compares it with *GEOS*. We conclude that both our models perform better as compared to *GEOS* in parsing. Furthermore, our joint

	P	R	F1
GEOS	0.82	0.63	0.71
O.S. (Parser)	0.88	0.75	0.81
O.S. (Joint)	0.89	0.80	0.84

Table 7.14: Precision, Recall and F1 scores of the parses induced by *GEOS* and our models when only the parsing model or the joint model is used.

	Deduction	Joint
QA Pairs	0.56	0.61
Demonstrations	0.64	0.68
QA + Demonstrations	0.68	0.70

Table 7.15: Accuracy of the programs induced by various versions of our joint model trained on question-answer pairs, demonstrations or a combination of the two. We provide results when we use the deduction model or the joint model.

model of parsing and deduction further improves the parsing accuracy. Then, we compared the programs induced by the aforementioned models with gold program annotations on the textbook dataset. Table 7.15 reports the accuracy of programs induced by various versions of our models. Our models when trained on demonstrations induces more accurate programs as compared to the semantic parsing and deduction model when trained on question-answer pairs. Moreover, the semantic parsing and deduction model when trained on question-answer pairs as well as demonstrations achieves an even better accuracy. Our joint model of parsing and deduction induces more accurate programs as compared to the deduction model alone.

User Study on Interpretability

A key benefit of our axiomatic solver is that it provides an easy-to-understand student-friendly demonstrative solution to geometry problems. This is important because students typically learn geometry by rigorous deduction whereas numerical solvers do not provide such interpretability.

To test the interpretability of our axiomatic solver, we asked 50 grade 6-10 students (10 students in each grade) to use *GEOS++* and our best performing system trained on question-answer pairs and demonstrations as a web-based assistive tool. They were each asked to rate how ‘interpretable’ and ‘useful’ the two systems were for their studies on a scale of 1-5. Table 7.16 shows the mean rating by students in each grade on the two facets. We can observe that students of each grade found our system to be more interpretable as well as more useful to them than *GEOS++*. This study supports the need and the efficacy of an interpretable solution for geometry problems. Our solution can be used as an assistive tool for helping students learn geometry on MOOCs.

	Interpretability		Usefulness	
	<i>GEOS++</i>	<i>O.S.</i>	<i>GEOS++</i>	<i>O.S.</i>
Grade 6	2.7	3.0	2.9	3.2
Grade 7	3.0	3.7	3.3	3.6
Grade 8	2.7	3.6	3.1	3.5
Grade 9	2.4	3.4	3.0	3.6
Grade 10	2.8	3.1	3.2	3.7
Overall	2.7	3.4	3.1	3.5

Table 7.16: User study ratings for *GEOS++* and our system (*O.S.*) trained on question-answer pairs and demonstrations by a number of grade 6-10 student subjects. Ten students in each grade were asked to rate the two systems on a scale of 1-5 on two facets: ‘interpretability’ and ‘usefulness’. Each cell shows the mean rating computed over ten students in that grade for that facet.

		Axiom Identification F1		SAT Scores		
		Strict Comp.	Relaxed Comp.	Practice	Official	Textbook
Content	Sentence Overlap	56.2	73.8	56	43	42
	Geometry entities	64.0	80.4	61	49	46
	Keywords	67.5	81.0	62	54	48
Discourse (Typography)	RST edge	66.6	78.9	58	46	44
	Axm, Thm, Corr.	62.6	77.8	57	47	43
	Equation	66.2	78.6	57	46	42
	Associated Diagram	68.5	84.4	61	52	49
	Bold / Underline	68.2	82.0	62	52	48
	Bounding box	59.7	75.5	55	47	40
	XML structure	67.4	80.6	60	51	46
	Unablated	70.4	87.6	64	55	51

Table 7.17: Ablation study results for the axiom identification component. We remove features of the axiom identification component one by one as listed in Table 7.4 and observe the fall in performance in terms of the axiom identification performance as well as the overall performance to gauge the value of the various features.

7.4.2 Feature Ablation

In this section, we will measure the value of the various features in our axiom harvesting and parsing pipeline. Note that we have described three set of features **f**, **g** and **h** corresponding to the various steps in our pipeline: axiom identification, axiom alignment and axiom parsing in Tables 7.4, 7.5 and 7.6. We will ablate each of the three features one by one via *backward selection*, i.e. we will remove features and observe how that affects performance.

Ablating Axiom Identification Features

Table 7.17 shows the fall in performance in terms of the axiom identification performance as well as the overall performance as we ablate various axiom identification features listed in Table 7.4. We can observe that removal of any of the features results in a loss of performance. Thus, all

		SAT Scores				
		F1	NMI	Practice	Official	Textbook
Content	Overlap	70.7	0.54	57	45	45
	LCS	78.7	0.64	61	53	49
	Number of Sentences	78.5	0.65	62	54	48
	Alignment Scores	72.6	0.57	59	49	48
	MT Metrics	74.8	0.60	62	52	49
	Summarization Metrics	75.9	0.63	62	54	50
Typography	XML Structure	71.5	0.57	58	47	46
	Equation Template	76.6	0.61	57	47	43
	Image Caption	77.9	0.65	62	53	47
Unablated		80.3	0.69	64	55	51

Table 7.18: Ablation study results for the axiom alignment component. We remove features of the axiom alignment component one by one as listed in Table 7.5 and observe the fall in performance in terms of the axiom alignment performance as well as the overall performance to gauge the value of the various features.

the content as well as typographical features are important for performance. We observe that the content features such as sentence overlap, geometry entity sharing and keyword usage are clearly important. At the same time, the various discourse features such as the RST relation, axiom, theorem, corollary annotation, use of equations and diagrams, bold/underline, bounding box and XML structure are all important. Most of these features depend on typographical information which vital in performance of the axiom identification component as well as the overall model. In particular, we can observe that the axiom, theorem, corollary annotation and the bounding box features contribute most to the performance of the model as they are direct indicators of the presence of an axiom mention.

Ablating Axiom Alignment Features

Table 7.18 shows the fall in performance in terms of the axiom alignment performance as well as the overall performance as we ablate various axiom alignment features listed in Table 7.5. We again observe that removal of any of the features results in a loss of performance. Thus, the various content as well as typographical features are important for performance. We observe that the content features such as unigram, bigram and entity overlap, length of the longest common subsequence, number of sentences and various aligner, MT and summarization scores are clearly important. At the same time, the various discourse features such as the XML structure, equation template and image caption match are all important. Note that these features depend on typographical information which is again vital in performance. In particular, we can observe that the overlap and the XML structure features contribute most to the performance of the model.

Ablating Axiom Parsing Features

Table 7.19 shows the fall in performance in terms of the axiom parsing performance as well as the overall performance as we ablate various axiom parsing features listed in Table 7.6. We again

	F1		SAT Scores		
	Literals	Full Parse	Practice	Official	Textbook
Span Similarity	71.8	64.6	51	40	42
No. of Relations	82.3	70.5	60	51	49
Span Lengths	86.0	72.0	63	54	50
Relative Position	83.9	69.2	60	52	47
Discourse Markers	77.4	68.4	55	48	47
Punctuations	73.5	65.0	52	45	45
Text Organization	74.4	66.2	52	47	46
RST Parse	84.6	70.8	62	52	49
Soricut & Marcu	83.2	69.8	61	52	50
Head Node, etc.	85.3	71.6	62	54	49
Syntax	75.5	66.6	54	47	46
Dominance	73.9	66.1	53	47	44
XML Structure	77.6	68.0	59	51	46
Unablated	87.5	73.2	64	55	51

Table 7.19: Ablation study results for the axiom parsing component. We remove features of the axiom parsing component one by one as listed in Table 7.6 and observe the fall in performance in terms of the axiom parsing performance as well as the overall performance to gauge the value of the various features.

observe that removal of any of the features results in a loss of performance. The axiom parsing component uses few content based features such as span similarity and no. of relations, span lengths and relative position, and various discourse features such as discourse markers, punctuations, text organization, RST parse, an existing discourse segmentor from Soricut and Marcu [266], node attachment, syntax, dominance and XML structure, and all are clearly important. In particular, we can observe that span similarity and punctuation features contribute most to the performance of the model.

7.4.3 Axioms Harvested

We qualitatively analyze the structured axioms harvested by our method. We show few most probable horn clause rules for some popular named theorems in geometry in Figure 7.10 along with the confidence of our method on the rules being correct. Note that some horn clause parsed rules can be incorrect. For example, the second most probable horn clause rule for the Pythagoras theorem is partially incorrect (doesn't state which angle is 90°). Similarly, the second and third most probable horn clause for the circle secant tangent theorem are also incorrect. Our problog solver can use these redundant but weighted horn clause rules for solving geometry problems.

7.4.4 Example Solutions and Error Analysis

Next, we qualitatively describe some example solutions of geometry problems as well as perform a qualitative error analysis. We first show some sample questions which our solver can answer correctly in Table 7.20. We also show the explanations generated by our deductive solver for these problems (constructed in the same way as described earlier). Note that these problems are diverse in terms of question types as well as the reasoning required to answer them and our solver

Pythagorous Theorem:

0.84 isTriangle(ABC) ^ angleMeasure(ABC, 90) → length(AB)² + length(BC)² = length(CA)²

0.53 isTriangle(PQR) ^ isRightTriangle(PQR) ^ angleMeasure(PQR, 90) → length(PQ)² + length(QR)² = length(RP)²

Sum of angles of a Triangle

0.94 isTriangle(ABC) → angleMeasure(ABC) + angleMeasure(BCA) + angleMeasure(CAB) = 180

Circle Secant Tangent Theorem

0.77 isCircle(O) ^ isTangentAt(PT, O, T) ^ isSecantAt(PA, O, A) ^ isSecantAt(PB, O, B) ^ liesOn(A, PB) → length(PA) * length(PB) = length(PT)²

0.46 isCircle(O) ^ isTangent(PT, O) ^ isSecant(PA, O) ^ isSecant(PB, O) → length(PA) * length(PB) = length(PT)²

0.41 isCircle(O) ^ isTangentAt(PT, O, T) ^ isSecantAt(PA, O, A) ^ isSecantAt(PB, O, B) → length(PA) * length(PB) = length(PT)²

Congruent Triangles

0.82 isTriangle(ABC) ^ isTriangle(DEF) ^ length(AB) = length(DE) ^ length(BC) = length(EF) ^ length(CA) = length(FD) → congruentTriangles(ABC, DEF)

0.73 isTriangle(ABC) ^ isTriangle(DEF) ^ length(AB) = length(DE) ^ angleMeasure(ABC) = angleMeasure(DEF) ^ length(BC) = length(EF) → congruentTriangles(ABC, DEF)

0.76 isTriangle(ABC) ^ isTriangle(DEF) ^ angleMeasure(ABC) = angleMeasure(DEF) ^ length(BC) = length(EF) ^ angleMeasure(BCA) = angleMeasure(EFD) → congruentTriangles(ABC, DEF)

0.78 isTriangle(ABC) ^ isTriangle(DEF) ^ angleMeasure(ABC, 90) ^ angleMeasure(DEF, 90) ^ length(AB) = length(DE) ^ length(CA) = length(FD) → congruentTriangles(ABC, DEF)

Figure 7.10: Horn clause rules for some popular named theorems in geometry harvested by our approach. We also show the confidence our method has on the rule being correct (which is used in reasoning via the problog solver).

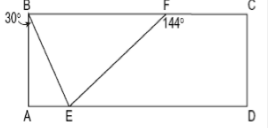
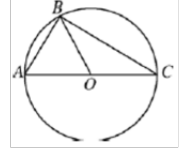
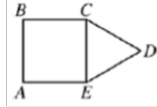
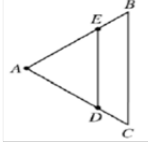
<div data-bbox="324 294 609 514" style="border: 1px solid black; padding: 5px;">  <p>In the accompanying diagram of rectangle ABCD, $\angle ABE = 30$ and $\angle CFE = 144$. Find $\angle BEF$.</p> </div> <p>1. Each angle of a rectangle is 90° $\Rightarrow \angle BAE = 90^\circ$</p> <p>2. Each angle of a rectangle is 90° $\Rightarrow \angle CDE = 90^\circ$</p> <p>3. Sum of internal angles of a triangle is 180° $\Rightarrow \angle BAE + \angle AEB + \angle EBA = 180^\circ$ $\Rightarrow 90^\circ + \angle AEB + 30^\circ = 180^\circ$ $\Rightarrow \angle AEB = 60^\circ$</p> <p>4. Sum of internal angles of a quadrilateral is 360° $\Rightarrow \angle CDE + \angle DEF + \angle EFC + \angle FCD = 180^\circ$ $\Rightarrow 90^\circ + \angle DEF + 144^\circ + 90^\circ = 360^\circ$ $\Rightarrow \angle DEF = 36^\circ$</p> <p>5. Angles on one side of a straight line is 180° $\Rightarrow \angle AEB + \angle BEF + \angle DEF = 180^\circ$ $\Rightarrow 60^\circ + \angle BEF + 36^\circ = 180^\circ$ $\Rightarrow \angle BEF = 84^\circ$</p>	<div data-bbox="917 441 1234 714" style="border: 1px solid black; padding: 5px;">  <p>In the figure above, triangle ABC is inscribed in the circle with center O and diameter AC. If $AB = AO$, what is the degree measure of angle ABO?</p> </div> <p>1. Circle Definition: The length of all line segments joining the center of a circle with any point that lies on the circle are equal, equal to the radius of the circle. $\Rightarrow AO = BO$</p> <p>2. A triangle with all equal sides is equilateral $\Rightarrow \triangle ABO$ is equilateral.</p> <p>3. Every internal angle of an equilateral triangle is 60° $\Rightarrow \angle ABO = 60^\circ$</p>
<div data-bbox="332 1102 617 1323" style="border: 1px solid black; padding: 5px;">  <p>In the figure above, CDE is an equilateral triangle and ABCE is a square with an area of 1. What is the perimeter of polygon ABCDE?</p> </div> <p>1. Side of a square is square root of the area. $\Rightarrow AB = BC = CE = EA = 1$</p> <p>2. All sides of an equilateral triangle are equal $\Rightarrow CD = DE = EA = 1$</p> <p>3. Perimeter of a polygon is a the sum of its sides $\Rightarrow \text{perimeter}(ABCDE) = AB + BC + CD + DE + EA$ $\Rightarrow \text{perimeter}(ABCDE) = 1 + 1 + 1 + 1 + 1$ $= 5$</p>	<div data-bbox="917 997 1242 1270" style="border: 1px solid black; padding: 5px;">  <p>In triangle ABC above, $AB = AC$, E is the midpoint of line AB, and D is the midpoint of line AC. If $AE = x$ and $ED = 4$, what is length BC?</p> </div> <p>1. Similar Triangles Note that $\frac{AD}{AC} = \frac{AE}{AB}$ and $\angle EAD = \angle BAC$ $\Rightarrow \triangle EAD \sim \triangle BAC$ (SAS similarity)</p> <p>2. Ratio of the lengths of corresponding sides of similar triangles are equal $\Rightarrow \frac{DE}{CB} = \frac{AD}{AC}$ $\Rightarrow \frac{DE}{CB} = \frac{1}{2}$ $\Rightarrow \frac{4}{CB} = \frac{1}{2}$ $\Rightarrow CB = 8$ $\Rightarrow BC = 8$</p>

Table 7.20: Some correctly answered questions along with explanations generated by our deductive solver for these problems.

can handle them.

We also show some failure cases of our approach in Table 7.21. There are a number of reasons that could lead to a failure of our approach to correctly answer a question. These include an error in parsing the diagram, the text, or an incorrect or incomplete knowledge in the form of geometry rules. As can be observed in the failure examples, and also evaluated by us in a small error analysis of 100 textbook questions, our approach answered 52 questions correctly. Amongst the 48 incorrectly answered questions, our diagram parse was incorrect for 12 questions, and the text parse was incorrect for 15 questions. Our formal language was insufficiently defined to handle 6 questions, i.e. the semantics of the question could not be adequately captured by the formal language. 21 questions were incorrectly answered due to missing knowledge of geometry in the form of rules. Note that several questions were incorrectly answered due to a failure of multiple system components (for example, failure of both the text and the diagram parser).

7.5 Answering Newtonian Physics Questions

We propose to apply the *Parsing to Programs* approach to the task of answering Newtonian Physics problems. An illustration of the same is given in Figure . Newtonian Physics forms a key component in the Physics curricula of a pre-university student. We collect a large dataset of question-answer pairs from physics textbooks widely used by students in India. We evaluate our trained systems on a held out dataset of questions from these textbooks and on practice and actual questions from the AP physics C mechanics exam. Our system currently achieves an overall accuracy of 68% on held-out questions from textbooks. In contrast, a small user study conducted by us on 10 students studying physics found that the average student score was only around 63%. On the AP Physics C mechanics practice and actual exams (1998 and 2012), our system correctly answered 50%, 42% and 54% of the questions, respectively. These scores are close to the average human performance on these exams.

The text parsing pipeline is the same as the one for solving geometry problems. We describe the pipeline for diagram parsing below:

7.5.1 Diagram Parsing Pipeline

Pre-university physics questions are often accompanied by diagrams. For example, in our training dataset described in section 7.5.4, about 20 percent of the questions are accompanied by diagrams. These diagrams are pretty complex and diverse. The diagrams typically have a large number of object categories and depict complex higher-order physical phenomena related to these objects (such as force, acceleration, velocity, etc.) which goes well beyond what natural images usually convey. The diagrams typically include object elements (e.g. drawings of blocks, wedges, etc.), textual elements (e.g. annotations of forces, acceleration, velocity, etc.), low-level diagrammatic elements (e.g. arrows, lines, etc.), high-level diagrammatic elements (e.g. axes, blocks, pulleys, etc.), plots and possibly other decorative elements. All these elements of the diagram must be recognized and organized in context to achieve a complete logical representation of the diagram. Next, we describe our pipeline approach for recognizing various diagram elements in diagrams.

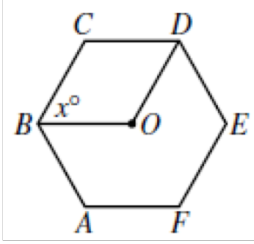
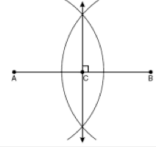
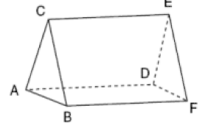
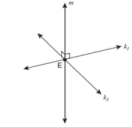
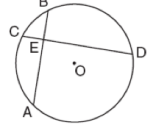

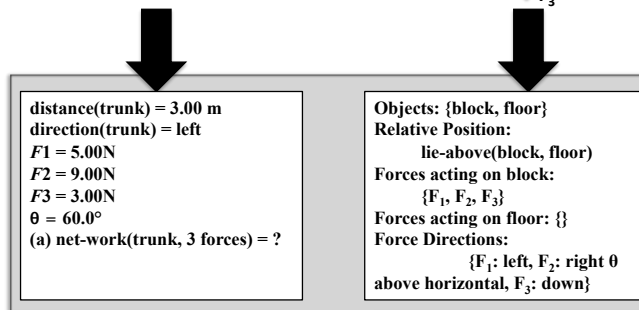
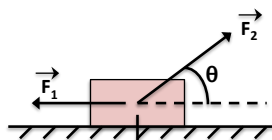
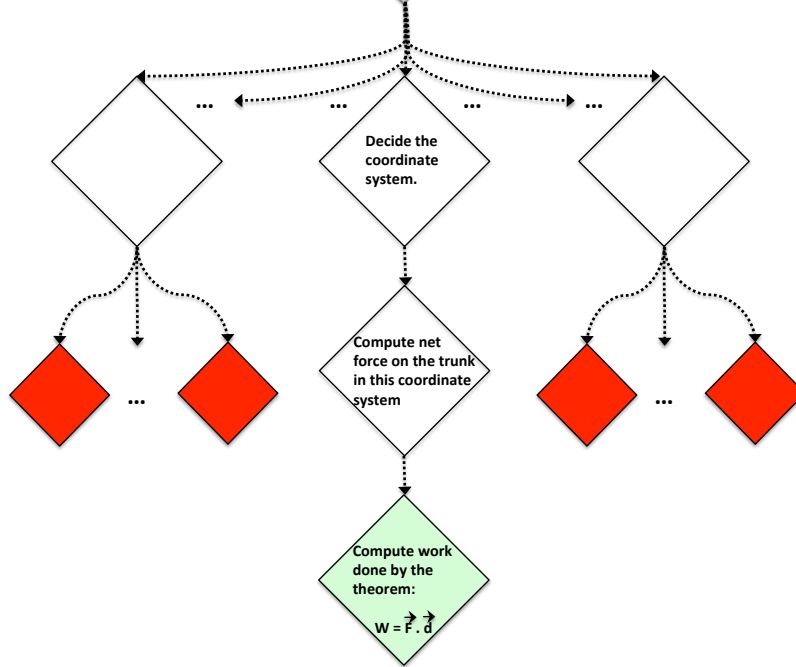
 <p>In the figure above, ABCDEF is a regular hexagon, and its center is point O. What is the value of x?</p>	 <p>The diagram at the right shows the construction of the perpendicular bisector of AB. Which statement is not true?</p> <p>AC = CB AC = 2AB CB = 1/2 AB AC + CB = AB</p>
 <p>The figure in the diagram at the right is a triangular prism. Which statement must be true?</p> <p>line DE = line AB line AD = line CE line AD = line BC line De = line BC</p>	 <p>Lines k_1 and k_2 intersect at point E. Line m is perpendicular to lines k_1 and k_2 at point E. Which statement is always true?</p> <p>Lines k1 and k2 are perpendicular. Line m is parallel to the plane determined by lines k_1 and k_2. Line m is perpendicular to the plane determined by lines k_1 and k_2. Line m is coplanar with lines k_1 and k_2.</p>
 <p>In the accompanying diagram of circle O, chords AB and CD intersect at E and angle AC : CB : BD : DA = 4 : 2 : 6 : 8. What is the angle DEB?</p> <p>36 degrees 90 degrees 100 degrees 126 degrees</p>	 <p>A cathedral window is built in the shape of a semicircle. If the window is to contain three stained glass sections of equal size, what is the area of each stained glass section? Express answer to the nearest square foot.</p> <p>1 sq. ft. 3 sq. ft. 13 sq. ft. 26 sq. ft.</p>

Table 7.21: Some example failure cases of our approach for solving SAT style geometry problems. In (i) the axiom set contains an axiom that internal angle of a regular hexagon is 120° and that each side of a regular polygon is equal. But there's no way to deduce that the angle CBO is half of the internal angle ABC (by symmetry). The other hand, the coordinate geometry solver can exploit these three facts as maximizing the satisfiability of the various constraints can answer the question. (ii) The solver does not contain any knowledge about construction. The question cannot be correctly interpreted and the coordinate geometry solver also gets it wrong. (iii) The solver does not contain any knowledge about construction or prisms. The question cannot be correctly interpreted and the coordinate geometry solver also gets it wrong. (iv) The question as well as the answer candidates cannot be correctly interpreted (as the concept of perpendicular to plane is not in the vocabulary). Both solvers get it wrong. (v) The parser cannot interpret that angle AC is indeed angle AEC. This needs to be understood by context as it defies the standard type definition of an angle. Both solvers get it wrong. (vi) Both diagram and text parsers fail here. Both solvers answer incorrectly.

Figure 7-27 shows three forces applied to a trunk that moves leftward by 3.00 m over a frictionless floor. The force magnitudes are $F_1 = 5.00\text{N}$, $F_2 = 9.00\text{N}$, and $F_3 = 3.00\text{N}$, and the indicated angle is $\theta = 60.0^\circ$. During the displacement, what is the net work done on the trunk by the three forces?



Question Parsing



Programmatic Solving

Figure 7.11: An illustrative representation of our approach on a sample question from the Newtonian Physics dataset. The approach solves the question in two stages. The first stage, *Question Parsing*, parses the question text and any associated diagram into an equivalent (weighted) logical expression in a typed first-order logic language. The logical expression for this example is shown in the two rectangular white boxes. Ideally, each literal in this expression is weighted. However, the weights are not shown in this figure for simplicity. The second stage, *Programmatic Solving* takes this formal representation of the question and solves it by performing a (probabilistic) search over a set of pre-defined programs. One of the paths in the search (which corresponds to the process required to solve the question) leads to the solution (shown in green), whereas others do not lead to any solution and are rejected (shown in red).

Detecting Low-level Diagrammatic Elements: Our physics diagrams have a number of diagrammatic elements such as arrows, lines, arcs, etc. We detect diagrammatic elements in three steps: In the first step, we apply a weak Gaussian blur on the raw image and then binarize it using a threshold selection method proposed in [209]. Then, we over-generate a large number of candidate diagrammatic element proposals using the boundary detection and grouping method from [150], Hough transforms [71] and by detecting parallel curved edge segments in a canny edge map. In the third step, we recursively merge proposals that exhibit a low residual when fit to a 1st or a 2nd degree polynomial. We then train a 2 class CNN resembling VGG-16 [263], with a fourth channel appended to the standard 3 channel RGB input. This fourth channel specifies the location of the diagrammatic element smoothed with a Gaussian kernel of width 5.

Detecting High-level Diagrammatic Elements: Sometimes, we may have to further assemble these low level diagrammatic elements (such as lines and arcs) to higher level diagrammatic elements (such as axes, blocks, wedges, pulleys, etc.) For this, we use Harris corner detectors [110] to identify possible locations of intersections of lines. Finally, we again merge lines that form higher level diagrammatic elements using combinatorial grouping [219] with a set of manually curated grouping rules for grouping each diagrammatic element.

Detecting Textual Elements: The problem of recognizing text in images is an old one and has led to an entire research area of optical character recognition (OCR) [126]. Most existing OCR systems are usually trained on character images, scanned documents, scenes and hence, do not work well in our the setting of diagrams from textbooks. To detect text labellings in diagrams, we use an off-the-shelf OCR system – *Tesseract*⁹ as a baseline model. However, since many textual elements are heavily structured (these include elements in vector notation (e.g. \vec{F}), greek alphabets (e.g. θ), physical quantities (e.g. 2 m/s)) and are usually longer than a single character, we improve the detection by training a text localizer using a CNN having the same architecture as AlexNet [153]. We used the Chars74K dataset [65], a dataset obtained from vector PDFs of a number of physics textbooks and a set of synthetic renderings of structured textual elements generated by us as training data.

Detecting Label associations: Often, diagram labels such as \vec{F}_1 , \vec{F}_2 , \vec{F}_3 refer to low-level diagrammatic elements such as the three arrows in Figure 7.1b. As a final step in diagrammatic element recognition, we map textual element labels with diagrammatic elements. Given the set of textual elements and the set of diagrammatic elements, we solve a label association problem that maps textual elements with diagrammatic elements. We first build a binary logistic regression classifier over pairs of textual elements and low-level diagrammatic elements. The probability that a textual element t_i maps to a low-level diagrammatic element d_j is given by the standard logistic form: $p_{ij} = P(y_{ij}|t_i, d_j; \mathbf{w}) = \left(1 + e^{-\mathbf{w} \cdot \mathbf{f}(t_i, d_j)}\right)^{-1}$. Finally, the textual element t_i is mapped to a low-level diagrammatic element \hat{d}_j by solving the classic bipartite graph matching ILP: $\max \sum_{i,j} p_{ij} y_{ij}$ s.t. $\sum_j y_{ij} = 1 \quad \forall i$. We use a small set of hand engineered features based on distance, shape, size and orientation of elements and labels.

Foreground Detection: Foreground detection [74] is an old computer vision technique which extracts the foreground of an image for further processing. For foreground detection, we classify every pixel in the diagram as a foreground vs background pixel. To achieve this, we

⁹<https://github.com/tesseract-ocr/tesseract>

build nonparametric kernel density estimates [214, 230] in RGB, texture and entropy spaces. We use these estimates as features for object detection.

Detecting Blobs and Objects: Blob detection [177] is another classical computer vision problem of detecting regions or blobs in an image in which some properties such as brightness or color are constant compared to surrounding regions. We tried several object proposal approaches in the literature like window classification [280], perceptual grouping [35, 105], cascaded ranking svm [316], objectness [3], selective search [277], global and local search [225] and edge boxes [319]. However, as all these approaches were developed for object proposal generation for natural images, we found that these did not work well for physics diagrams.

To tackle this, we used the foreground probability map described above to detect a set of blob segments produced by a set of classifiers with features capturing location, size, central and Hu moments, etc. These segment proposals were then combined with multi-scale combinatorial grouping [219], a recent grouping strategy, to achieve a ranked list of object proposals with corresponding confidence score from the model. Next, these object proposal scores are combined (by learning a linear interpolant on the training set) with the scores from a discriminatively trained part-based model [85] trained on physics questions for object detection that focus on the detection of manually selected list of objects commonly seen in physics diagrams (blocks, pulleys, etc.) to achieve a ranked list of object proposals and corresponding confidence scores.

Incorporating Question Text for Object Detection: Often, the corresponding question texts provide important cues for detecting these visual elements. For example, the question in Figure 7.2 mentions the object ‘trunk’. While it is unlikely that the object recognition component will correctly recognize the object ‘trunk’ as ‘trunk’ doesn’t appear again in the training dataset as an object, the mention of the noun phrase ‘trunk’ in the question text and the context in which it appears is a very important cue for this object being ‘trunk’. Hence, we build a text based object detector that uses logistic regression to classify each noun phrase in the question text as an object or not. We use a small set of manually engineered features for the prediction problem: (a) if the noun phrase is included in a list of objects manually build by us by looking at the dev set, (b) if the noun phrase is an object category in ImageNet, and (c) if the noun phrase is the agent/patient (determined using the Turbo dependency parser¹⁰) of a small list of actions taking place in our dev set (e.g. pull, run, hit ...).

Mapping Blobs and Objects to Object Labels: Finally, our system maps an object category detected by our diagram based blob/object detector to an object category detected by the text based object detector. For this, we build a binary logistic regression classifier over pairs of object categories and set of blobs/objects. The probability that an object category c_i maps to a blob/object o_j is given by the standard logistic form: $p_{ij} = P(y_{ij}|c_i, o_j; \mathbf{w}) = \left(1 + e^{-\mathbf{w} \cdot \mathbf{f}(c_i, o_j)}\right)^{-1}$. Finally, the object category \hat{c}_i is mapped to the most probable blob/object o_j using the bipartite graph matching ILP described before. We use a small set of hand engineered features based on (a) cosine similarity of proposal label of the blob/object and the object category, (b) composition of diagrammatic and textual elements recognized in the image which overlap with the object and context words (we use a context of 2 words on the left and 5 words on the right), and (c) average image similarity (we use mutual information) between top 10 images retrieved on Google by querying for the object category and the blob/object.

¹⁰<http://www.cs.cmu.edu/ark/TurboParser/>

Detecting Plot elements: A significant proportion of diagrams are plots (graphs, charts, etc.). Hence, to solve questions with plots, we need to be able to extract information from the plots and associate the information with corresponding legend entries. These plots usually have high variations and often are accompanied with heavy clutter and deformation. We use FigureSeer [262] to parse the plots and generate a structured tabular representation of the plot information.

From Diagram Elements to Literals: Finally, we use all the aforementioned diagram element detectors along with corresponding detection scores to obtain the interpretation for the question. We use a set of rules – one for every predicate. The rules decide if the predicate holds for a set of diagram elements which are type consistent with the arguments of the predicate.

7.5.2 Domain Theory as Programs

Subject knowledge of Newtonian physics is a crucial component in our solver. We present the domain knowledge to the system in the form of structured programs. Some example programs are shown in Figure 7.12.

```

def vector_addition(Vectors vectors):
    result = zero_vector()
    for vector in vectors:
        result = result + vector
    return result

def angle_bw_vectors(Vector vec1, Vector vec2):
    return cos_inv(dot(vec1, vec2)/(norm(vec1)*norm(vec2)))

def project_vector(Vector vec, Direction theta):
    return (vec*cos(theta), vec*sin(theta))

def implicit_g_force(Mass m, Forces forces):
    if not forces.contains(("mg i + 0 j")):
        forces.append(("mg i + 0 j"))

def Newton_II_law(Mass m, Forces forces, Accelerations accs):
    net_force = vector_addition(forces)
    net_acceleration = vector_addition(accs)
    return Constraint(net_force = m * net_acceleration)

def conservation_of_momentum(Mass m1, Velocity v1_initial, Mass m2, Velocity v2_initial, Velocity v1_final, Velocity v2_final):
    preconditions = [external_force_on_system() == None]
    return Constraint(m1*v1_initial+m2*v2_initial = m1*v1_final+m2*v2_final)

```

Figure 7.12: Example programs used by our approach.

Some of these programs perform very basic functions such as vector addition, computing angle between vectors, unit conversion, etc. Others, however, perform more complex functions

– such as applying Newton’s laws of motion or conservation of momentum, etc. A number of axioms denote laws of physics as some mathematical expressions. For example, the Newton’s second law is expressed simply as $\vec{F}_{net} = m \times \vec{a}$. Here \vec{F}_{net} stands for the vector quantity representing the net force on a body. m stands for the mass of the body and \vec{a} stands for the acceleration of the body. These programs define a set of preconditions which must be satisfied for it to be executed. When the preconditions are satisfied, the programs define the mathematical expression as a constraint on the model. These constraints are then solved to obtain the answer. We wrote a total of 237 such programs. Let \mathcal{P} represent this set of programs. For any program $p \in \mathcal{P}$, let $p^{(pr)}$ denote the precondition required to execute the program. We use this set of programs to answer the physics problems via the following deductive solver.

7.5.3 The Deductive Solver

Given access to the domain theory in the form of programs, we solve the physics problem by searching for program applications that can lead to the problem solution. We use a forward chaining search procedure exploring various possible program applications. Algorithm 2 describes the procedure.

Algorithm 1: Forward Chaining approach for solving physics problems

Data: Weighted set of literals L representing the question and Domain knowledge \mathcal{P} .

1 Do

2 **1. Match Programs:** Match the pre-conditions of the programs against the set of literals i.e. find all programs $p \in \mathcal{P}$ s.t. the precondition p^{pr} can be unified with some set of literals L .

3 **2. Select Program:** Sample a program (randomly uniformly) among the matching programs. Stop if no program can be applied.

4 **3. Apply Program:** Apply the chosen program by adding the result to the set of literals/constraint set.

5 while $\#iterations < N_{upperbound}$;

We score the program applications as a function of the scores of various literals in the program’s precondition. The score of a literal is given by the confidence score from question parser. In case it is a derived literal, its score is given by the function value of the program application that derived the literal. We explored various scoring functions: minimum, arithmetic mean, geometric mean and harmonic mean of all literal scores. We found that taking the harmonic mean performed the best. Hence, we use harmonic mean of precondition literals as the scoring function in our system.

Finally, we used an off-the shelf library¹¹ to solve the model constraints introduced by the programs. Then, the following answering interface uses the search results to answer the question.

Answering Interface (Handling Various Question and Answer Types): The physics examinations in our datasets consists of a number of question and answer types. While a majority

¹¹<http://docs.sympy.org/dev/modules/solvers/solvers.html#sympy.solvers.solvers.solve>

<i>Edge Boxes</i>	<i>O.S. - Text</i>	<i>O.S.</i>
24.66	52.23	68.32

Table 7.22: Jaccard similarity between detected diagram elements and gold elements for Edge Boxes and two versions of our system’s (O.S.) diagram parser: our parser which does not use text information and our full parser.

of questions directly ask about a particular physical quantity, there are a substantial number of questions which do not fit in this paradigm. For example, there are some *which of these are not true, select the odd one out, match the following* questions. To handle a variety of questions, we build an answering interface. The interface calls the deductive solver described above and answers the question based on the type of the question or the kind of answer sought.

7.5.4 Experiments

Datasets We validate our system on two types of datasets: a dataset of physics questions taken from popular pre-university physics textbooks and few AP Physics C: Mechanics courses. We train our model on physics questions taken from three popular pre-university physics textbooks: *Resnick Halliday Walker*, *D. B. Singh* and *NCERT*. Millions of students in India study physics from these books every year and these books are available online. We manually marked chapters relevant for Newtonian physics in these books and then parsed them using Adobe Acrobat’s *pdf2xml* parser. This resulted into a dataset of 4941 questions, out of which 1019 had associated diagrams. We partitioned the dataset into a random split of 1000 training, 500 development and 3441 test questions. We also annotated ground truth logical forms for the training and development question texts and diagrams. These logical forms are used for training our system. We also evaluated our system on section 1 of three AP Physics C Mechanics tests¹². Section 1 of the AP Physics C Mechanics practice test comprised of 10 questions and the official tests for 1998 and 2012 comprised of 75 and 35 questions respectively.

Results: We now evaluate the question parsing models individually. For diagram parsing, we compute the Jaccard similarity between the diagram elements detected by various versions of our diagram parser and compare them to gold elements. We consider *Edge Boxes* [319] – the best performing prior computer vision technique explored by us, and two variants of our diagram parser: diagram parser excluding text information and our full diagram parser. Table 7.22 reports the results on the development set. Our diagram parser achieves a score of 68.32 which is much better than Edge Boxes. Prior computer vision techniques work on element and blob detection for natural images and do not port well to diagrams. However, our carefully engineered element detector works well. We also observe that mapping elements to element labels by incorporating text information contributes to an improvement in the score.

Next, we evaluate our text based parser. We compare the parses induced by our models with gold parses on the dev set. Table 7.23 reports Precision, Recall and F1 scores of the parses induced. For comparison purposes, we build a rule-based parser baseline. A similar baseline was proposed in [253] for comparing to their geometry solver. The baseline uses a set of man-

¹²The other sections of the tests are subjective which we leave as future work.

	P	R	F1
Rule-based	0.82	0.27	0.41
O.S.	0.63	0.75	0.68

Table 7.23: Precision, Recall and F1 scores of parses induced by our system’s text parser compared to a rule based parser.

	T	P	1998	2012
Humans	63	52	44	48
O.S.	68	50	42	54

Table 7.24: Scores of our system compared to average score by 10 students on our dataset from physics textbooks (T), AP Physics C Mechanics - Section 1 practice test (P) and official tests for two years: 1998 and 2012.

ually designed high-precision rules. Each rule compares the dependency tree of each sentence to pre-defined templates, and if a template pattern is matched, the rule outputs the relation corresponding to that template. Our text-based parser achieves a F1 score of 0.68, a significant improvement over the rule-based parser (0.41).

Finally, we report the performance of our overall system on the task of solving Newtonian physics problems. We performed a user study with 10 students¹³ who were each asked to solve questions from various datasets. For the AP Physics C exams, each student took the entire test. Whereas, for the textbook dataset, each student was asked to answer a random selection of 100 questions in the test split. We score the students as well as our model as the percentage of correctly answered questions. We compare the results of our system with the average score achieved by the students on the various datasets. On the textbook questions dataset, our system achieves a score of 68% which is better than the average student score of 63%. On all the three AP Physics exams as well, our system achieves close to the average student score, superseding it in the 2012 exam.

7.6 Key Insights and Learnings from P2P systems

Our approach for *parsing to programs* is very general, and in principle, can be used to answer a wide variety of standardized test problems given an accurate semantic parser and an accurate reasoner which has access to the domain knowledge. However, this is almost always hard as both the tasks of semantic parsing (i.e. mapping text and diagrams to accurate formal representations) and program knowledge extraction (i.e. extracting subject knowledge from textbooks) is very hard and must be performed with high accuracy to render this approach feasible. In our work, we had the following key observations and learnings which allowed the framework to work well for these hard problems:

- (1) We showed how we can leverage the structure of the problem and build a simple shallow

¹³All the students selected for the user study scored at least 4 (“well qualified to receive college credit”) or 5 (“extremely well qualified to earn college credit”) on the 2016 AP Physics C exam.

semantic parser which is trained as a pipeline process by incorporating existing machine learning models and software modules for parts of the problem. This lets us leverage unlabeled data and build an accurate semantic parser with minimal supervision.

(ii) We also showed that we can tackle the hard problem of extracting programmatic knowledge from textbooks by relying on rich discourse and formatting features in these textbooks. Textbooks are meant to effectively convey the content to students in as easy a way as possible, and in order to do so, they rely on a lot of rich formatting features. We show how we can use these formatting features as cues which aid information extraction.

(iii) Despite the above ideas which make accurate semantic parsing and program extraction work, we over generate the assertions (from our semantic parser) and the knowledge rules (from our program extractor) in our logical program. While this does result in some loss of interpretability, this is important to have a good performance on these problems.

(iv) In this chapter, we handled questions which were multimodal (they comprised of a piece of question text as well as a diagram). We noticed that the diagram and the question text provide cues for parsing and interpreting each other. Thus, we came up with a joint model for parsing the question text and the diagram where we used the question text to help us parse the diagram and we used the diagram to help us parse the question text by aligning diagram mentions to text mentions.

(v) The overall system comprises of a number of learning objectives and subproblems. Each of them are trained separately with different loss functions and objectives which are specific to the problem piece at hand. In the future, it would be interesting to consider jointly learning all these subproblems.

(vi) In this chapter, we took a symbolic approach to answering these reasoning problems. While these approaches can work well for these problems, we have a number of key issues which require fresh supervision and domain knowledge for solving such problems. In the future, we would like to tackle this issue by incorporating a combination of representation learning methods with symbolic learning. This would be an interesting area of future research.

(vii) A key motivation for the *parsing to programs* framework is its interpretability and explainability. In the future, we would like to explore how we can use the model explanations generated by such automatic problem solvers to teach students in an educational setting.

7.7 Conclusion

We described a framework called Parsing to Programs, combining ideas from parsing of natural language and diagrams with probabilistic programming for situated question answering. We used it to develop a system that can solve pre-university level Euclidean geometry and Newtonian physics problems. Our system achieved a performance close the student average on questions from various textbooks, geometry questions taken from previous SAT examinations and section 1 of Advanced Placement (AP) Physics C mechanics exams. As a key step in the parsing to programs framework, we developed methods to parse diagrams to rich meaning representations and also extract structured axiomatic knowledge from reading a number of textbooks. The formal meaning representations along with the extracted structured axiomatic knowledge was used to deeply reason about these problems and solve them.

Chapter 8

Joint Question Answering and Question Generation

So far in this thesis, we studied a number of novel question answering tasks which focuses on answering questions from standardized tests such as reading comprehensions, elementary science tests and advanced math and science examinations. All these approaches were data intensive and required significant amount of supervision in the form of question answer pairs. Some models such as the P2P framework also relied on annotations for meaning representations and annotations of axiomatic information in textbooks. All this supervision is expensive to obtain and forms a key bottleneck for scaling up these techniques to other tasks and domains. In order to make these techniques more scalable and robust, we must tackle the key issue of **supervision** and be able to build question answering models that require less supervision and annotation.

In this chapter, we explore self-training [318] as a way to bootstrap additional data and reduce the amount of supervision required to train question answering models. The self-training approach is based on a key observation that question answering (QA) and question generation or question asking (AQ) are closely related¹ tasks. And we can use this relationship between these two tasks as a way to carefully create additional training data that can further be used to improve the question answering and question generation models.

There has been a lot of prior work in both question answering and question generation. However, the literature on QA and AQ views the two as entirely separate tasks. In this paper, we explore this relationship among the two tasks by jointly learning to answer as well as ask questions. The close relationship between QA and AQ is useful for a number of reasons. – the most important being that the two can be used in conjunction to generate novel questions from free text and then answers for the generated questions. We can use this to perform self-training and leverage unlabeled text to augment training of QA and AQ models.

QA and AQ models are typically trained on question answer pairs which are often expensive to obtain in many domains. However, it is much cheaper to obtain large quantities of unlabeled text. Our self-training (or self-labeling) procedure leverages unlabeled text to boost the quality of our QA and AQ models. This is achieved by a careful data augmentation procedure which uses existing pre-trained QA and AQ models to generate additional labeled question answer pairs.

¹We can think of QA and AQ as inverse of each other.

This additional data is then used to retrain our QA and AQ models and the procedure is repeated.

The addition of synthetic labeled data needs to be performed carefully. During self-training, typically the most *confident* samples, along with their predicted labels, are added to the training set [318]. The performance of our QA model can be used as a proxy for computing the *confidence* value of the questions. In fact, we describe a suite of heuristics inspired from curriculum learning [17] to select the unlabeled samples (sentences) to be labeled and added to the training set at each epoch. Curriculum learning is inspired from the incremental nature of human learning and orders training samples on the *easiness* scale so that *easy* samples can be introduced to the learning algorithm first and *harder* samples can be introduced successively. We show that selecting training samples in increasing order of *easiness* leads to improvements over a random sample introduction baseline.

We focus on the non-factoid question answering task of machine comprehension² [224, 227] where the task is to answer a question \mathbf{q} based on a passage \mathbf{p} . We model the machine comprehension task as an answer sentence selection task i.e., given the set of sentences in the passage \mathbf{p} , the task is to select the sentence $\mathbf{s} \in \mathbf{p}$ that contains the answer \mathbf{a} . This treatment of QA as an answer sentence selection task is quite common in literature (e.g. see [308]). We model the AQ task as the task of transforming a sentence in the passage into a question. Previous literature has often treated the AQ task as one of transforming text sentences into questions (e.g. see [115]) via some set of manually engineered rules, etc. However, we take a completely end-to-end neural approach for the same.

Formally, let \mathcal{D} be a labeled dataset of (passage, question, answer) triples where the answer is given by selecting a sentence in the passage. We use this as a labeled dataset for training our QA as well as AQ model. Besides, we assume access to unlabeled text \mathcal{T} which will be used to augment the training of the two models.

8.1 The Question Answering Model

As described earlier, we model the QA task the task of selecting an answer sentence from the passage. Hence, for every input question, we treat each sentence in the corresponding passage as a candidate answer. We employ a neural network model inspired from the *Attentive Reader* framework proposed in [44, 117]. Note that this choice of QA model is arbitrary and we can replace it with any other successful model.

We first map all words in the vocabulary to corresponding d dimensional vector representations via an embedding matrix $E \in \mathbb{R}^{d \times V}$. Thus, the input passage \mathbf{p} can be denoted by the word sequence $\{p_1, p_2, \dots, p_{|\mathbf{p}|}\}$ and the question \mathbf{q} can similarly be denoted by the word sequence $\{q_1, q_2, \dots, q_{|\mathbf{q}|}\}$ where each token $p_i \in \mathbb{R}^d$ and $q_i \in \mathbb{R}^d$.

We use a bi-directional LSTM [103] with dropout regularization as in [310] to encode contextual embeddings of each word in the passage.

$$\vec{\mathbf{h}}_t = LSTM_1(p_t, \vec{\mathbf{h}}_{t-1}), \quad \overleftarrow{\mathbf{h}}_t = LSTM_2(p_t, \overleftarrow{\mathbf{h}}_{t+1})$$

²Although, we note that our approach is fairly general and can be extended to general QA settings.

The final contextual embeddings \mathbf{h}_t are given by concatenation of the forward and backward pass embeddings: $\mathbf{h}_t = [\vec{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$. Similarly, we use another bi-directional LSTM and encode contextual embeddings of each word in the question.

Then, we use attention mechanism [12] to compute the alignment distribution a based on the relevance among passage words and the question.

$$a_i = \text{softmax}(\mathbf{q}^T \mathbf{W} \mathbf{h}_i)$$

The output vector \mathbf{o} is a weighted combination of all contextual embeddings:

$$\mathbf{o} = \sum_i a_i \mathbf{h}_i$$

Finally, the correct answer a^* among the set of candidate answers \mathcal{A} is given by:

$$a^* = \arg \max_{a \in \mathcal{A}} \mathbf{w}^T \mathbf{o}$$

We learn this model by maximizing the log-likelihood of correct answers. Let $\{\mathbf{p}^{(i)}, \mathbf{q}^{(i)}, \mathbf{a}^{(i)}\}_{i=1}^N$ be the training set.

$$\mathcal{L}_{QA} = \sum_{i=1}^N \log P(\mathbf{a}^{(i)} | \mathbf{p}^{(i)}, \mathbf{q}^{(i)}; \theta)$$

Here, θ represents all model parameters to be estimated. We use beam search for inference.

8.2 The Question Asking Model

We use a sequence to sequence model (seq2seq) [273] with soft attention [12] as our question asking model. Sequence to sequence models have been shown to be successful in a number of NLP tasks such as machine translation [12], speech recognition [48], caption generation [279], dialog [278], etc. Infact, a similar seq2seq framework has also been used for question asking previously. See [70, 274] for examples.

We assume that our question asking model transduces an input sequence \mathbf{x} to an input sequence \mathbf{y} . Here, the input sequence is a sentence in the passage and the output sequence is a generated question. Let $\mathbf{x} = \{x_1, x_2, \dots, x_{|\mathbf{x}|}\}$, $\mathbf{y} = \{y_1, y_2, \dots, y_{|\mathbf{y}|}\}$ and \mathcal{Y} be the space of all possible output questions. Thus, we can represent the AQ task as finding $\hat{\mathbf{y}} \in \mathcal{Y}$ such that:

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} P(\mathbf{y} | \mathbf{x})$$

Here, $P(\mathbf{y} | \mathbf{x})$ is the conditional probability of a question sequence \mathbf{y} given input sequence \mathbf{x} . The model is a seq2seq model with soft attention described below:

Decoder: Following [273], the conditional factorizes over token level predictions:

$$P(\mathbf{y} | \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} P(y_t | \mathbf{y}_{<t}, \mathbf{x})$$

$\mathbf{y}_{<t}$ represents the subsequence of words generated prior to the time step t . For the decoder, we again follow [273] and model $P(y_t|\mathbf{y}_{<t}, \mathbf{x})$ as:

$$P(y_t|\mathbf{y}_{<t}, \mathbf{x}) = \text{softmax} \left(\mathbf{W} \tanh \left(\mathbf{W}_t [\mathbf{h}_t^{(d)}; \mathbf{c}_t] \right) \right)$$

Here, $\mathbf{h}_t^{(d)}$ is the decoder RNN state at time step t , and \mathbf{c}_t is the attention based encoding of the input sequence \mathbf{x} at decoding time step t (described later). Also \mathbf{W} and \mathbf{W}_t are model parameters to be learned. We use an LSTM with dropout [310] as the decoder RNN. The LSTM generates the new decoder state $\mathbf{h}_t^{(d)}$ given the representation of previously generated word y_{t-1} obtained using a look-up dictionary, and the previous decoder state $\mathbf{h}_{t-1}^{(d)}$.

Encoder: We use a bi-directional LSTM [103] with attention mechanism as our sentence encoder. More specifically, we have two LSTM’s: one that makes a forward pass in the sequence and another that makes a backward pass:

$$\begin{aligned} \vec{\mathbf{h}}_t^{(e)} &= \text{LSTM} \left(x_t, \vec{\mathbf{h}}_{t-1}^{(e)} \right) \\ \overleftarrow{\mathbf{h}}_t^{(e)} &= \text{LSTM} \left(x_t, \overleftarrow{\mathbf{h}}_{t+1}^{(e)} \right) \end{aligned}$$

We follow the LSTM modeling framework with dropout regularization [310] in our implementation. The final context dependent token representation $\mathbf{h}_t^{(e)}$ is the concatenation of the forward and backward pass token representations: $\mathbf{h}_t^{(e)} = [\vec{\mathbf{h}}_t^{(e)}; \overleftarrow{\mathbf{h}}_t^{(e)}]$. To obtain the final context dependent token representation \mathbf{c}_j at the decoding time step j , we take a weighted average over the token representations:

$$\mathbf{c}_j^{(d)} = \sum_{i=1}^{|\mathbf{x}|} a_{ij} \mathbf{h}_i^{(e)}$$

Following [12], the attention weights a_{ij} are calculated by bilinear scoring followed by softmax normalization:

$$\mathbf{a}_{ij} = \frac{\exp \left(\mathbf{h}_j^{(e)T} \mathbf{W} \mathbf{h}_i^{(d)} \right)}{\sum_{i'} \exp \left(\mathbf{h}_j^{(e)T} \mathbf{W} \mathbf{h}_{i'}^{(d)} \right)}$$

Learning and Inference: We train the encoder decoder framework by maximizing data log-likelihood on a large training set with respect to all the model parameters θ . Let $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^N$ be the training set. The data log-likelihood can be written down as:

$$\begin{aligned} \mathcal{L}_{AQ} &= \sum_{i=1}^N \log P \left(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}; \theta \right) \\ &= \sum_{i=1}^N \sum_{j=1}^{|\mathbf{y}^{(i)}|} \log P \left(y_j^{(i)} | \mathbf{x}^{(i)}, \mathbf{y}_{<j}^{(i)}; \theta \right) \end{aligned}$$

We use beam search as in previous works on seq2seq [273] for inference. Also, as in previous works, we introduce a $\langle \text{UNK} \rangle$ token to model rare words during decoding. These $\langle \text{UNK} \rangle$ tokens are finally replaced by the token in the input sentence with the highest attention score.

8.3 A Semi-supervised Framework for Joint Training of QA and AQ models

Finally, we describe our semi-supervised framework for jointly training the QA and AQ models. In this setting, we assume that we are given unlabeled text \mathcal{T} in addition to the passages, question and answer pairs.

Self-training [229, 302], also known as self-teaching, is one of the earliest techniques using both labeled and unlabeled data to improve learning. During self-training, the learner keeps on labeling unlabeled examples and retraining itself on an enlarged labeled training set. We extend the self-training idea to jointly learn two models (namely, QA and AQ) iteratively. The QA and AQ models are first trained on a labeled corpus. Then, the AQ model is used to create more questions from a large unlabeled text corpus and the QA model is used to answer these newly created questions. These new questions (carefully selected by an oracle – details later) are then added to the training set and the two models are retrained. This procedure can be repeated as long as the two models continue to improve. Algorithm 2 describes the procedure in detail.

Algorithm 2: Joint Semi-supervised Training of QA and AQ models

Input : Training set of passage, question, answer triples $(p, q, a) \in \mathcal{D}$, some additional unlabeled text \mathcal{T} , number of joint training iterations N , question selector oracle \mathcal{QS} , and hyper-parameters (initial sample size k and sample multiplier m).

Output: The question answering (QA) and question asking (AQ) models θ_{qa} and θ_{aq} , respectively.

- 1 $\theta_{qa}^{(0)} \leftarrow$ Train initial question answering model.
- 2 $\theta_{aq}^{(0)} \leftarrow$ Train initial question asking model.
- 3 **Initialize** $i=0$
- 4 **while** $i < N$ **do**
- 5 $\mathbf{CQ}_i \leftarrow$ Set of candidate questions based on the unlabeled text \mathcal{T} generated using our question asking model $\theta_{aq}^{(i)}$ which are not in \mathcal{D} .
- 6 $\mathbf{Q}_i \leftarrow k \times m^i$ questions drawn from \mathbf{CQ}_i using our question selector oracle \mathcal{QS} .
- 7 $\mathbf{A}_i \leftarrow$ Set of answers to questions \mathbf{Q}_i obtained using our question answering model $\theta_{qa}^{(i)}$.
- 8 Let the new set of questions \mathbf{Q}_i and corresponding sentences \mathbf{S}_i and answers \mathbf{A}_i be \mathcal{D}' .
- 9 $\theta_{qa}^{(i+1)} \leftarrow$ Update question answering model on \mathcal{D}' .
- 10 $\theta_{aq}^{(i+1)} \leftarrow$ Update question asking model on \mathcal{D}' .
- 11 Add \mathcal{D}' to \mathcal{D} .
- 12 $i++$
- 13 **end**

	SQUAD			MS MARCO			WikiQA		
	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
# Questions	82,326	4,806	5,241	87,341	5,273	5,279	1,040	140	293
# Question-Answer Pairs	676,193	39,510	42,850	440,573	26,442	26,604	20,360	2,733	6,165
# Answers per Question	8.2	8.2	8.2	5.0	5.0	5.0	19.6	19.5	21.0

Table 8.1: Statistics of the three machine comprehension datasets used in evaluating our QA and AQ models.

The Question Selection Oracle

A key challenge in self-training is the selection of which unlabeled data samples to label. The self-training process may erroneously label some unlabeled examples which can sidetrack the learning process. Thus, we implement a question selection oracle which determines which questions to add among a potentially very large set of questions generated by the AQ model in each iteration.

Traditional wisdom in self-training [229, 302] advises selecting a subset of questions on which the models have the highest confidence. We experiment with this idea, introducing self-training oracles which introduce questions in the order of how confident the QA and AQ models are on the correctness of the question. This is calculated by computing the ratio of the score of the question and the second highest scoring question in the beam during decoding in the AQ and QA models and taking an average of the two.

We further explore some key ideas for question selection based on recent works in *curriculum learning* and *diversity* and show that they can be used to further improve the standard self-training oracle.

1. The first idea, *curriculum learning* [17], requires ordering data samples on the easiness scale, so that easy samples can be introduced to the learning algorithm first and harder samples can be introduced successively. The main challenge in learning this curriculum is that it requires the identification of easy and hard samples. However, in our setting, such a ranking of easy and hard questions is difficult to obtain. Moreover, a human judgement of ‘easiness’ of a question might not correlate with what is easy for the algorithmS in the feature and hypothesis space. We explore various heuristics that define a measure of easiness and learn the ordering by selecting samples using this measure.
2. The second idea is that of diversity. A number of cognitive scientists [34] argue that alongside curriculum learning, it is important to introduce diverse (even if sometimes hard) samples. Inspired by this, we introduce a measure of diversity and show that the curriculum learning heuristics introduced by us coupled with diversity leads to further improvements.

Curriculum Learning

Studies in cognitive science [154, 217, 264] have shown that humans learn much better when the training examples are not randomly presented but organized in increasing order of difficulty. In the machine learning community, this idea was introduced in the nomenclature of *curriculum learning* [17], where a curriculum is designed by ranking samples based on manually curated

difficulty measures. A manifestation of that idea is self-paced learning (SPL) [134, 135, 157] which selects questions based on the local loss term of the sample.

We explore the following heuristics for the our oracle:

1) Greedy Optimal (GO): The simplest and greedy optimal heuristic would be to pick a question q which has the minimum expected effect on the QA and AQ models. The expected effect on adding q can be written as:

$$\sum_{a \in \mathcal{A}} p(a^* = a) \mathbb{E}[\mathcal{L}_{QA/AQ}]$$

$p(a^* = a)$ can be estimated by computing the scores of each of the answer candidates for q and normalizing them. $\mathbb{E}[\mathcal{L}_{QA/AQ}]$ can be estimated by retraining the models after adding this question.

2) Change in Objective (CiO): Choose the question q that causes the smallest increase in the QA/AQ model objective. If there are multiple questions with the smallest increase in objective, pick one of them randomly.

3) Mini-max (M^2): Chooses question q that minimizes the regularized expected risk when including the question with the answer candidate a that yields the maximum error.

$$\hat{q} = \arg \min_q \max_{a \in \mathcal{A}} \mathcal{L}_{QA/AQ}$$

4) Expected Change in Objective (ECiO): In this greedy heuristic, we pick a question q which has the minimum expected effect on the model. The expected effect can be written as $\sum_a p(a^* = a) \times \mathbb{E}[L_{QA/AQ}]$. Here, $p(a^* = a)$ can again be achieved by computing the scores of each of the answer candidates for q and normalizing them and $\mathbb{E}[L_{QA/AQ}]$ can be estimated by running inference.

5) Change in Objective-Expected Change in Objective (CiO - ECiO): We pick a question q which has the minimum value of the difference between the change in objective and the expected change in objective described above. Intuitively, the difference represents how much the model is surprised to see this new question.

Diversity

The strategy of introducing easy questions first and then gradually introducing harder questions is intuitive as it helps the learning process. Yet, it has one key deficiency. Under curriculum learning, by focusing on easy questions first, our learning algorithm is usually not exposed to a diverse set of questions. This is particularly a problem for deep-learning approaches that learn representations during the process of learning. Hence, when a harder question arrives, it is usually hard for the learner to adjust to this new question as the current representation may not be appropriate for the new level of difficulty. We tackle this by introducing an explore and exploit (*E&E*) strategy.

The explore and exploit strategy ensures that while we still select easy questions first, we also want to make our selection as diverse as possible. We define a measure for diversity as the angle between the question samples seen as vectors in a feature space: $\angle \mathbf{q}_i, \mathbf{q}_j = \text{Cosine}^{-1} \left(\frac{|\mathbf{q}_i \mathbf{q}_j|}{\|\mathbf{q}_i\| \|\mathbf{q}_j\|} \right)$.

The *E&E* solution picks the question which optimizes a convex combination of the curriculum learning objective and the sum of angles between the candidate question pick and all the candidate questions. The convex combination is tuned on the dev set.

Ensembling

Finally, we introduce an ensembling strategy where we combine any given number of the above heuristics into an ensemble. The ensembling strategy is indeed very simple. It computes the ratio of the score of the suggested question and the average score over remaining questions for all the heuristics and picks the question with the highest ratio. As we will see in our results, this ensembling strategy works well in practice.

8.4 Experiments

Implementation: We perform the same preprocessing on all the texts. We lower-case all texts in the dataset and we use NLTK for word tokenization. For training our neural networks, we only keep the most frequent 50k words (including entity and placeholder markers), and map all other words to a special <UNK> token. We choose word embedding size $d = 100$, and use the 100-dimensional pretrained GloVe word embeddings [216] for initialization. We set k and m (hyperparameters for self-training) by grid search on a held-out development set. The deep learning models are implemented using Keras with the Theano backend.

Datasets: We report our results on three public datasets: *SQUAD* [224], *MS MARCO* [204], and *Wiki QA* [298].

SQUAD is a cloze-style reading comprehension dataset with questions posed by crowd workers on a set of Wikipedia articles, where the answer to each question is a segment of text from the corresponding reading passage. *MS MARCO* contains questions which are real anonymized queries issued through *Bing* or *Cortana* and the documents are related web pages which may or help answer the question. Finally, *WikiQA* is also a dataset of queries taken from Bing query logs. Based on user clicks, each query is associated with a Wikipedia page. Infact, the summary paragraph of the page as is taken as candidate answer sentences, with labels on whether the sentence is a correct answer to the question provided by crowd workers.

While *WikiQA* is directly an answer sentence selection task, the other two are not. Yet, we treat the *SQUAD* and *MS MARCO* tasks as the answer sentence selection task assuming a one to one correspondence between answer sentences and annotated correct answer spans³. Also, *SQUAD* and *MS MARCO* have a hidden test set so we only use the training and development sets for our evaluation purposes and we further split the provided development set into a dev and test set. This is also the data analysis setting used in previous works [70, 274]. In fact, we use the same setting as in [274] for comparison. The statistics of the three datasets and the respective train, dev and test splits are given in Table 8.1. For *WikiQA* dataset, we use the standard data split given in [298].

We use a large randomly subsampled corpus of English Wikipedia and use the first paragraph of each document as unlabeled text for self-training. In our experiments, we describe various

³A very small proportion of answers (< 0.2% in training set) span two or more sentences.

	SQUAD			MS MARCO		
	MAP	MRR	P@1	MAP	MRR	P@1
WordCnt	0.3956	0.4014	0.1789	0.8089	0.8168	0.6887
NormWordCnt	0.4223	0.4287	0.2030	0.8714	0.8787	0.7958
CDSSM	0.4425	0.4489	0.2284	0.7978	0.8041	0.6721
ABCNN	0.4691	0.4767	0.2629	0.8685	0.8750	0.7843
QA	0.4712	0.4783	0.2628	0.8580	0.8647	0.7740
QA-SelfTrain(100)	0.5236	0.4934	0.2734	0.8809	0.8904	0.7987
QA-SelfTrain(1000)	0.5372	0.5022	0.2842	0.8848	0.8946	0.8012
M²	0.4885	0.4896	0.2682	0.8603	0.8721	0.7852
ECiO	0.4979	0.4940	0.2725	0.8769	0.8856	0.7928
GO	0.5057	0.4950	0.2744	0.8786	0.8886	0.7932
CiO	0.5112	0.4983	0.2772	0.8788	0.8903	0.7950
CiO-ECiO	0.5166	0.5002	0.2803	0.8811	0.8919	0.7981
Ensemble	0.5386	0.5041	0.2837	0.8858	0.8954	0.8003
Ensemble+E&E(10000)	0.5393	0.5067	0.2887	0.8887	0.8961	0.8007

Table 8.2: Performance of our model variants and the four QA baselines on *SQUAD* and *MS MARCO* datasets. The baseline numbers are taken from [274]. The grey part of the table shows the effect of various question selection heuristics on our SelfTrain(10000) models.

models trained on varying amount of unlabeled data and report the implications of having more unlabeled data.

Evaluation Metrics: Following [70, 274], we evaluate our QA system with three standard evaluation metrics: *Mean Average Precision* (MAP), *Mean Reciprocal Rank* (MRR) and *Precision@1* (P@1). For question asking, we use some automatic evaluation metrics from machine translation and summarization: *BLEU-4* [212], *METEOR* [67] and *Rouge_L* [172] to measure the overlap between the generated questions and ground truth questions.

Baselines: We use the following four baselines that have been used in previous works [274] for our QA model. The first two baselines have been taken from [298, 305], and are based on simple word overlap which have been shown to be very strong baselines. **WordCnt** and **NormWordCnt** compute word co-occurrence between a question sentence and the candidate answer sentence. While **WordCnt** uses unnormalized word co-occurrence, **NormWordCnt** uses normalized word co-occurrence. The third and fourth baselines are **CDSSM** [258] and **ABCNN** [305] which use a neural network approach to model semantic relatedness of sentence pairs.

For the *WikiQA* dataset, we follow [274] and compare to various accomplished baselines on this answer sentence selection task namely, **CNN** [298], **APCNN** [248], **NASM** [193] and **ABCNN** [305].

For AQ, we compare our model against the following four baselines which have also been used in some previous works [70]. The first baseline is a simple **IR** baselines taken from [233] which generates questions by memorizing them from the training set and uses edit distance [163] to calculate distance between a question and the input sentence. The second baseline is a MT system, **MOSES** [149] where the idea is to model question generation as a translation task where raw sentences are treated as source language texts and questions are treated as target language texts. Following [70], KenLM [112] is used as a tri-gram language model on target side texts,

	MAP	MRR
CNN	0.6652	0.6520
APCNN	0.6957	0.6886
NASM	0.7069	0.6886
ABCNN	0.7018	0.6921
QA	0.6914	0.6747
QA-SelfTrain(100)	0.7133	0.7036
QA-SelfTrain(1000)	0.7185	0.7076
M ²	0.6936	0.6786
ECiO	0.6957	0.6827
GO	0.6995	0.6850
CiO	0.7022	0.6938
CiO-ECiO	0.7094	0.6987
Ensemble	0.7178	0.7076
Ensemble+E&E(10000)	0.7194	0.7080

Table 8.3: Performance of our model variants and the four QA baselines on *WikiQA*. The baseline numbers are taken from [274]. The grey part of the table shows the effect of various question selection heuristics on our SelfTrain(10000) models.

the system is tuned with MERT on dev set. The third baseline, **DirectIn**, uses the longest sub-sentence of the input sentence (using a set of simple sentence splitters) as the question. The fourth and final baseline, **H&S** is a rule-based overgenerate-and-rank system proposed by [116]. Following [70], we take the top question in the ranked list for comparison.

Evaluating Question Answering: First, we evaluate the QA model. **QA** is the variant of our model without self-training. **QA/AQ-SelfTrain(K)** is the variant where we perform self-training using K Wikipedia paragraphs. For all the **QA/AQ-SelfTrain(K)** models, we use an Ensemble+E&E as the question selection oracle, which performs the best. We vary K to see the impact of the size of unlabeled Wikipedia paragraphs on the self-training model. For the Self-training(10000) models (QA or AQ jointly trained with self-training using 10000 Wikipedia paragraphs), we show results for various question selection oracles (in shaded grey). We will discuss the implications of the question selection oracle later in our analysis.

Table 8.2 shows the results of the QA evaluations on the *SQUAD* and *MS MARCO* datasets. We can observe that our QA model has competitive or better performance over all the four baselines on both datasets in terms of all the three evaluation metrics. When we perform self-training, learning both the QA and AQ models jointly using English Wikipedia as unlabeled data, we see substantially better results. Similarly, Table 8.3 shows the results of the QA evaluations on the *WikiQA* dataset. We can again observe that our QA model is competitive to the four strong baselines. When we perform self-training while jointly learning the QA and AQ models, we see improved performance.

Evaluating Question Asking: Table 8.4 shows the results of the AQ evaluations on the three datasets on each of the three evaluation metrics: *BLEU4*, *METEOR* and *ROUGE*. We can observe that the AQ model described in our paper performs much better than each of the four baselines. We again observe that self-training while jointly training the QA and AQ models leads to even

	SQUAD			MS MARCO			WikiQA		
	BIEU4	METEOR	ROUGE	BIEU4	METEOR	ROUGE	BIEU4	METEOR	ROUGE
IR	1.07	7.77	20.85	0.81	5.42	15.78	0.93	6.89	19.98
MOSES	0.31	10.49	17.88	0.27	9.74	15.82	0.32	10.26	17.27
DirectIn	11.25	14.91	22.51	10.82	13.35	20.38	10.94	14.18	22.01
H&S	11.23	16.00	31.03	10.16	15.07	30.00	10.35	15.30	30.72
AQ	12.31	16.67	39.78	11.14	15.60	37.26	11.38	16.08	38.42
AQ-SelfTrain(100)	14.14	18.70	42.46	13.25	17.10	40.28	13.10	17.00	40.93
AQ-SelfTrain(1000)	14.27	18.78	42.93	13.61	17.87	41.23	13.22	18.34	42.72
M²	12.46	16.95	40.27	11.56	15.93	38.32	11.83	16.84	39.26
ECiO	12.79	17.40	40.92	12.11	16.32	38.86	12.14	17.04	39.82
GO	13.12	17.73	41.24	12.75	16.66	39.47	12.56	17.62	40.31
CiO	13.59	17.94	41.57	13.00	16.83	40.02	12.88	18.13	40.97
CiO-ECiO	13.97	18.18	41.90	13.41	17.16	40.65	13.22	18.34	41.28
Ensemble	14.37	18.57	42.73	13.56	17.40	40.92	14.26	18.91	43.26
Ensemble+E&E(10000)	14.28	18.79	42.97	13.74	17.89	41.07	15.26	19.45	44.77

Table 8.4: Performance of our model variants and the four AQ baselines on the *SQUAD*, *MS MARCO* and *WikiQA* datasets. The grey part of the table shows the effect of various question selection heuristics on our SelfTrain(10000) models.

better performance. These results show that self-training and leveraging the relationship between QA and AQ is very useful for boosting the performance of both models, while additionally only using cheap unlabeled data.

Evaluating the Question Selection Oracle: As discussed earlier, the choice of which subset of questions to add to our labeled dataset while self-training is important. To evaluate the various heuristics proposed in our paper, we show the effect of the question selection oracle on the final QA and AQ performance in Tables 8.2, 8.3 and 8.4. These comparisons are shown in the shaded grey portions of the tables for the **QA/AQ-SelfTrain(10000)** model, i.e. self-training with 10,000 Wikipedia paragraphs as unlabeled data.

We can observe that all the proposed heuristics (and ensembling and diversity strategies) lead to improvements in the final performance of both QA and AQ. The heuristics arranged in increasing order of performance are: **M²**, **ECiO**, **GO**, **CiO** and **CiO-ECiO**. While the choice of which heuristic to pick seems to make a lesser impact on the final performance, we do see a much more significant performance gain by **ensembling** to combine the various heuristics and using **E&E** to incorporate diversity. As described earlier, the incorporation of diversity is important because the neural network models which learnt latent representations of data usually find it hard to adjust to new level of difficulty of questions as the current representation may not be appropriate for the new level of difficulty.

Does more unsupervised text always help? An important question to ask is: Does more unsupervised text always help our models? That is, will the performance always increase if we add more and more unsupervised data during self-training. According to our results in Tables 8.2, 8.3 and 8.4, the answer is "perhaps yes". As we can observe from these tables, the performance of the QA and AQ models improves as we increase K , the size of the unsupervised data during training of the various **QA+SelfTrain(K)** models. Having said that, we do see a tapering effect on the performance results, so it is also clear that the performance will be capped by some upper-bound and we will need better ways of modeling semantics and natural language understanding to make progress in QA and AQ thereafter.

8.5 Conclusion

We described self-training algorithms for jointly learning to answer and ask questions while leveraging unlabeled data. We experimented with neural models for question answering and question generation and various careful strategies for question filtering based on curriculum learning and diversity promotion. This led to improved performance for both question answering and question generation on multiple datasets and new state-of-the-art results on WikiQA and TrecQA datasets. In the future, we would like to extend this observation to reduce the amount of supervision needed to model the other question answering tasks especially the other standardized test problems tackled by us. We would also like to extend this idea to incorporate various redundant question answering and question generation models thereby using consistency as a way to estimate correctness.

Chapter 9

Conclusion and Future Work

In this thesis, we proposed the task of teaching machines to automatically solve some standardized tests such as reading comprehensions, and various elementary school math and science tests as a way to benchmark AI. We proposed two broad set of techniques based on latent answer-entailing structures when solving tests without any underlying domain theory and a parsing to programs framework for solving tests with an underlying domain theory such a math and science tests. We also laid out approaches to generate questions and how these automatically generated noisy questions can further be used to improve the question answering models. We augment training of these approaches with unlabelled data to handle the issue of lack of supervision. In the future, we would like to generalize these two frameworks and release easy-to-use toolkits for practitioners to build their own question answering systems in their domain of interest. We would like to build more such systems for examinations in more advanced areas such as medical diagnostics, accountancy, finance, etc. We would like to collaborate with various MOOCs and standardized testing preparation firms such as ETS and explore the use of this technology as a tool to assist students.

This thesis also raises more questions and there are many avenues for future exploration. In particular, we are interested in the following questions:

- What is the limit of the proposition of using standardized tests as a benchmark for AI as a driver towards developing AI? While a system that can answer a range of standardized test problems is still far in the horizon and attempts to pass standardized tests can still help us make progress towards machine intelligence, there are a number of human abilities such as creativity, common sense, spatial reasoning, etc. that are not tested by standardized tests. So do we need more another benchmark in addition to standardized tests?
- How do we better model and incorporate common sense knowledge in these models?
- How can we build solutions that can together solve a broader set of standardized test problems? How do we generalize across different tasks?
- How do we build these solutions with minimal task-specific supervision? Can we use prior work in *representation learning*, *active (perhaps interactive) learning*, *few shot learning*, *weak supervision*, *domain transfer* and *multi-task learning* to achieve this goal.
- A lot of prior knowledge required to solve these problems is present in symbolic form. However, it is well known today that sub-symbolic representations work very well for

many NLP problems. Can we combine these symbolic and sub-symbolic methods in order to incorporate this prior knowledge and yet be able to leverage the ability to learn sub-symbolic representations?

- Do we really even need to solve such hard problems in a completely end-to-end way? Can we come up with human-in-the-loop solutions [240] to leverage the rich domain knowledge that humans are very good at?

Bibliography

- [1] Sunita B Aher and LMRJ Lobo. Combination of machine learning algorithms for recommendation of courses in e-learning system based on historical data. *Knowledge-Based Systems*, 51:1–14, 2013. 2
- [2] Kirsti M Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102, 2005. 2
- [3] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, 2012. 7.2, 7.5.1
- [4] R.L. Allington and P.M. Cunningham. Children benefit from modeling, demonstration, and explanation, 2010. 7.4.1
- [5] Chris Alvin, Sumit Gulwani, Rupak Majumdar, and Supratik Mukhopadhyay. Synthesis of geometry proof problems. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 245–252, 2014. 7.1
- [6] Gabor Angeli and Christopher D. Manning. Naturalli: Natural logic inference for common sense reasoning. In *EMNLP*, 2014. 7.1
- [7] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015. URL <http://arxiv.org/abs/1505.00468>. 1
- [8] Noriko H Arai and Takuya Matsuzaki. The impact of ai on education—can a robot get into the university of tokyo? In *Proc. ICCE*, pages 1034–1042, 2014. 2
- [9] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. 7.4.1
- [10] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 1, 2013. 7.1
- [11] Yigal Attali and Jill Burstein. Automated essay scoring with e-rater® v. 2.0. *ETS Research Report Series*, 2004(2), 2004. 2
- [12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 8.1, 8.2
- [13] Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni Mausam, and Robert Bart.

“out of the box” information extraction: a case study using bio-medical texts. Technical report, University of Washington, 2002. 7.1

- [14] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W13-2322>. 1, 5.2
- [15] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2670–2676, 2007. 7.1
- [16] Marco Baroni and Alessandro Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721, 2010. 5.2
- [17] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009. 1, 8, 1, 13
- [18] Randy Elliot Bennett and Isaac I Bejar. Validity and automad scoring: It’s not only the scoring. *Educational Measurement: Issues and Practice*, 17(4):9–17, 1998. 2
- [19] J. Berant and P. Liang. Semantic parsing via paraphrasing. In *ACL*, 2014. 7.1
- [20] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1533–1544, 2013. 4, 7.1
- [21] Phil Blunsom and Trevor Cohn. Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 65–72. Association for Computational Linguistics, 2006. 1, 4, 4.2
- [22] Antoine Bordes, Nicolas Usunier, and Jason Weston. Label ranking under ambiguous supervision for learning semantic correspondences. In *ICML*, 2010. 7.1
- [23] Ronald J Brachman and Hector J Levesque. *Readings in knowledge representation*. Morgan Kaufmann Publishers Inc., 1985. 7.1
- [24] SRK Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. Learning high-level planning from text. In *ACL*, 2012. 7.1
- [25] Eric Breck, Marc Light, Gideon S Mann, Ellen Riloff, Brianne Brown, Pranav Anand, Mats Rooth, and Michael Thelen. Looking under the hood: Tools for diagnosing your question answering engine. In *Proceedings of the workshop on Open-domain question answering-Volume 12*, pages 1–8. Association for Computational Linguistics, 2001. 2
- [26] Peter Brusilovsky and Christoph Peylo. Adaptive and intelligent web-based educational systems. *International Journal of Artificial Intelligence in Education (IJAIED)*, 13:159–172, 2003. 2

- [27] Daphna Buchsbaum, Alison Gopnik, Thomas L Griffiths, and Patrick Shafto. Children’s imitation of causal action sequences is influenced by statistical and pedagogical evidence. *Cognition*, 120(3):331–340, 2011. 7.4.1
- [28] William C. Bulko. Understanding text with an accompanying diagram. In *IEA/AIE*, 1988. 7.1
- [29] John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, et al. Issues, tasks and program structures to roadmap research in question & answering (q&a). In *In Document Understanding Conferences Roadmapping Documents*, pages 1–35, 2001. 7.4.1
- [30] Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, and Martin Chodorow. Enriching automated essay scoring using discourse marking. *ERIC*, 2001. 2
- [31] Lucas P Butler and Ellen M Markman. Preschoolers use pedagogical cues to guide radical reorganization of category knowledge. *Cognition*, 130(1):116–127, 2014. 7.4.1
- [32] Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. Knowitnow: Fast, scalable information extraction from the web. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 563–570, 2005. 7.1
- [33] Sander Canisius and Caroline Sporleder. Bootstrapping information extraction from field books. In *EMNLP-CoNLL*, pages 827–836, 2007. 7.1
- [34] Nathaniel Freeman Cantor. *Dynamics of learning*. Foster and Stewart publishing corporation, Buffalo, NY, 1946. 2
- [35] João Carreira, Fuxin Li, and Cristian Sminchisescu. Object recognition by sequential figure-ground ranking. *International journal of computer vision*, 98(3):243–262, 2012. 7.2, 7.5.1
- [36] Tsz Ping Chan, Chris Callison-Burch, and Benjamin Van Durme. Reranking bilingually extracted paraphrases using monolingual distributional similarity. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 33–42, 2011. 4.2, 5.2
- [37] Yee Seng Chan and Hwee Tou Ng. Maxsim: A maximum similarity metric for machine translation evaluation. In *The 2008 Annual Conference of the Association for Computational Linguistics (ACL)*, 2008. ??
- [38] Chia-Hui Chang, Chun-Nan Hsu, and Shao-Cheng Lui. Automatic information extraction from semi-structured web pages by pattern discovery. *Decision Support Systems*, 35(1): 129–147, 2003. 7.1, 7.3.4
- [39] Chia-Hui Chang, Mohammed Kayed, Moheb R Girgis, and Khaled F Shaalan. A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering*, 18(10):1411–1428, 2006. 7.1
- [40] Maria D Chang, Jon W Wetzel, and Kenneth D Forbus. Spatial reasoning in comparative analyses of physics diagrams. In *International Conference on Spatial Cognition*, pages

268–282. Springer, 2014. 7.1

- [41] Ming-Wei Chang, Lev Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 280–287, 2007. 3.4.2
- [42] Ming-Wei Chang, Lev Ratinov, and Dan Roth. Structured learning with constrained conditional models. *Machine learning*, 88(3):399–431, 2012. 3.4, 3.4.2, 7.3.1
- [43] Eugene Charniak. *Toward a model of children’s story comprehension*. PhD thesis, Massachusetts Institute of Technology, 1972. 2
- [44] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*, 2016. 8.1
- [45] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051, 2017. 2
- [46] David Chen, Joohyun Kim, and Raymond Mooney. Training a multilingual sportscaster: Using perceptual context to learn language. *JAIR*, 37, 2010. 7.1
- [47] David L. Chen and Raymond J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pages 859–865, August 2011. 7.1
- [48] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*, pages 577–585, 2015. 8.2
- [49] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. *Machine proofs in geometry: Automated production of readable proofs for geometry theorems*, volume 6. World Scientific, 1994. 7.1
- [50] Peter Clark. Elementary School Science and Math Tests as a Driver for AI:Take the Aristo Challenge! In *Proceedings of IAAI*, 2015. 1, 6
- [51] Peter Clark and Oren Etzioni. My computer is an honor student - but how intelligent is it? standardized tests as a measure of ai. In *Proceedings of AI Magazine*, 2016.
- [52] Peter Clark, Phil Harrison, Niranjana Balasubramanian, and Oren Etzioni. Constructing a textual kb from a biology textbook. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 74–78. Association for Computational Linguistics, 2012. 7.1
- [53] Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. Combining retrieval, statistics, and inference to answer elementary science questions. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 6, 6.4
- [54] Stephen Clark and Stephen Pulman. Combining symbolic and distributional models of meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction at Stanford at ICAI 2007*, 2007. 1
- [55] Kenneth Mark Colby. *Artificial Paranoia: A Computer Simulation of Paranoid Processes*. Elsevier Science Inc., New York, NY, USA, 1975. ISBN 0080181627. 1, 2

- [56] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002. 3.4.2
- [57] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011. 4.2, 5.2, 6.3
- [58] Jim Cowie and Wendy Lehnert. Information extraction. *Communications of the ACM*, 39(1), 1996. 7.1
- [59] S. Cucerzan and E. Agichtein. Factoid question answering over unstructured and structured content on the web. In *Proceedings of TREC 2005*, 2005. 4.2
- [60] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *ACL*, 2004. 7.1
- [61] Bhavana Dalvi, Sumithra Bhakthavatsalam, Chris Clark, Peter Clark, Oren Etzioni, Anthony Fader, and Dirk Groeneveld. IKE - an interactive tool for knowledge extraction. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016*, pages 12–17, 2016. 7.1
- [62] Donald Davidson. *The individuation of events*. Springer, 1969. 5
- [63] Randall Davis, Howard Shrobe, and Peter Szolovits. What is a knowledge representation? *AI magazine*, 14(1):17, 1993. 7.1
- [64] Tom Davis. Geometry with computers. Technical report, Stanford University, 2006. 7.1
- [65] T. E. de Campos, B. R. Babu, M. Varma, and Manik Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal, February 2009*. 7.2, 7.5.1
- [66] Michael Denkowski and Alon Lavie. Extending the meteor machine translation evaluation metric to the phrase level. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 250–253. Association for Computational Linguistics, 2010. ??
- [67] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380, 2014. 8.4
- [68] Pedro Domingos, Daniel Lowd, Stanley Kok, Aniruddh Nath, Hoifung Poon, Matthew Richardson, and Parag Singla. Unifying logical and statistical ai. In *Proceedings of the 31st Annual Symposium on Logic in Computer Science*, pages 1–11, 2016. 1
- [69] Hendrik Drachler, Katrien Verbert, Olga C Santos, and Nikos Manouselis. Panorama of recommender systems to support learning. In *Recommender systems handbook*, pages 421–451. Springer, 2015. 2
- [70] Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*, 2017. 2, 8.2, 8.4, 8.4

- [71] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. 7.2, 7.5.1
- [72] Martin Ebner and Andreas Holzinger. Successful implementation of user-centered game based learning in higher education: An example from civil engineering. *Computers & education*, 49(3):873–890, 2007. 2
- [73] Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. Reading to learn: Constructing features from semantic abstracts. In *EMNLP*, 2009. 7.1
- [74] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric model for background subtraction. In *European conference on computer vision*, pages 751–767. Springer, 2000. 7.5.1
- [75] Katrin Erk and Sebastian Padó. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '08*, pages 897–906, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL <http://dl.acm.org/citation.cfm?id=1613715.1613831>. 5.2
- [76] Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Methods for domain-independent information extraction from the web: An experimental comparison. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 391–398, 2004. 7.1, 7.3.4
- [77] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008. 7.1
- [78] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117, 2004. 4, 4.1.2
- [79] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, et al. From captions to visual concepts and back. In *CVPR*, 2014. 7.1
- [80] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. In *ECCV*, 2010. 7.1
- [81] Edward Feigenbaum, Pamela McCorduck, and H Penny Nii. *The rise of the expert company*, volume 240. New York: Times Books, 1988. 7.1
- [82] Edward A Feigenbaum and Julian Feldman. *Computers and thought*. The AAAI Press, 1963. 7.1
- [83] Richard M Felder, Donald R Woods, James E Stice, and Armando Rugarcia. The future of engineering education ii. teaching methods that work. *Chemical Engineering Education*, pages 26–39, 2000. 7.4.1
- [84] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

- [85] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010. 7.2, 7.5.1
- [86] Vanessa Wei Feng and Graeme Hirst. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521, 2014. 11, 2, 5, ??
- [87] Ronald W Ferguson and Kenneth D Forbus. Telling juxtapositions: Using repetition and alignable difference in diagram understanding. *Advances in analogy research*, pages 109–117, 1998. 7.1
- [88] Ronald W Ferguson and Kenneth D Forbus. Georep: A flexible tool for spatial representation of line drawings. In *AAAI/IAAI*, pages 510–516, 2000. 7.1
- [89] David A Ferrucci. Introduction to *Watson*. *IBM Journal of Research and Development*, 56(3.4):1–1, 2012. 4.1.2, 6
- [90] Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics, 2006. 7.3.1
- [91] Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. A discriminative graph-based parser for the abstract meaning representation. In *ACL*, 2014. 7.1
- [92] Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436. Association for Computational Linguistics, June 2014. 2
- [93] Robert M French. Subcognition and the limits of the turing test. *Mind*, 99:53–65, 1996. 1
- [94] Dov M Gabbay, Christopher John Hogger, and John Alan Robinson. *Handbook of Logic in Artificial Intelligence and Logic Programming: Volume 5: Logic Programming*. Clarendon Press, 1998. 7.1
- [95] Xiao-Shan Gao and Qiang Lin. Mmp/geometer—a software package for automated geometric reasoning. In *International Workshop on Automated Deduction in Geometry*, pages 44–66. Springer, 2002. 7.1
- [96] Ruifang Ge and Raymond J. Mooney. Discriminative reranking for semantic parsing. In *ACL*, 2006. 7.1
- [97] Dan Goldwasser and Dan Roth. Learning from natural instructions. In *IJCAI*, 2011. 7.1
- [98] Dan Goldwasser and Dan Roth. Learning from natural instructions. *Machine Learning*, 94(2):205–232, 2014. 7.4.1

- [99] Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. Improving image-sentence embeddings using large weakly annotated photo collections. In *ECCV*, 2014. 7.1
- [100] Noah D Goodman and Andreas Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. <http://dippl.org>, 2014. Accessed: 2017-3-14. 1
- [101] John L Gordon. Creating knowledge maps by exploiting dependent relationships. *Knowledge-based systems*, 13(2):71–79, 2000. 2
- [102] Kartik Goyal, Sujay Kumar Jauhar, Huiying Li, Mrinmaya Sachan, Shashank Srivastava, and Eduard H. Hovy. A structured distributional semantic model for event co-reference. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 467–473, 2013. 5.2
- [103] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. *Artificial Neural Networks: Formal Models and Their Applications—ICANN 2005*, pages 753–753, 2005. 8.1, 8.2
- [104] Eugene Grois and David C Wilkins. Learning strategies for story comprehension: a reinforcement learning approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 257–264. ACM, 2005. 2
- [105] Chunhui Gu, Joseph J Lim, Pablo Arbeláez, and Jitendra Malik. Recognition using regions. In *Computer Vision and Pattern Recognition (CVPR), 2009*, pages 1030–1037. IEEE, 2009. 7.2, 7.5.1
- [106] Sumit Gulwani, Vijay Anand Korthikanti, and Ashish Tiwari. Synthesizing geometry constructions. In *ACM SIGPLAN Notices*, volume 46–6, pages 50–61. ACM, 2011. 7.1
- [107] Sonal Gupta and Raymond J. Mooney. Using closed captions as supervision for video activity recognition. In *AAAI*, 2010. 7.1
- [108] Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir. Reasoning about robocup soccer narratives. In *UAI*, 2011. 7.1
- [109] Sanda M. Harabagiu, Steven J. Maiorano, and Marius A. Paşca. Open-domain textual question answering techniques. *Natural Language Engineering*, 9(3):231–267, September 2003. ISSN 1351-3249. doi: 10.1017/S1351324903003176. URL <http://dx.doi.org/10.1017/S1351324903003176>. 2
- [110] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988. 7.3.1, 7.2, 7.5.1
- [111] Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat. *Building Expert Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983. ISBN 0-201-10686-8. 7.2
- [112] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable modified kneser-ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages

- 690–696. Association for Computational Linguistics, 2013. 8.4
- [113] Mary Hegarty and Marcel Adam Just. 10 understanding machines from text and diagrams. *Knowledge acquisition from text and pictures*, 1989. 7.1
- [114] Michael Heilman. *Automatic factual question generation from text*. PhD thesis, Carnegie Mellon University, 2011. 2
- [115] Michael Heilman and Noah A Smith. Question generation via overgenerating transformations and ranking. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA LANGUAGE TECHNOLOGIES INST, 2009. 8
- [116] Michael Heilman and Noah A Smith. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics, 2010. 8.4
- [117] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692, 2015. 2, 8.1
- [118] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children’s books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015. 2
- [119] Lynette Hirschman, Marc Light, Eric Breck, and John D Burger. Deep read: A reading comprehension system. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 325–332. Association for Computational Linguistics, 1999. 2
- [120] Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. In *NAACL*, 2015. 7.1
- [121] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4.2, 6.4
- [122] Mark Hopkins, Cristian Petrescu-Prahova, Roie Levin, Ronan Le Bras, Alvaro Herrasti, and Vidur Joshi. Beyond sentential semantic parsing: Tackling the math SAT with a cascade of tree transducers. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 806–815, 2017. 2
- [123] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of EMNLP*, 2014. 2
- [124] Minghao Hu, Yuxing Peng, and Xipeng Qiu. Mnemonic reader for machine comprehension. *CoRR*, abs/1705.02798, 2017. 2
- [125] Gwo-Jen Hwang. A conceptual map model for developing intelligent tutoring systems. *Computers & Education*, 40(3):217–235, 2003. 2
- [126] S Impedovo, L Ottaviano, and S Occhinegro. Optical character recognition survey. *International Journal of Pattern Recognition and Artificial Intelligence*, 5(01n02):

1–24, 1991. 7.5.1

- [127] Shachar Itzhaky, Sumit Gulwani, Neil Immerman, and Mooly Sagiv. Solving geometry problems using a combination of symbolic and numerical reasoning. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 457–472. Springer, 2013. 7.1
- [128] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Empirical Methods in Natural Language Processing*, 2014. 4.2, 6.4
- [129] Mohit Iyyer, Jordan L. Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 633–644, 2014. 2
- [130] Peter Jackson. *Introduction to expert systems*. Addison-Wesley Pub. Co., Reading, MA, 1986. 1, 7
- [131] Peter Jansen, Mihai Surdeanu, and Peter Clark. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 977–986, 2014. 6.3
- [132] Peter Jansen, Mihai Surdeanu, and Peter Clark. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 977–986, 2014. 4.2, 1, 5.2
- [133] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002. 4.2
- [134] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the ACM International Conference on Multimedia*, pages 547–556. ACM, 2014. 13
- [135] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. 13
- [136] Claudio Rosito Jung and Rodrigo Schramm. Rectangle detection based on a windowed hough transform. In *Proceedings of the Computer Graphics and Image Processing, XVII Brazilian Symposium, SIBGRAPI '04*, pages 113–120. IEEE Computer Society, 2004. 7.1
- [137] Deepak Kapur. Using gröbner bases to reason about geometry problems. *Journal of Symbolic Computation*, 2(4):399–408, 1986. 7.1
- [138] Rohit J. Kate and Raymond J. Mooney. Learning language semantics from ambiguous supervision. In *AAAI*, 2007. 7.1
- [139] Rohit J Kate, Yuk Wah, Wong Raymond, and J Mooney. Learning to transform natural to formal languages. In *Proceedings of AAAI-05*. Citeseer, 2005. 1, 7.1

- [140] Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. Question answering via integer programming over semi-structured knowledge. *arXiv preprint arXiv:1604.06076*, 2016. 2
- [141] Tushar Khot, Niranjan Balasubramanian, Eric Gribkoff, Ashish Sabharwal, Peter Clark, and Oren Etzioni. Markov logic networks for natural language question answering. *arXiv preprint arXiv:1507.03045*, 2015. 2, 6
- [142] Tushar Khot, Ashish Sabharwal, and Peter Clark. Answering complex questions using open information extraction. *arXiv preprint arXiv:1704.05572*, 2017. 2
- [143] Joo Hyun Kim and Raymond J. Mooney. Adapting discriminative reranking to grounded language learning. In *ACL*, 2013. 7.1
- [144] Matthew Klenk and Ken Forbus. Cognitive modeling of analogy events in physics problem solving from examples. Technical report, NORTHWESTERN UNIV EVANSTON IL, 2007. 7.1
- [145] Matthew Klenk and Ken Forbus. Measuring the level of transfer learning by an ap physics problem-solver. In *AAAI-07/IAAI-07 Proceedings: 22nd AAAI Conference on Artificial Intelligence and the 19th Innovative Applications of Artificial Intelligence Conference*, pages 446–451, 2007. 7.1
- [146] Matthew Klenk and Ken Forbus. Analogical model formulation for transfer learning in ap physics. *Artificial intelligence*, 173(18):1615–1638, 2009. 7.1
- [147] Matthew Klenk and Ken Forbus. Exploiting persistent mappings in cross-domain analogical learning of physical domains. *Artificial intelligence*, 195:398–417, 2013. 7.1
- [148] Matthew Klenk, Kenneth D Forbus, Emmett Tomai, Hyeonkyeong Kim, and Brian Kyckelhahn. Solving everyday physical reasoning problems by analogy using sketches. In *Proceedings of NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 20. MIT Press, 2005. 7.1
- [149] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, ACL '07*, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. 8.4
- [150] Iasonas Kokkinos. Highly accurate boundary detection and grouping. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2520–2527. IEEE, 2010. 7.2, 7.5.1
- [151] R Koncel-Kedziorski, Hannaneh Hajishirzi, and Ali Farhadi. Multi-resolution language grounding with weak supervision. In *EMNLP*, 2014. 7.1
- [152] Jayant Krishnamurthy, Oyvind Tafjord, and Aniruddha Kembhavi. Semantic parsing to probabilistic programs for situated question answering. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages

- 160–170. The Association for Computational Linguistics, 2016. 7.4.1, 7.4.1, 7.4.1
- [153] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 7.2, 7.5.1
- [154] Kai A Krueger and Peter Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394, 2009. 13
- [155] Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. Baby talk: Understanding and generating image descriptions. In *CVPR*, 2011. 7.1
- [156] Nidhi Kulkarni and Mark Finlayson. jmwe: A java toolkit for detecting multi-word expressions. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 122–124, 2011. 5
- [157] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010. 13
- [158] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. Learning to automatically solve algebra word problems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2014. 1, 2
- [159] T Kwiatkowski, E Choi, Y Artzi, and L Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*, 2013. 7.1
- [160] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289, 2001. 3.4.2, 7.3.4
- [161] Tao Lei, Fan Long, Regina Barzilay, and Martin C Rinard. From natural language specifications to program input parsers. In *Association for Computational Linguistics (ACL)*, 2013. 7.1
- [162] Jochen L Leidner, Tiphaine Dalmas, Bonnie Webber, Johan Bos, and Claire Grover. Automatic multi-layer corpus annotation for evaluating question answering methods: Cbc4kids. In *In Proceedings of the 3rd International Workshop on Linguistically Interpreted Corpora*, 2003. 2
- [163] VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966. 8.4
- [164] Hector J Levesque. The winograd schema challenge. In *Proceeding of KR*, 2010. 1
- [165] Keqiang Li, Xiaoqing Lu, Haibin Ling, Lu Liu, Tianxiao Feng, and Zhi Tang. Detection of overlapped quadrangles in plane geometric figures. In *ICDAR*, pages 260–264. IEEE, 2013. 7.1
- [166] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7, 2002. 4.2

- [167] Yang Li and Peter Clark. Answering elementary science questions by constructing coherent scenes using background knowledge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2007–2012, 2015. 6
- [168] Chen Liang, Zhaohui Wu, Wenyi Huang, and C Lee Giles. Measuring prerequisite relations among concepts. In *EMNLP*, pages 1668–1674, 2015. 7.1
- [169] Percy Liang, Michael I. Jordan, and Dan Klein. Learning semantic correspondences with less supervision. In *ACLAFNLP*, 2009. 7.1
- [170] Percy Liang, Michael I Jordan, and Dan Klein. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 590–599. Association for Computational Linguistics, 2011. 7.1
- [171] Vladimir Lifschitz, Leora Morgenstern, and David Plaisted. Knowledge representation and classical logic. *Foundations of Artificial Intelligence*, 3:3–88, 2008. 7.1
- [172] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *In Proceedings of the ACL-04 workshop: Text summarization branches out*, 2004. ??, 8.4
- [173] Chungan Lin and Ramakant Nevatia. Building detection and description from a single intensity image, 1998. 7.1
- [174] Dekang Lin and Patrick Pantel. Dirt@ sbt@ discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328, 2001. 4.2, 5.2
- [175] Shieu-Hong Lin. Data mining for student retention management. *Journal of Computing Sciences in Colleges*, 27(4):92–99, 2012. 2
- [176] Xinggang Lin, Shigeyoshi Shimotsuji, Michihiko Minoh, and Toshiyuki Sakai. Efficient diagram understanding with characteristic pattern detection. *CVGIP*, 30(1), 1985. 7.1
- [177] Tony Lindeberg. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention. *International Journal of Computer Vision*, 11(3):283–318, 1993. 7.5.1
- [178] Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Andrew Senior, Fumin Wang, and Phil Blunsom. Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744*, 2016. 7.1
- [179] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction for rationale generation: Learning to solve and explain algebraic word problems. In *Association for Computational Linguistics (ACL)*, 2017. 7.1
- [180] Chang Liu, Xinyun Chen, Eui Chul Shin, Mingcheng Chen, and Dawn Song. Latent attention for if-then program synthesis. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4574–4582. Curran Associates, Inc., 2016. URL <http://papers.nips.cc/paper/6284-latent-attention-for-if-then-program-synthesis.pdf>. 7.1

- [181] Hanxiao Liu, Wanli Ma, Yiming Yang, and Jaime Carbonell. Learning concept graphs from online educational data. *Journal of Artificial Intelligence Research*, 55:1059–1090, 2016. 2, 7.1
- [182] Rui Liu, Junjie Hu, Wei Wei, Zi Yang, and Eric Nyberg. Structural embedding of syntactic trees for machine comprehension. *CoRR*, abs/1703.00572, 2017. 2
- [183] Jan Lukasiewicz. O logice trójwartościowej. *Studia Filozoficzne*, 270(5), 1988. 7.3.1
- [184] Bill MacCartney, Michel Galley, and Christopher D Manning. A phrase-based alignment model for natural language inference. In *Proceedings of the conference on empirical methods in natural language processing*, pages 802–811, 2008. 4, 4.2
- [185] William C Mann and Sandra A Thompson. {Rhetorical Structure Theory: Toward a functional theory of text organisation}. *Text*, 3(8):234–281, 1988. 4, 4.2, 1, 5.2, 6.3, 7.3.4
- [186] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014. 7.4, 7.4.1
- [187] Daniel Marcu. *The theory and practice of discourse parsing and summarization*. MIT Press, 2000. 7.3.4, ??
- [188] Brent Martin, Antonija Mitrovic, Kenneth R Koedinger, and Santosh Mathan. Evaluating and improving adaptive educational systems with learning curves. *User Modeling and User-Adapted Interaction*, 21(3):249–283, 2011. 2
- [189] Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai, and Noriko H Arai. Semantic parsing of pre-university math problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2131–2141, 2017. 2
- [190] Cynthia Matuszek, Dieter Fox, and Karl Koscher. Following directions using statistical machine translation. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 251–258. IEEE, 2010. 7.1
- [191] Andrew N Meltzoff. Understanding the intentions of others: re-enactment of intended acts by 18-month-old children. *Developmental psychology*, 31(5):838, 1995. 7.4.1
- [192] Andrew N. Meltzoff and M. Keith Moore. Imitation of facial and manual gestures by human neonates. *Science*, 198(4312):75–78, 1977. ISSN 0036-8075. doi: 10.1126/science.198.4312.75. URL <http://science.sciencemag.org/content/198/4312/75>. 7.4.1
- [193] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736, 2016. 8.4
- [194] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/N13-1090>. 5.2

- [195] P Hartley Millar. On the point of the imitation game. *Mind*, 82(328):595–597, 1973. 1
- [196] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 236–244, 2008. 5.2, 6.3
- [197] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Tanti 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, January 25-30, 2015, Austin, Texas, USA.*, pages 2302–2310, 2015. 7.1
- [198] Sparsh Mittal and Ankush Mittal. Versatile question answering systems: seeing in synthesis. *International Journal of Intelligent Information and Database Systems*, 5(2):119–142, 2011. 4
- [199] Dan Moldovan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)*, 21(2):133–154, 2003. 4.1.2
- [200] Hankyu Moon, Rama Chellappa, and Azriel Rosenfeld. Optimal edge-based shape detection. *IEEE Transactions on Image Processing*, 11(11):1209–1227, 2002. 7.1
- [201] James H Moor. An analysis of the turing test. *Philosophical Studies*, 30(4):249–257, 1976. 1
- [202] Karthik Narasimhan and Regina Barzilay. Machine comprehension with discourse relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1253–1262, 2015. 2, 5.3
- [203] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268, 2016. URL <http://arxiv.org/abs/1611.09268>. 2
- [204] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016. 8.4
- [205] Gordon Novak. Diagrams for solving physical problems. *Diagrammatic reasoning: Cognitive and computational perspectives*, 1995. 7.1
- [206] Gordon S. Novak and William C. Bulko. Understanding natural language with diagrams. In *AAAI*, pages 465–470, 1990. 7.1
- [207] Joseph D Novak. *Learning, creating, and using knowledge: Concept maps as facilitative tools in schools and corporations*. Routledge, 2010. 2
- [208] Lawrence O’Gorman and Rangachar Kasturi. *Document image analysis*, volume 39. Cite-

seer, 1995. 7.1

- [209] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975. 7.2, 7.5.1
- [210] Saurabh Pal. Mining educational data using classification to decrease dropout rate of students. *CoRR*, abs/1206.3078, 2012. URL <http://arxiv.org/abs/1206.3078>. 2
- [211] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016. 2
- [212] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, 2002. 8.4
- [213] Terence Parsons. *Events in the Semantics of English*, volume 5. In MIT Press, 1990. 5
- [214] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962. 7.5.1
- [215] Fuchun Peng and Andrew McCallum. Information extraction from research papers using conditional random fields. *Information processing & management*, 42(4):963–979, 2006. 7.1
- [216] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 8.4
- [217] Gail B Peterson. A day of great illumination: Bf skinner’s discovery of shaping. *Journal of the Experimental Analysis of Behavior*, 82(3):317–328, 2004. 13
- [218] Emmanouil Platanios, Hoifung Poon, Tom M Mitchell, and Eric J Horvitz. Estimating accuracy from unlabeled data: A probabilistic logic approach. In *Advances in Neural Information Processing Systems*, pages 4361–4370, 2017. 7.3.1
- [219] J Pont-Tuset, P Arbelaez, J Barron, F Marques, and J Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE TPAMI*, 2016. 7.5.1
- [220] Hoifung Poon and Pedro Domingos. Unsupervised semantic parsing. In *EMNLP*, 2009. 7.1
- [221] Marc Prensky. Digital game-based learning. *Computers in Entertainment (CIE)*, 1(1): 21–21, 2003. 2
- [222] Richard L Purtil. Beating the imitation game. *Mind*, 80(318):290–294, 1971. 1
- [223] Chris Quirk, Raymond J. Mooney, and Michel Galley. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Process-*

- ing, *ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 878–888, 2015. 7.1
- [224] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL <http://arxiv.org/abs/1606.05250>. 2, 8, 8.4
- [225] Pekka Rantalankila, Juho Kannala, and Esa Rahtu. Generating object segmentation proposals using global and local search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2417–2424, 2014. 7.2, 7.5.1
- [226] Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale semantic parsing without question-answer pairs. *TACL*, 2(Oct), 2014. 7.1
- [227] Matthew Richardson, J.C. Christopher Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, 2013. 1, 2, 4, 4, 4.2, 5.3, 6.4, 8
- [228] Ellen Riloff and Michael Thelen. A rule-based question answering system for reading comprehension tests. In *Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding systems-Volume 6*, pages 13–19. Association for Computational Linguistics, 2000. 2
- [229] Ellen Riloff, Janyce Wiebe, and Theresa Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 25–32. Association for Computational Linguistics, 2003. 8.3, 13
- [230] Murray Rosenblatt et al. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956. 7.5.1
- [231] Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of EMNLP*, 2015. 2
- [232] Subhro Roy, Tim Vieira, and Dan Roth. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13, 2015. 2
- [233] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015. 8.4
- [234] Mrinmaya Sachan and Eric Xing. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 453–463, 2016. 2
- [235] Mrinmaya Sachan and Eric Xing. Self-training for jointly learning to ask and answer questions. In *Proceedings of HLT-NAACL*, 2018. 2
- [236] Mrinmaya Sachan and Eric P. Xing. Machine comprehension using rich semantic representations. In *Proceedings of ACL*, 2016. 2
- [237] Mrinmaya Sachan and Eric P. Xing. Learning to solve geometry problems from natural language demonstrations in textbooks. In *Proceedings of the 6th Joint Conference on*

*Lexical and Computational Semantics, *SEM @ACM 2017, Vancouver, Canada, August 3-4, 2017*, pages 251–261, 2017. 7.4.1

- [238] Mrinmaya Sachan and Eric P. Xing. Parsing to programs: A framework for situated qa. In *Proceedings of KDD*, 2018. 7
- [239] Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. Learning answer-entailing structures for machine comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2015. 2, 5.3
- [240] Mrinmaya Sachan, Eduard Hovy, and Eric P Xing. An active learning approach to coreference resolution. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. 9
- [241] Mrinmaya Sachan, Kumar Dubey, and Eric P. Xing. Science question answering using instructional materials. In *Proceedings of ACL*, 2016. 1, 2
- [242] Mrinmaya Sachan, Kumar Dubey, and Eric Xing. From textbooks to knowledge: A case study in harvesting axiomatic knowledge from textbooks to solve geometry problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 773–784, 2017. 7
- [243] Mrinmaya Sachan, Avinava Dubey, Jaime G. Carbonell, Eduard Hovy, Dan Roth, Tom Mitchell, and Eric P. Xing. Discourse in multimedia: A case study in information extraction. In *Under Submission*, 2018. 7
- [244] Mrinmaya Sachan, Avinava Dubey, Dan Roth, Tom Mitchell, and Eric P. Xing. Learning pipelines with limited data and domain knowledge: A study in parsing physics problems. In *Proceedings of NIPS*, 2018. 7
- [245] Mrinmaya Sachan, Minjoon Seo, Hannaneh Hajishirzi, and Eric P. Xing. Parsing to programs: A framework for situated question answering. In *Under Submission*, 2018. 7
- [246] Ozair Saleem and Seemab Latif. Information extraction from research papers by data integration and data validation from multiple header extraction sources. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, pages 177–180, 2012. 7.1
- [247] M. Sammons, V. Vydiswaran, T. Vieira, N. Johri, M. Chang, D. Goldwasser, V. Srikumar, G. Kundu, Y. Tu, K. Small, J. Rule, Q. Do, and D. Roth. Relation alignment for textual entailment recognition. In *TAC*, 2009. 4
- [248] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*, 2016. 8.4
- [249] Ayse Pinar Saygin, Ilyas Cicekli, and Varol Akman. Turing test: 50 years later. *Minds and machines*, 10(4):463–518, 2000. 1
- [250] Stefan Schaal. Learning from demonstration. In M. I. Jordan and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 1040–1046. MIT Press, 1997. 7.4.1
- [251] Doris Schattschneider and James King. *Geometry Turned On: Dynamic Software in Learning, Teaching, and Research*. Mathematical Association of America Notes, 1997.

- [252] John R Searle. Minds, brains, and programs. *THE BEHAVIORAL AND BRAIN SCIENCES*, 3:417–457, 1980. 1
- [253] Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. Solving geometry problems: combining text and diagram interpretation. In *Proceedings of EMNLP*, 2015. 1, 2, 7.3.2, 7.3.4, 7.4, 7.4.1, 7.4.1, 7.4.1, 7.4.1, 7.5.4
- [254] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016. 2
- [255] Parantu K Shah, Carolina Perez-Iratxeta, Peer Bork, and Miguel A Andrade. Information extraction from full text scientific articles: Where are the keywords? *BMC bioinformatics*, 4(1):20, 2003. 7.1
- [256] Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Marco A Valenzuela-Escárcega, Peter Clark, and Michael Hammond. Tell me why: Using question answering as distant supervision for answer justification. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 69–79, 2017. 2
- [257] Jude W Shavlik. Combining symbolic and neural learning. *Machine Learning*, 14(3): 321–331, 1994. 1
- [258] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM, 2014. 8.4
- [259] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. *CoRR*, abs/1609.05284, 2016. URL <http://arxiv.org/abs/1609.05284>. 2
- [260] Stuart M. Shieber. Lessons from a restricted turing test. *Commun. ACM*, 37(6):70–78, June 1994. ISSN 0001-0782. doi: 10.1145/175208.175217. URL <http://doi.acm.org/10.1145/175208.175217>. 1
- [261] Nobuyuki Shimizu and Andrew R. Haas. Learning to follow navigational route instructions. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 1488–1493, 2009. 7.1
- [262] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Kumar Divvala, and Ali Farhadi. Figureseer: Parsing result-figures in research papers. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII*, pages 664–680, 2016. 7.5.1
- [263] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 7.2, 7.5.1
- [264] Burrhus F Skinner. Reinforcement today. *American Psychologist*, 13(3):94, 1958. 13
- [265] Ellery Smith, Nicola Greco, Matko Bosnjak, and Andreas Vlachos. A strong lexical matching method for the machine comprehension test. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1693–1698. Asso-

ciation for Computational Linguistics, 2015. 2, 5.3

- [266] Radu Soricut and Daniel Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics, 2003. 7.3.4, ??, ??, ??, 7.4.2
- [267] Rohini K Srihari. Computational models for integrating linguistic and visual information: A survey. *Artificial Intelligence Review*, 8(5-6), 1994. 7.1
- [268] Shashank Srivastava and Dirk Hovy. A walk-based semantically enriched tree kernel over distributed word representations. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1411–1416, 2013. 4.2, 5.2, 6.3
- [269] Constance A Steinkuehler. Learning in massively multiplayer online games. In *Proceedings of the 6th international conference on Learning sciences*, pages 521–528. International Society of the Learning Sciences, 2004. 2
- [270] Asher Stern and Ido Dagan. Biutee: A modular open-source system for recognizing textual entailment. In *Proceedings of the ACL 2012 System Demonstrations*, pages 73–78, 2012. 8, 6.4
- [271] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002. 7.4
- [272] Arafat Md Sultan, Steven Bethard, and Tamara Sumner. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association of Computational Linguistics – Volume 2, Issue 1*, pages 219–230, 2014. 4
- [273] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 8.2
- [274] Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*, 2017. (document), 2, 8.2, 8.4, 8.2, 8.4, 8.3
- [275] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordani, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830, 2016. 2
- [276] Alan M Turing. Computing machinery and intelligence. *Mind*, pages 433–460, 1950. 1, 2
- [277] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2): 154–171, 2013. 7.2, 7.5.1
- [278] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015. 8.2
- [279] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A

- neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015. 8.2
- [280] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004. 7.2, 7.5.1
- [281] Adam Vogel and Daniel Jurafsky. Learning to follow navigational directions. In *ACL*, 2010. 7.1
- [282] Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. Take-lab: Systems for measuring semantic text similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 441–448, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/S12-1060>. 7.4.1
- [283] Henning Wachsmuth. *Text analysis pipelines: towards ad-hoc large-scale text mining*, volume 9383. Springer, 2015. 7.3.1
- [284] Di Wang and Eric Nyberg. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 707–712, 2015. 5.3
- [285] Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 700–706, 2015. 2
- [286] Shuting Wang, Chen Liang, Zhaohui Wu, Kyle Williams, Bart Pursel, Benjamin Brautigam, Sherwyn Saul, Hannah Williams, Kyle Bowen, and C Lee Giles. Concept hierarchy extraction from textbooks. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, pages 147–156. ACM, 2015. 7.1
- [287] Shuting Wang, Alexander Ororbia, Zhaohui Wu, Kyle Williams, Chen Liang, Bart Pursel, and C Lee Giles. Using prerequisites to extract concept maps from textbooks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 317–326. ACM, 2016. 7.1
- [288] Tong Wang, Xingdi Yuan, and Adam Trischler. A joint model for question answering and question generation. *CoRR*, abs/1706.01450, 2017. 2
- [289] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January 1966. ISSN 0001-0782. doi: 10.1145/365153.365168. URL <http://doi.acm.org/10.1145/365153.365168>. 2
- [290] Joseph Weizenbaum. *Computer power and human reason: From judgment to calculation*. WH Freeman & Co, 1976. 1
- [291] Ben Wellner, Lisa Ferro, Warren Greiff, and Lynette Hirschman. Reading comprehension

- tests for computer-based understanding evaluation. *Natural Language Engineering*, 12 (4):305–334, dec 2006. ISSN 1351-3249. 2
- [292] Wu Wen-Tsun. Basic principles of mechanical theorem proving in elementary geometries. *Journal of automated Reasoning*, 2(3):221–252, 1986. 7.1
- [293] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015. 4, 4.2, 12, 5.3
- [294] Sean Wilson and Jacques D Fleuriot. Combining dynamic geometry, automated geometry theorem proving and diagrammatic proofs. In *Workshop on User Interfaces for Theorem Proving (UITP)*, 2005. 7.1
- [295] Jian Wu, Jason Killian, Huaiyu Yang, Kyle Williams, Sagnik Ray Choudhury, Suppawong Tuarob, Cornelia Caragea, and C. Lee Giles. Pdfmef: A multi-entity knowledge extraction framework for scholarly documents and semantic search. In *Proceedings of the 8th International Conference on Knowledge Capture, K-CAP 2015*, 2015. 7.1
- [296] Surjeet Kumar Yadav, Brijesh Bharadwaj, and Saurabh Pal. Mining education data to predict student’s retention: a comparative study. *arXiv preprint arXiv:1203.2987*, 2012. 2
- [297] Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. Type- and content-driven synthesis of SQL queries from natural language. *CoRR*, abs/1702.01168, 2017. 7.1
- [298] Yi Yang, Scott Wen-tau Yih, and Chris Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, September 2015. 8.4, 8.4
- [299] Yiming Yang, Hanxiao Liu, Jaime G. Carbonell, and Wanli Ma. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 159–168, 2015. 7.1
- [300] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. A lightweight and high performance monolingual word aligner. In *ACL (2)*, pages 702–707, 2013. 4, 6.4, ??
- [301] Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Semi-markov phrase-based monolingual alignment. In *Proceedings of EMNLP*, 2013. 4.2
- [302] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995. 8.3, 13
- [303] Wentau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013. 4, 4.1.1, 4.2
- [304] Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada, July 2017. URL <https://arxiv.org/abs/1704.01696>. 7.1

- [305] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*, 2015. 2, 8.4
- [306] Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. Attention-based convolutional neural network for machine comprehension. *arXiv preprint arXiv:1602.04341*, 2016. 5.3
- [307] Chun-Nam Yu and T. Joachims. Learning structural svms with latent variables. In *Proceedings of International Conference on Machine Learning (ICML)*, 2009. 3.4, 3.4.1
- [308] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014. 8
- [309] A. L. Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Comput.*, 2003. 3.4.1, 4.1.1
- [310] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014. 8.1, 8.2
- [311] John M. Zelle and Raymond J. Mooney. Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the 11th National Conference on Artificial Intelligence. Washington, DC, USA, July 11-15, 1993.*, pages 817–822, 1993. 1, 7.1
- [312] John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996. 1, 7.1
- [313] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, 2005. 7.1
- [314] Dell Zhang and Wee Sun Lee. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32. ACM, 2003. 4.2
- [315] Xin Zhang and Fuchun Sun. Circle text expansion as low-rank textures. In *ICDAR*, pages 202–206. IEEE, 2011. 7.1
- [316] Ziming Zhang, Jonathan Warrell, and Philip HS Torr. Proposal generation for object detection using cascaded ranking svms. In *Computer Vision and Pattern Recognition (CVPR), 2011*, pages 1497–1504. IEEE, 2011. 7.2, 7.5.1
- [317] Jun Zhu, Ning Chen, and Eric P Xing. Infinite latent svm for classification and multi-task learning. In *Advances in neural information processing systems*, pages 1620–1628, 2011. 4
- [318] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. 1, 8
- [319] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014. 7.2, 7.5.1, 7.5.4