Towards Literate Artificial Intelligence

Mrinmaya Sachan MRINMAYS@CS.CMU.EDU
Machine Learning Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

Standardized tests are often used to test students as they progress in the formal education system. These tests are widely available and measurable with clear evaluation procedures and metrics. Hence, these can serve as good tests for AI. We propose approaches for solving some of these tests. We broadly categorize these tests into two categories: open domain question answering tests such as reading comprehensions and elementary school science tests, and closed domain question answering tests such as intermediate or advanced math and science tests. We present alignment based approach with multi-task learning for the former. For closed domain tests, we propose a parsing to programs approach which can be seen as a natural language interface to expert systems. We also describe approaches for question generation based on instructional material in both open domain as well as closed domain settings. Finally, we show that we can improve both the question answering and question generation models by learning them jointly. This mechanism also allows us to leverage cheap unlabelled data for learning the two models. Our work can be easily applied for the social good in the education domain. We perform studies on human subjects who found our approaches useful as assistive tools in education.

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 (Turing, 1950). 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 (Colby, 1975; Shieber, 1994; Weizenbaum, 1976) 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 classroom and 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 (French, 1996; Levesque, 2010; Millar, 1973; Moor, 1976; Purtill, 1971; Saygin et al., 2000; Searle, 1980) 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 AI 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 (Clark, 2015). These include tests on understanding passages and stories and answering questions about them (Richardson et al., 2013), math and science question answering using instructional material (Kushman et al., 2014; Sachan et al., 2016; Seo et al., 2015), visual question answering (Antol et al., 2015), 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 (Clark, 2015). For example, spatial/kinematic reasoning, 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 (section 3.2.1 and 3.2.2), answering science questions using instructional material (section 3.2.3), math and science questions in geometry (section 4.2.1) and Newtonian physics (section 4.2.2). Such problems frequent the curriculum of students and also appear in standardised tests such as the *SAT* or *Advanced Placement* college level courses.

The reading comprehension tests require the system to answer a set of multiple-choice comprehension questions based on a text passage. We show (see Section 3.2.1) 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 (Blunsom and Cohn, 2006). 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. We extend this approach by using various types of language representations – bag of words, syntactic parses and the abstract meaning representation (Banarescu et al., 2013). Our results show that richer language representations lead to improved performance. We also apply this approach in the *Allen Institute of AI*'s Kaggle challenge for 8^{th} grade science question answering. Here, we re-describe the *answer-entailing* structure for the task 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. In this case, we also used external domain-specific knowledge resources such as science dictionaries, study guides and semi-structured tables to further refine the answer-entailing structure and show significant improvements over a number of lexical and neural network baselines which have been shown to do well on this task.

Advanced math and science curriculum offer us problems which are quite challenging and require significant domain knowledge and reasoning capabilities. Yet, the existence of well-defined axioms and laws of how math and science works constrains the reasoning space in a manageable way. For example, geometry problems typically describe a scenario with a small number of primitives (lines, quadrilaterals, circles, etc) and require the application of the laws and axioms of geometry (Pythagoras theorem, alternating angles, etc) to solve them. Similarly, Newtonian Physics problems describe a scenario with some objects (pulleys, blocks, etc.) and require the application of various laws of Physics (Newtons laws, conservation laws, etc.) to solve them. The subject knowledge can often be extracted from textbooks and appropriate representations can be learned by formulating appropriate learning formulations similar to those used in semantic parsing. To solve these problems, we propose a *parsing to programs* framework which combines ideas from semantic parsing (Kate et al., 2005; Zelle and Mooney, 1993; 1996) and probabilistic programming (Goodman and Stuhlmüller, 2014). 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. This can be seen as a natural language interface to expert systems (Jackson, 1986). We use this technique to build two systems – the first answers SAT style geometry questions and the second answers Newtonian physics problems. 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 examinations held in 1998 and 2012.

In the second part of this thesis, we look at the inverse of the question answering problem – question asking. Here, we build structured approaches which generate questions from instructional material. This would be useful for building instructional material and practice question banks for students. We again describe two variants of our question asking setup (a) for generating open-domain reading comprehension questions given a set of text passages, and (b) generating math and science questions given some domain theory in the form of programs. Furthermore, we show that the question answering and question asking problems are closely related and we propose a self-training method for jointly learning to ask as well as answer questions, leveraging unlabeled text along with labeled question answer pairs for learning.

Wherever possible, we also show that our techniques for answering and asking questions using instructional material can be useful for the social good in education. 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 with human subjects (students), we (plan to) show that the subjects find our tools for answering as well as asking questions useful. These studies give us hope that our solution can be used as assistive tools for helping students learn better.

To summarize, this thesis hopes to make the following contributions:

- We propose standardized tests as new tests for AI. Standardized tests are already being used to test skill set of humans, and due to their ready availability and challenging degree of intelligence required to answer them, they can serve as challenging tests for development of AI systems in the future.
- We tackle a number of standardized tests such as reading comprehensions, science and math question answering and propose a suite of carefully engineered solutions for them. Our experiments show that we can attain near-human performance on many of these tests. We also lay down a number of outstanding challenges for improving our solutions.
- We further address the task of generating questions for various standardized tests, thus creating educational content and practice material for students.
- We show that we can jointly model the question generation

and question answering tasks in the aforementioned applications by self-training. This allows us to utilize unlabeled data in conjunction with labeled data to build better question answering and question generation systems in these low-resource domains.

- Finally, we use our solutions to develop various applications for student learning. We use them to perform some small-scale user studies with students and find that the students indeed find them useful.

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 (Turing, 1950) 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 behaviours, 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 (Weizenbaum, 1966) and PARRY (Colby, 1975) as two early attempts at cracking the test. While the final nature of the results has also been debated, it can 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 learn in school and 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 provide a graded measure of machine intelligence with respect to humans. Moreover, 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 (Charniak, 1972), proposed a *background* model to answer questions about children's stories. Also, famously, Hirschman et al. (1999) 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 Grois and Wilkins (2005); Harabagiu et al. (2003); Wellner et al. (2006). Riloff and Thelen (2000) 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. (2001) 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. (2003) 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.

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 Richardson et al. (2013) who also crowd-source a 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 (Hill et al., 2015) from FAIR, CNN/Daily Mail (Hermann et al., 2015) released by Google DeepMind, Stanford Question Answering Dataset or SQuAD (Rajpurkar et al., 2016a), LAnguage Modeling Broadened to Account for Discourse Aspects or LAMBADA dataset (Paperno et al., 2016) from University of Trento and University of Amsterdam, QuizBowl questions (Iyyer et al., 2014) from University of Maryland and University of Colorado, NewsQA dataset (Trischler et al., 2016) from Maluuba Research, and MS MARCO (Nguyen et al., 2016a) from Microsoft. In-

fact, a number of companies and startups such as Allen Institute for AI [1] and Microsoft Maluuba[2] focus on the idea of building solvers for various standardized tests. Various lexical approaches (Richardson et al., 2013; Smith et al., 2015), structured prediction approaches (Narasimhan and Barzilay, 2015; Sachan and Xing, 2016; Sachan et al., 2015; 2016; Wang et al., 2015), and neural network approaches (Chen et al., 2017; Hu et al., 2017; Liu et al., 2017; Seo et al., 2016; Shen et al., 2016; Yin et al., 2015) 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 (Khashabi et al., 2016; Khot et al., 2015; 2017; Sharp et al., 2017), solving algebra word problems (Hopkins et al., 2017; Hosseini et al., 2014; Kushman et al., 2014; Matsuzaki et al., 2017; Roy and Roth, 2015; Roy et al., 2015) and SAT style geometry problems (Seo et al., 2015). 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 (Arai and Matsuzaki, 2014). 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 refered to as question asking in this thesis, has been pursued before in a number of works such as Heilman (2011) which focuses on generating questions based on given sources of knowledge. More recently, Du et al. (2017); Tang et al. (2017) built on the sequence to sequence paradigm of deep learning to generate questions conditioned on an answer sentence. We build upon the relationship between question answering and question asking and propose a joint training framework for the two.

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 (Aher and Lobo, 2013; Drachsler et al., 2015; Gordon, 2000; Hwang, 2003; Liu et al., 2016; Novak, 2010), to track student knowledge and recommend next steps using adaptive learning systems or game-based learning (Brusilovsky and Peylo, 2003; Ebner and Holzinger, 2007; Martin et al., 2011; Prensky, 2003; Steinkuehler, 2004), 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 (Ala-Mutka, 2005; Attali and Burstein, 2004; Bennett and Bejar, 1998; Burstein et al., 2001), and predicting enrollment, placements and boosting retention (Lin, 2012; Pal, 2012; Yadav et al., 2012). In our work, we show how question answering and question generation can be used to assist student learning by performing some small-scale user studies with students studying these subjects.

[1] http://allenai.org/

[2] http://www.maluuba.com/

3. Open-domain Question Answering from Instructional Material

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.

3.1. Methodology

Let us formalize the machine comprehension setup. For each question $q_i \in Q$, let $A_i = \{a_{i1}, \ldots, a_{im}\}$ 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 $f : (q, \mathscr{K}) \rightarrow a$ that, given a question q_i and background knowledge $\mathscr{K}$ (texts/resources required to answer the question), outputs an answer $\hat{a}_i \in A_i$. We consider a scoring function $S_{\mathbf{w}}(q, a; \mathscr{K})$ (with model parameters $\mathbf{w}$) and a prediction rule $f_{\mathbf{w}}(q_i) = \hat{a}_i = \underset{a_{ij} \in A_i}{\arg\max}\, S_{\mathbf{w}}(q_i, a_{ij}; \mathscr{K})$. 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:

$$\min_{\mathbf{w}} \sum_{q_i \in Q} L_{\mathbf{w}}(a_i^*, f_{\mathbf{w}}(q_i); \mathscr{K}) + \Omega(\mathbf{w}) \tag{1}$$

3.1.1. Alignment based approach to machine comprehension

We cast machine comprehension 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 $\mathbf{h}_i = \{h_{i1}, \ldots, h_{im}\}$ 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.

We use latent structures, called *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 (Blunsom and Cohn, 2006), RTE (MacCartney et al., 2008), paraphrase (Yao et al., 2013), QA (Yih et al., 2013), 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.

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 learned 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 $h_j \in \mathbf{h} \setminus h^*$ for some slack ξ_i. This can be formulated as the following optimization problem:

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

It is intuitive to use 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) (Yuille and Rangarajan, 2003). 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.

3.1.2. 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 (Ferrucci, 2012; Moldovan et al., 2003) 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 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 learn classifications of sub-tasks and then extend our LSSVM to multi-task settings. 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 $Score_{\mathbf{w}_0,\mathbf{v}_s}(h,\mathbf{z}) = (\mathbf{w}_0 + \mathbf{v}_s)^T\psi(h,\mathbf{z})$.

We extend a trick that Evgeniou and Pontil (2004) used for linear SVM to reformulate this problem into an objective that looks like

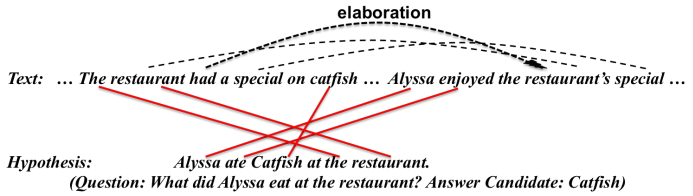


Figure 1. The *answer-entailing structure* for an example from MCTest500 dataset. The question and answer candidate are combined to generate a hypothesis sentence. Then latent alignments are found between the hypothesis and the appropriate snippets in the text. The solid red lines show the word alignments 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.

(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:

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

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 $\mathbf{w} = (\sqrt{\mu}\mathbf{w}_0, \mathbf{v}_1, \ldots, \mathbf{v}_S)$. 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 (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.

3.2. Applications

3.2.1. Answering Reading Comprehension Questions

Reading comprehensions tests evaluate the reader's ability to read, process, understand the meaning of a given piece of test and answer questions about it. 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.

Answer-entailing structures: Figure 1 shows an example answer-entailing structure in our setting. The latent structure includes selection of a snippet in the text selected as a proxy for the evidence followed by one-to-one word alignment from the hypothesis to the snippet. We only consider snippets of 1, 2 and 3 sentences as most questions can be answered by 3 sentences in the passage. In our example, 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

		Single	Multiple	All
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
+MTL	QAClassification	66.18	67.37	66.83
+MTL	TaskClassification	67.65	**67.99**	**67.83**

Table 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 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.

constituents should be compared to each other. These alignments can then be used to generate more effective features. We refer the reader to the paper (Sachan et al., 2015) or the detailed thesis proposal for details about the feature set, various choices for task selection for the multi-task setup, baselines, and evaluation. We use a suite of lexical and neural network baselines to compare against our work. 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 classification[3] from (Li and Roth, 2002). 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 (Weston et al., 2015). 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.

We compare multiple variants of our LSSVM 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 NDCG_4 (Järvelin and Kekäläinen, 2002). Unlike classification accuracy which evaluates if the prediction is correct or not, NDCG_4, being a measure of ranking quality, evaluates the position of the correct answer in our predicted ranking.

Table 1 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.

We can also 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 better separation between the various strategies needed to learn an overarching list of subtasks required to solve the machine comprehension task. 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.

3.2.2. 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 model, 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 "sembanking"). AMR, a new semantic representation in standard neo-Davidsonian (Davidson, 1969; Parsons, 1990) 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.

We also used the AMR representation for 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 2 for an example). Then, our unified max-margin model jointly learns the latent structure (subgraph selection and alignment) and the QA model.

The 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 en-

[3] http://cogcomp.cs.illinois.edu/Data/QA/QC/

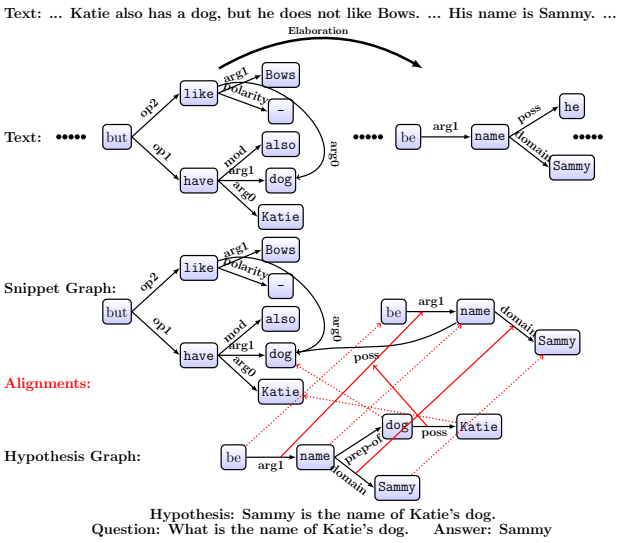


Figure 2. Example latent *answer-entailing* structure from the MCTest dataset. The question and answer candidate are combined to generate a hypothesis. This hypothesis is AMR parsed to construct a hypothesis meaning representation graph after some post-processing (subsubsection 3.2.2). Similar processing 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).

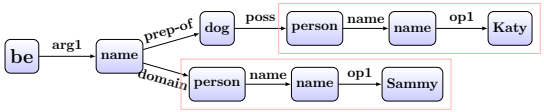


Figure 3. AMR parse for the hypothesis in Figure 2. The person nodes are merged to achieve the hypothesis meaning representation graph.

tity ("person"). For example, the hypothesis meaning representation graph in Figure 2 was achieved by merging the AMR parse shown in Figure 3.

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 structures[4] 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 2.

[4]Rhetorical structure theory (Mann and Thompson, 1988) tells us that sentences with discourse relations are related to each other. Previous works in QA (Jansen et al., 2014) 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.

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: $\mathbf{z}_s = \arg\max_{\mathbf{z}_s} \mathbf{w}_s^T \mathbf{f}(G', G, \mathbf{t}, h, \mathbf{z}_s)$. 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 constrains 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 $(v^1_{e'}, v^2_{e'})$ that form the end points of e' must also be aligned to vertices (v^1_e, v^2_e) 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 $v^1_{e'}$ align with v^1_e and $v^2_{e'}$ align with v^2_e. If exactly one of the relations is an inverse relation (case 2), we enforce that $v^1_{e'}$ align with v^2_e and $v^2_{e'}$ align with v^1_e. Hence, we introduce the following constraints:

$$z_{e'e} \leq z_{v^1_{e'} v^1_e} \text{ and } z_{e'e} \leq z_{v^2_{e'} v^2_e} \quad \forall e'.e \text{ in case 1}$$
$$z_{e'e} \leq z_{v^1_{e'} v^2_e} \text{ and } z_{e'e} \leq z_{v^2_{e'} v^1_e} \quad \forall e'.e \text{ in case 2}$$

		Single	Multiple	All
AMR	Subgraph	67.28	65.24	66.16
	Subgraph+Negation	69.48	66.46	67.83
+MTL	QClassification	70.59	67.99	69.17
	QAClassification	71.32	68.29	69.67
	TaskClassification	**72.05**	**68.90**	**70.33**

Table 2. 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 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. Baseline results are shown in table 1

We again use MCTest-500 dataset (Richardson et al., 2013) and the same evaluation setup as before. We refer the readers to Sachan and Xing (2016) for details about the feature set. We compare our AMR subgraph containment approach where we consider our modifications for negation and multi-task learning as well in Table 2. 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 significantly, 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 subtasks for machine comprehension defined in Weston et al. (2015) does better than QAClassification that learns the question answer classification. QAClassification in turn performs better than QClassification that learns the question classification only. These results, together, provide validation for our approach of subgraph matching over meaning representation graphs, and the incorporation of negation and multi-task learning.

3.2.3. Science Question Answering from Instructional Materials

We also use our approach in the task of answering multiple-choice elementary science tests (Clark, 2015) using the science curriculum of the student and other domain specific knowledge resources. 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. In this case, our approach learns latent *answer-entailing structures* that align question-answers with appropriate snippets in the curriculum. 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 4). These domain-specific knowledge resources can be very useful forms of knowledge representation as shown in previous works (Clark et al., 2016).

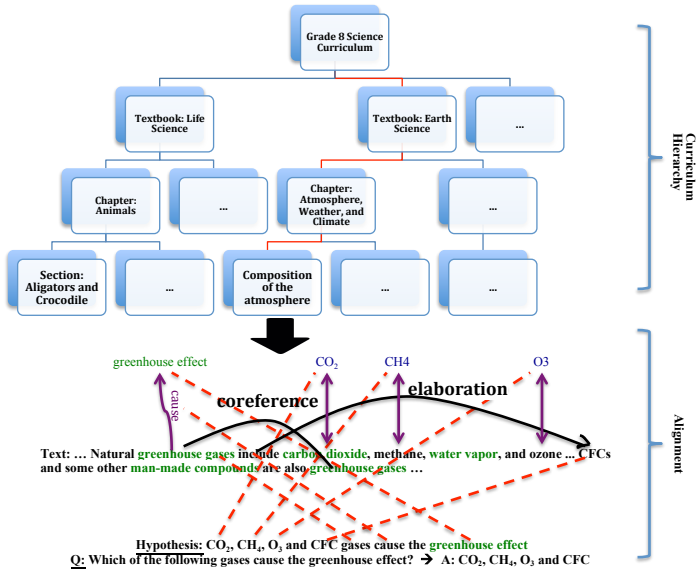


Figure 4. An example *answer-entailing structure* for science question answering. The answer-entailing structure consists of selecting a particular textbook from the curriculum, picking a chapter in the textbook, picking a section in the chapter, picking sentences in the section and then aligning words/mwe's 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 = 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.

In this application, 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 (Ferrucci, 2012) 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 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 knowl-

edge 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.

We used a set of 8^{th} grade science questions released as the training set in the Allen AI Science Challenge[5] 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 8^{th} 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. 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. We compare our work to a number of lexical and neural network baselines.

We refer the interested readers to Sachan et al. (2016) for details about the feature set. We compare variants of our method 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 5 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 above. 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.

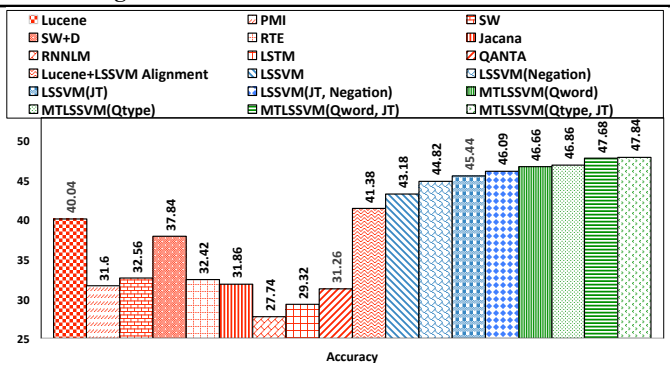
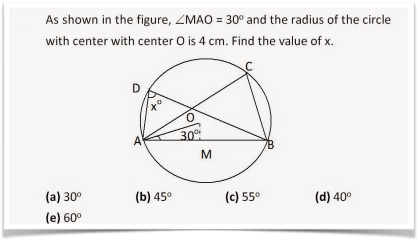


Figure 5. Variations of our method vs several baselines on the Science QA dataset. Differences between the baselines and LSSVMs, the improvement due to negation, the improvements due to multi-task learning and joint-learning are significant ($p < 0.05$) using the two-tailed paired T-test.



Text Description:

measure(∠MAO, 30°)
isCircle(O)
radius(O, 4 cm)
?x

Diagram:

liesOn(A, circle O), liesOn(B, circle O),
liesOn(C, circle O), liesOn(D, circle O)
isLine(AB), isLine(BC), isLine(CA), isLine(BD), isLine(DA)
isTriangle(ABC), isTriangle(ABD), isTriangle(AOM)
measure(∠ADB, x), measure(∠MAO, 30°)
measure(∠AMO, 90°)
…

Figure 6. An example SAT style geometry problem with its corresponding text and diagram parse. Each extracted relation is weighted i.e. it carries a weight corresponding to the model confidence for its prediction. However, we do not shown that here for simplicity.

4. Closed Domain Question Answering

In section 3, we looked at open domain question answering tasks such as reading comprehensions or science question answering. However, a number of problems, such as SAT style math and science problems are very domain specific. These domains are often accompanied by well-defined axioms and laws of these domains. For example, geometry questions typically involve a number of geometry primitives (lines, quadrilaterals, circles, etc) and require students to use axioms and theorems of geometry (Pythagoras theorem, alternating angles, etc) to solve them. Similarly, Newtonian Physics problems typically describe a scene comprising of primitive objects (pulleys, blocks, etc.) and require the application of laws of Physics (Newtons laws, conservation laws, etc.) to solve them. These closed domain questions can often have a reliable automatic semantic analysis as they have a formal logical basis and can be mapped to a formal language such as first-order logic by linguistic processing approaches such as semantic parsing (Kate et al., 2005; Zelle and Mooney, 1993; 1996). This formal representation of the question can be used as an input to a domain-specific expert system like programmatic solver to solve them.

We take two examples of closed domain question answering tasks. The first is the task of answering SAT style geometry problems. SAT geometry tests the student's knowledge of Euclidean geometry in its classical sense, including the study of points, lines, planes, angles, triangles, congruence, similarity, solid figures, circles, and analytical geometry. A typical geometry problem is provided in Figure 6. A geometry question typically in-

[5] https://www.kaggle.com/c/the-allen-ai-science-challenge/

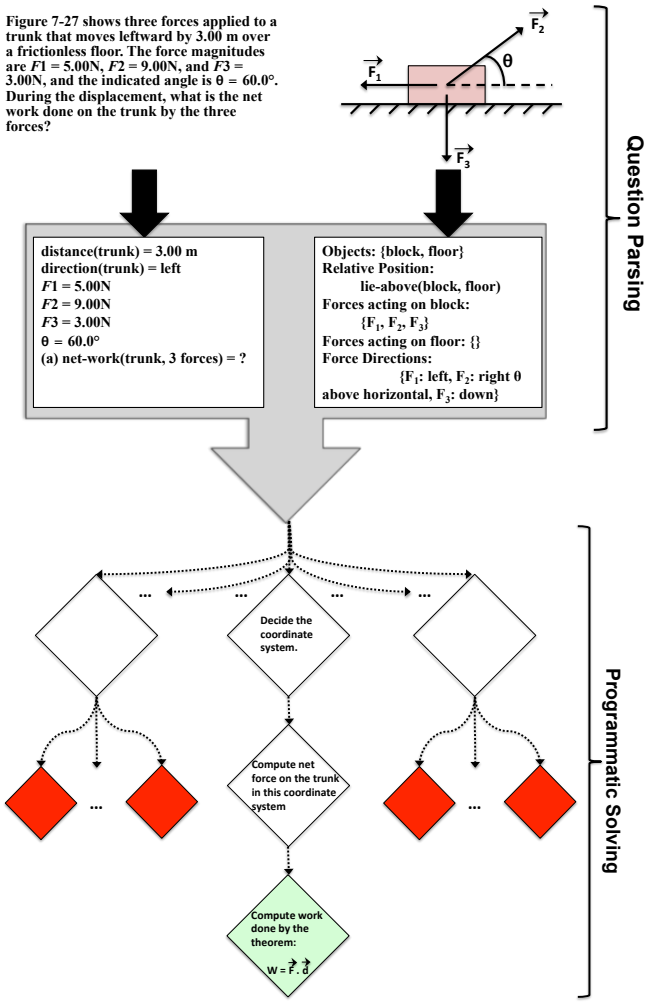


Figure 7. Our two-stage approach for solving a Newtonian Physics question. The first stage, *Question Parsing*, parses the question text and diagram into a (weighted) logical expression in a typed first-order logical language. The logical expression for this example is shown in the two rectangular white boxes. Each literal in this expression is weighted (weights not shown for simplicity). The second stage, *Programmatic Solving*, takes this formal representation and performs probabilistic reasoning using a set of pre-defined axiom programs. A sequence of program applications (corresponding to the process required to solve the question) may lead to the solution (shown in green).

cludes a textual description 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 geometry.

The second task is of solving university level Newtonian Physics questions such as the one in Figure 7. 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 be able to interpret both the question text as well as the associated 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.).

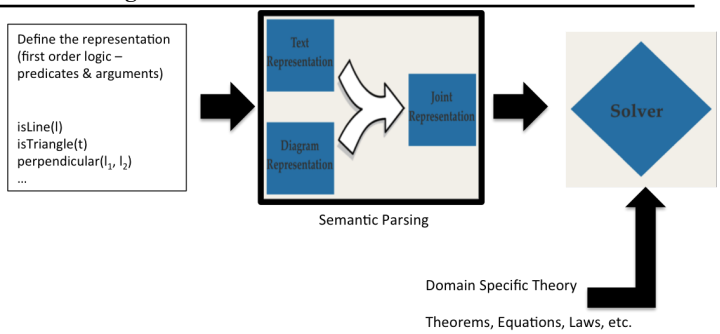


Figure 8. The framework of our Parsing to Programs approach. The approach solves the question in two stages. The first stage, *Question Parsing*, parses the question text and any associated diagram into an equivalent (weighted) logical expression in a typed first-order logic language. The second stage, *Programmatic Solving* takes this formal representation of the question and solves it using the domain specific theory provided to the system.

4.1. 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 8. This can be perceived as a natural language interface to expert systems (Hayes-Roth et al., 1983) which were popular in early stages of AI for solving these kinds of problems.

4.1.1. Question Text Parsing

For question parsing, we first pick the formal domain representation. The formal representation is usually provided by a domain expert or a programmer, but may be learned automatically in certain cases. However, we assume this to be given in our work. We choose a first-order logic representation that includes known numbers or entities as constants, unknown numbers or entities as variables, relations as predicates and properties of entities as functions. The parsers learn a set of relations that potentially correspond to the problem text (or diagram) along with confidence scores.

Question 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 may perform some *relation completion* post processing. We use a log-linear model based relation extractor to maps question text to a logical expression in the formal language. Our semantic parser uses a part-based log-linear model which combines the various steps in question parsing in a joint model. Our parser first maps words or phrases in the input text x to corresponding concepts in the formal language. Then, it identifies relations between identified concepts. Finally, it performs relation completion. We choose a log-linear model over the parses which decomposes into two parts. Let $p = \{p_1, p_2\}$ where p_1 denotes the concepts identified in p and p_2 denotes the identified relations. The log-linear model also factorizes into two components for concept and relation identification:

$$P(p|x;\boldsymbol{\theta}_p) = \frac{1}{Z(x;\boldsymbol{\theta}_p)} \exp\left(\boldsymbol{\theta}_p^T \boldsymbol{\phi}(p,x)\right)$$

$$\boldsymbol{\theta}_p^T \boldsymbol{\phi}(p,x) = \boldsymbol{\theta}_{p1}^T \boldsymbol{\phi}_1(p_1,x) + \boldsymbol{\theta}_{p2}^T \boldsymbol{\phi}_2(p_2,x)$$

$Z(x;\boldsymbol{\theta}_p)$ is the partition function of the log-linear model and $\boldsymbol{\phi}$ is the concatenation $[\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2]$. The complexity of searching for the highest scoring latent parse is exponential. Hence, we use beam search with a fixed beam size for inference. That is, in each step, we only expand few most promising candidates so far given by the current score. We first infer p_1 to identify a beam of concepts. Then, we infer p_2 to identify relations among candidate concepts. We find the optimal parameters $\boldsymbol{\theta}_p$ using maximum-likelihood estimation with L2 regularization:

$$\boldsymbol{\theta}_p^* = \arg\max_{\boldsymbol{\theta}_p} \sum_{(x,p)\in Train} \log P(p|x;\boldsymbol{\theta}_p) - \lambda ||\boldsymbol{\theta}_p||_2^2$$

We use L-BFGS to optimize the objective.

4.1.2. Diagram Parsing

The diagram parser is usually more domain specific and uses a combination of existing computer vision machinery for detecting 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. This is followed by some rule-based mapping of the detected diagram parse to the formal language of the domain. We will describe application specific implementations of our diagram parsers later.

4.1.3. Programmatic Solving

We assume the domain theory in the form of structured programs. The domain theory can again be provided by a domain expert or a programmer, or as we shown later in section 4.2.1, it could be harvested automatically from textbooks. Given the domain theory, we perform classical logical inference with the domain knowledge and the formal problem description obtained from our parser output. At a high level, the programmatic solver could be any expert system for that domain which can take the formal problem description as input.

4.2. Applications

4.2.1. Answering Geometry Questions

We first apply the *Parsing to Programs* approach described above to solve SAT style geometry problems. Our approach on diagram and question text parsing which is inspired from previous work in this area (Seo et al., 2014; 2015). We write axioms of geometry in the form of horn-clause rules and show that we can use probabilistic logic to solve these problems. 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 4.2.1).

Question Parsing: Seo et al. (2015) proposed *GEOS*, the first automated approach for solving geometry problems. We extend the approach of *GEOS* to parse geometry problems. *GEOS* chose the formal problem description as 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. We use the same formal representation in our work.

Our parser learns a set of relations that potentially correspond to the problem text (or diagram) along with confidence scores using the question text parser described before. Then, a subset of relations that maximize the joint text and diagram score are picked as the problem description. For diagram parsing, *GEOS* uses a publicly available diagram parser for geometry problems (Seo et al., 2014) that provides confidence scores for each literal to be true in the diagram. We use the diagram parser from *GEOS* to handle diagrams in our work. Relation completion is performed using a deterministic rule-based approach as in *GEOS* which handles *implicit concepts* like the "Equals" relation in the sentence "Circle O has a radius of 5" and *coordinating conjunctions* like "bisect" between the two lines and two angles in "AM and CM bisect BAC and BCA". We refer the interested reader to section 4.3 in (Seo et al., 2015) for details.

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 3 gives some examples of geometry theorems written as horn clause rules. The free vari-

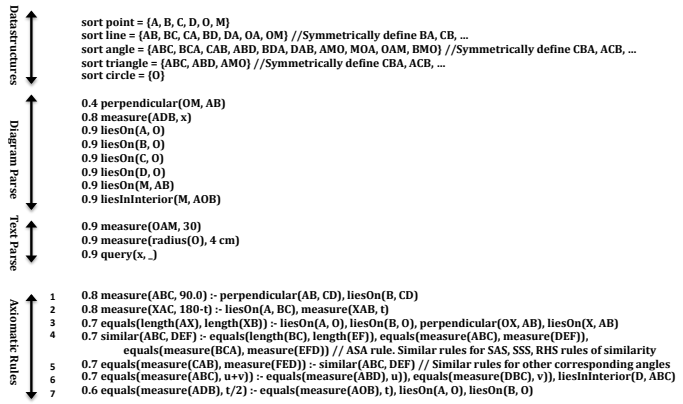


Figure 9. A sample logical program (in prolog style) that solves the problem in Figure 6. The program consists of a set of data structure declarations that correspond to types in the prolog program, a set of declarations from the diagram and text parse and a subset of the geometry axioms written as horn 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.

ables in the theorems are universally quantified. The variables are also typed. For example, *ABC* can be of type *triangle* or *angle* but not *line*.

A sample logical program (in prolog notation) that solves the problem in Figure 6 is given in Figure 9. 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.

Harvesting Axiomatic Knowledge from Textbooks: We proposed 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 (Chang et al., 2003; Etzioni et al., 2004), 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 across multiple textbooks[6], and (b)

[6]The 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, Pythagoras theorem would typically be introduced after introducing the notion of a right angled triangle.

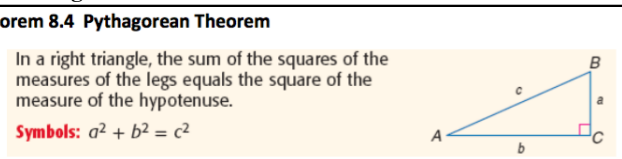


Figure 10. An excerpt of a textbook from our dataset that introduces the Pythagoras theorem. The textbook has a lot of typographical features that can be used to harvest this theorem: The textbook explicitly labels it as a "theorem"; there is a colored bounding box around it; an equation writes down 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: $isTriangle(ABC) \wedge perpendicular(AC, BC) \implies BC^2 + AC^2 = AB^2$.

utilizing rich contextual and typographical features[7] 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.

We refer the reader to Sachan et al. (2017) for details about our work. We used our model to extract and parse axiomatic knowledge from a novel dataset 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 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.

We used 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 9. Table 4 shows the results for our best end-to-end system and compares it to *GEOS* on the practice and official SAT dataset from (Seo et al., 2015) 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

[7]Textbooks contain rich context and typographical information (see Figure 10 for an illustrative example). We use this rich information as features in our model.

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) ∧ liesOn(C,AB)	angle(ACD) + angle(DCB) = 180°
Vertically Opp. Angles	intersectAt(AB, CD, M)	angle(AMC) = angle(BMD)

Table 3. Examples of geometry theorems as horn clause rules.

	Practice	Official	Textbook
GEOS	61	49	32
Our System	**64**	**55**	**51**
Oracle	80	78	72

Table 4. Scores for solving geometry questions on the SAT practice and official datasets and a dataset of questions from the 20 textbooks. We use SATâĂŹs 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).

	Interpretability		Usefulness	
	GEOS	***O.S.***	***GEOS***	***O.S.***
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 5. 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.

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 'interpretable' and 'useful' the two systems were on a scale of 1-5. Table 5 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.

4.2.2. Answering Newtonian Physics Questions

We also applied the *Parsing to Programs* approach to the task of answering Newtonian Physics problems. 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.

Question Parsing: We again chose our logical language as a subset of typed first-order logic comprising of *constants* (3.00 m, 5.00 N, 60°, etc.), *variables* (F_1, θ, etc.), and a hand-picked set of 138 predicates (*equals*, *mass*, *distance*, *force*, *speed*, *velocity*, *work*, etc.). Each element in the language is strongly typed. For example, the predicate *mass* takes the first argument of the type "object" and the second argument of the type "mass-quantity" such as *2kg*, *3g*, etc. The question text parse is represented as a logical formula over constants, variables, existential quantifiers and conjunctions over applications of predicates to a set of arguments. We used our two-stage statistical model described before for question text parsing.

Physics 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 were recognized and organized in context to achieve a complete logical representation of the diagram. We proposed a pipeline approach for recognizing various diagram elements in diagrams and mapping them to our formal language. We refer the reader to the detailed proposal document for details.

Programmatic Solving: We present the domain knowledge to the system in the form of structured programs. Some example programs are shown in Figure 11. 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 $\mathscr{P}$ represent this set of programs. For

```
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((âĂIJ-mg i + 0 jâĂİ)):
        forces.append((âĂIJmg i + 0 jâĂİ))

def Newton_II_law(Mass m, Forces forces, Accelerations accs):
    net_force = vector_additon(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 11. Example programs used by our approach.

any program $p \in \mathscr{P}$, 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.

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 $\mathscr{P}$.

Do

1. **Match Programs:** Match the pre-conditions of the programs against the set of literals i.e. find all programs $p \in \mathscr{P}$ s.t. the precondition p^{pr} can be unified with some set of literals L.
2. **Select Program:** Sample a program (randomly uniformly) among the matching programs. Stop if no program can be applied.
3. **Apply Program:** Apply the chosen program by adding the result to the set of literals/constraint set.

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.

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

Table 6. 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.

Finally, we used an off-the shelf library[8] 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 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.

Results: We performed a user study with 10 students[9] 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 in Table 6. 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.

5. Ongoing Work: Generating Questions from Instructional Material

So far, we have described several approaches for answering questions about instructional material. These included automated ways to answer reading comprehension questions and questions about science and math curriculum of the students. Now, we invert the problem and aim to create natural questions from the

[8] http://docs.sympy.org/dev/modules/solvers/solvers.html#sympy.solvers.solvers.nsolve

[9] All the students selected for the user study scored atleast 4 ("well qualified to receive college credit") or 5 ("extremely well qualified to earn college credit") on the 2016 AP Physics C exam.

instructional material. One direct application of question generation in this context is in the area of education. We can use the generated questions for automated testing and resource creation.

Similar to question answering, we bifurcate the task of question asking from instructional material into two categories: (a) generating open domain questions based on natural language text e.g. generating reading comprehension questions and (b) generating questions in closed domains such as math and science material. In the latter, we will primarily focus on generating geometry questions.

5.1. Ongoing Work: Generating Open Domain Reading Comprehension Questions

Generating open domain questions based on natural language text is a difficult but an understudied problem. Recently, however, there has been some traction to this task due to the resurgence of deep learning (Du et al., 2017; Tang et al., 2017).

Similar to these works, we build on a sequence to sequence model with soft attention (Bahdanau et al., 2014; Sutskever et al., 2014) as our base question asking model for generating reading comprehension questions. We evaluate our approach on three datasets comprising of question answer pairs: *SQUAD*, *MS MARCO* and *WikiQA* datasets. Previous literature has often treated the AQ task as one of transforming text sentences into questions (e.g. see (Heilman and Smith, 2009)) via some set of manually engineered rules, etc. These rule-based approaches are not scalable as the transformation rules need to be hand engineered. We will show the results of a simple seq2seq AQ approach later.

5.1.1. Ongoing Work: Learning what to ask

Another important aspect of question asking is knowing what is a good question to ask. Not all questions are equally good. The answer of this could vary widely based on the application at hand.

We propose to use an over-generate and re-rank strategy to finally choose what questions to ask based on measures of fluency and correctness. The approach would re-rank the beam of possible questions generated by our previous model. We assume some supervision in the form of preference (possibly provided by turkers) for questions for fluency and correctness. We will follow evaluation setups similar to question asking (Heilman, 2011) and paraphrase generation (Callison-Burch, 2008). Following them, in the instructions given to raters, a preferred output question will be defined as a question where at least part of the meaning of the input sentence is preserved. Raters will be shown output questions alongside its corresponding input sentence and will be asked which one is preferred most in terms of fluency and correctness.

Given these preferences, we plan to use our rank SVM formulation to re-rank the top-K outputs generated by our various models when performing beam search. We hope to show that the re-ranking approach can lead to improvements in question asking in terms of automatic evaluation metrics as well as in human evaluations.

5.2. Ongoing Work: Generating Geometry Questions

For generating questions based on math and science material, we plan to generate controlled natural questions based on the set of rules and axioms in the specific domain. For example, we propose to generate geometry questions with desired solution characteristics (such as difficulty level, use of a certain set of concepts, axioms, etc.) such that it can be used as instructional content. This includes generation of the question text as well as associated diagrams.

As a case study, we work on generating SAT style geometry questions. We assume access to a set of axioms of geometry. This can be manually curated or harvested from textbooks as described earlier. To generate a new geometry problem, we can assume that we are provided with a geometry figure. These may be obtained from an existing smaller dataset or may be queried from the web. We leave figure generation as future work. Given a geometry diagram, we use the diagram parser (Seo et al., 2014) to obtain the logical formula parse for the diagram.

Given the geometry language $\mathscr{L}$ (described previously) and the set of axioms $\mathscr{T}$, we first generate geometry problems over that figure in the form of pairs of assumptions and goals in the geometry language. Then, given the problems in the formal language, we would generate natural language text to generate natural questions. This model can be trained on an existing corpus of geometry questions in order to generate more similar questions. We refer the reader to the detailed proposal document for details.

6. Ongoing Work: Self Training for Joint Question Answering and Question Generation

In this section, we argue that QA and AQ are closely related[10] tasks. Yet, the literature on QA and AQ views the two as entirely separate tasks. Hence, we explore this relationship among the two tasks by jointly learning to answer as well as ask questions. An improved ability to answer as well as ask questions will help us build better *curious* machines that can interact with humans by asking as well as answering 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 unlabeled free text and then answers for the generated questions. We 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. This additional data is then used to retrain our QA and AQ models and the procedure is repeated. For more details, we refer the reader to the detailed proposal.

[10] We can think of QA and AQ as inverse of each other.

	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
(Tang et al., 2017)	0.4836	0.4911	0.2751	0.8643	0.8716	0.7814
SelfTrain(0)	0.4712	0.4783	0.2628	0.8580	0.8647	0.7740
SelfTrain(100)	0.5236	0.4934	0.2734	0.8809	0.8904	0.7987
SelfTrain(1000)	0.5372	0.5022	0.2842	0.8848	0.8946	**0.8012**
SelfTrain(10000)	**0.5393**	**0.5067**	**0.2887**	**0.8887**	**0.8961**	0.8007

Table 7. Performance of our models and QA baselines on *SQUAD* and *MS MARCO*.

	MAP	MRR
CNN (Yang et al., 2015)	0.665	0.652
APCNN (Santos et al., 2016)	0.696	0.689
NASM (Miao et al., 2016)	0.707	0.689
ABCNN (Yin et al., 2015)	0.702	0.692
KVMN (Miller et al., 2016)	0.707	0.727
(Wang et al., 2016b)	0.706	0.723
(Wang et al., 2016a)	0.734	0.742
(Wang and Jiang, 2016)	0.743	0.755
(Tang et al., 2017)	0.700	0.684
SelfTrain(0)	0.691	0.675
SelfTrain(100)	0.718	0.719
SelfTrain(1000)	0.734	0.733
SelfTrain(10000)	**0.754**	**0.753**

Table 8. Performance of our models and the QA baselines on the *WikiQA* dataset. Our model achieves the new state-of-the-art on this dataset.

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 (Zhu, 2005). The performance of our QA and AQ models 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 (Bengio et al., 2009) 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. Further details of these heuristics is provided in the extended proposal document.

A number of cognitive scientists also 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 we can achieve further improvements by coupling the curriculum learning heuristics with a measure for diversity. Further details are available in the extended proposal document.

In theory, we can jointly perform question answering and question asking on both open domain question answering (specifically reading comprehension tasks) and closed domain question answering (specifically geometry question answering). Here, we report some preliminary results on joint QA and AQ on reading comprehension tasks and leave joint QA and AQ for closed domain tasks as future work.

Tables 7, 8, 9 and 10 show the results of the QA and AQ evaluations on four datasets: *SQUAD* (Rajpurkar et al., 2016b), *MS MARCO* (Nguyen et al., 2016b), *WikiQA* (Yang et al., 2015)

	MAP	MRR
(He and Lin, 2016)	0.758	0.822
(He et al., 2015)	0.762	0.830
(Tay et al., 2017)	0.770	0.825
(Rao et al., 2016)	0.780	0.834
SelfTrain(0)	0.742	0.813
SelfTrain(100)	0.776	0.831
SelfTrain(1000)	0.783	0.836
SelfTrain(10000)	**0.798**	**0.854**

Table 9. Performance of our models and the QA baselines on the *TrecQA* dataset. Our model achieves the state-of-the-art on this dataset.

and *TrecQA* (Wang et al., 2007). We observe that self-training while jointly training the QA and AQ models leads to better performance than QA or AQ alone. 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. We also obtained new state-of-the-art results on *WikiQA* and *TrecQA* datasets for the QA task.

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 7, 8, 9 and 10, 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 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.

7. Conclusion and Future Work

In this thesis, we propose the task of teaching machines to automatically solve some standardized tests such as reading comprehensions, and various elementary school math and science tests. We proposed two broad set of techniques based on latent answer-entailing structures when solving open-domain question answering tasks 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 in both these settings 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 low data problem. Finally, we hope to show that these approaches can potentially be useful for students.

In the future, we would like to generalize the two frameworks and release easy-to-use tool kits for practitioners to build their own question answering systems in their domains of interest. We would like to build more systems for examinations in more advanced areas such as medical diagnostics, accountancy, finance, etc. Finally, 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 help students.

	SQUAD			MS MARCO			WikiQA			TrecQA		
	BLEU4	METEOR	ROUGE	BlEU4	METEOR	ROUGE	BlEU4	METEOR	ROUGE	BlEU4	METEOR	ROUGE
IR	1.07	7.77	20.85	0.81	5.42	15.78	0.93	6.89	19.98	0.83	5.73	16.34
MOSES	0.31	10.49	17.88	0.27	9.74	15.82	0.32	10.26	17.27	0.29	9.86	17.02
DirectIn	11.25	14.91	22.51	10.82	13.35	20.38	10.94	14.18	22.01	9.59	12.21	19.76
H&S	11.23	16.00	31.03	10.16	15.07	30.00	10.35	15.30	30.72	9.19	12.72	23.38
(Tang et al., 2017)	5.03	-	-	9.31	-	-	3.15	-	-	-	-	-
(Du et al., 2017)	12.28	16.62	39.75	-	-	-	-	-	-	-	-	-
SelfTrain(0)	12.31	16.67	39.78	11.14	15.60	37.26	11.38	16.08	38.42	10.96	14.25	27.27
SelfTrain(100)	14.14	18.70	42.46	13.25	17.10	40.28	13.10	17.00	40.93	11.63	15.05	29.07
SelfTrain(1000)	14.27	18.78	42.93	13.61	17.87	**41.23**	13.22	18.34	42.72	12.24	15.93	30.26
SelfTrain(10000)	14.28	**18.79**	**42.97**	**13.74**	**17.89**	41.07	**15.26**	**19.45**	**44.77**	**14.87**	**16.88**	**31.91**

Table 10. Performance of our model variants and the various QG baselines on the *SQUAD*, *MS MARCO*, *WikiQA* and *TrecQA* datasets.

8. Time Line

- December 2017: Submission of completed work on "Joint Question Answering and Question Asking for Reading Comprehensions" to NAACL 2018. (Completed)
- January 2018: Thesis proposal.
- February 2018: Submission of mostly completed work on "Physics QA" to KDD 2018.
- February 2018: Submission of partially completed work on "Question Asking for Geometry" to ACL 2018.
- June 2018: Submission of planned case studies of usefulness of our Work for Educational applications to arXiv and later to CHI 2019 in July.
- July-August 2018: Thesis Writing and Defence.

References

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.

Kirsti M Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102, 2005.

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.

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.

Yigal Attali and Jill Burstein. Automated essay scoring with e-rater® v. 2.0. *ETS Research Report Series*, 2004(2), 2004.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

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.

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.

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.

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.

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.

Peter Brusilovsky and Christoph Peylo. Adaptive and intelligent web-based educational systems. *International Journal of Artificial Intelligence in Education (IJAIED)*, 13:159–172, 2003.

Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, and Martin Chodorow. Enriching automated essay scoring using discourse marking. *ERIC*, 2001.

Christopher Callison-Burch. *Paraphrasing and translation*. PhD thesis, John Hopkins University, 2008.

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.

Eugene Charniak. *Toward a model of children's story comprehension.* PhD thesis, Massachusetts Institute of Technology, 1972.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051, 2017.

Peter Clark. Elementary School Science and Math Tests as a Driver for AI:Take the Aristo Challenge! In *Proceedings of IAAI*, 2015.

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.

Kenneth Mark Colby. *Artificial Paranoia: A Computer Simulation of Paranoid Processes*. Elsevier Science Inc., New York, NY, USA, 1975. ISBN 0080181627.

Donald Davidson. *The individuation of events*. Springer, 1969.

Hendrik Drachsler, Katrien Verbert, Olga C Santos, and Nikos Manouselis. Panorama of recommender systems to support learning. In *Recommender systems handbook*, pages 421–451. Springer, 2015.

Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*, 2017.

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.

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.

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.

David A Ferrucci. Introduction to ÃćÂĂÂIJthis is watsonÃćÂĂÂİ. *IBM Journal of Research and Development*, 56(3.4):1–1, 2012.

Robert M French. Subcognition and the limits of the turing test. *Mind*, 99:53–65, 1996.

Noah D Goodman and Andreas Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org, 2014. Accessed: 2017-3-14.

John L Gordon. Creating knowledge maps by exploiting dependent relationships. *Knowledge-based systems*, 13(2):71–79, 2000.

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.

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.

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.

Hua He and Jimmy J. Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 937–948, 2016.

Hua He, Kevin Gimpel, and Jimmy J. Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1576–1586, 2015.

Michael Heilman. *Automatic factual question generation from text*. PhD thesis, Carnegie Mellon University, 2011.

Michael Heilman and Noah A Smith. Question generation via overgenerating transformations and ranking. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA LANGUAGE TECHNOLOGIES INST, 2009.

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.

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.

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.

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.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of EMNLP*, 2014.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. Mnemonic reader for machine comprehension. *CoRR*, abs/1705.02798, 2017.

Gwo-Jen Hwang. A conceptual map model for developing intelligent tutoring systems. *Computers & Education*, 40(3):217–235, 2003.

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.

Peter Jackson. *Introduction to expert systems*. Addison-Wesley Pub. Co., Reading, MA, 1986.

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.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

Rohit J Kate, Yuk Wah, Wong Raymond, and J Mooney. Learning to transform natural to formal languages. In *Proceedings of AAAI-05*. Citeseer, 2005.

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.

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.

Tushar Khot, Ashish Sabharwal, and Peter Clark. Answering complex questions using open information extraction. *arXiv preprint arXiv:1704.05572*, 2017.

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.

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.

Hector J Levesque. The winograd schema challenge. In *Proceeding of KR*, 2010.

Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7, 2002.

Shieu-Hong Lin. Data mining for student retention management. *Journal of Computing Sciences in Colleges*, 27(4):92–99, 2012.

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.

Rui Liu, Junjie Hu, Wei Wei, Zi Yang, and Eric Nyberg. Structural embedding of syntactic trees for machine comprehension. *CoRR*, abs/1703.00572, 2017.

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.

William C Mann and Sandra A Thompson. {Rhetorical Structure Theory: Toward a functional theory of text organisation}. *Text*, 3(8): 234–281, 1988.

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.

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.

Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736, 2016.

P Hartley Millar. On the point of the imitation game. *Mind*, 82(328): 595–597, 1973.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *CoRR*, abs/1606.03126, 2016.

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.

James H Moor. An analysis of the turing test. *Philosophical Studies*, 30 (4):249–257, 1976.

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.

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, 2016a. URL http://arxiv.org/abs/1611.09268.

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*, 2016b.

Joseph D Novak. *Learning, creating, and using knowledge: Concept maps as facilitative tools in schools and corporations*. Routledge, 2010.

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.

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.

Terence Parsons. *Events in the Semantics of English*, volume 5. In MIT Press, 1990.

Marc Prensky. Digital game-based learning. *Computers in Entertainment (CIE)*, 1(1):21–21, 2003.

Richard L Purtill. Beating the imitation game. *Mind*, 80(318):290–294, 1971.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016a. URL http://arxiv.org/abs/1606.05250.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016b. URL http://arxiv.org/abs/1606.05250.

Jinfeng Rao, Hua He, and Jimmy Lin. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1913–1916, 2016. ISBN 978-1-4503-4073-1.

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.

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 sytems-Volume 6*, pages 13–19. Association for Computational Linguistics, 2000.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of EMNLP*, 2015.

Subhro Roy, Tim Vieira, and Dan Roth. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13, 2015.

Mrinmaya Sachan and Eric P. Xing. Machine comprehension using rich semantic representations. In *Proceedings of ACL*, 2016.

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.

Mrinmaya Sachan, Kumar Dubey, and Eric P. Xing. Science question answering using instructional materials. In *Proceedings of ACL*, 2016.

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. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 784–795, 2017.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*, 2016.

Ayse Pinar Saygin, Ilyas Cicekli, and Varol Akman. Turing test: 50 years later. *Minds and machines*, 10(4):463–518, 2000.

John R Searle. Minds, brains, and programs. *THE BEHAVIORAL AND BRAIN SCIENCES*, 3:417–457, 1980.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. Diagram understanding in geometry questions. In *Proceedings of AAAI*, 2014.

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.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

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.

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.

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.

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. Association for Computational Linguistics, 2015.

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.

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.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*, 2017.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Enabling efficient question answer retrieval via hyperbolic neural networks. *CoRR*, abs/1707.07847, 2017. URL http://arxiv.org/abs/1707.07847.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830, 2016.

Alan M Turing. Computing machinery and intelligence. *Mind*, pages 433–460, 1950.

Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016a.

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.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*, volume 7, pages 22–32, 2007.

Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*, 2016.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Sentence similarity learning by lexical decomposition and composition. *CoRR*, abs/1602.07019, 2016b.

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.

Joseph Weizenbaum. *Computer power and human reason: From judgment to calculation.* WH Freeman & Co, 1976.

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.

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.

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.

Yi Yang, Scott Wen-tau Yih, and Chris Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, September 2015.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Semi-markov phrase-based monolingual alignment. In *Proceedings of EMNLP*, 2013.

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.

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.

A. L. Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Comput.*, 2003.

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.

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.

Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.