Learning to Solve Geometry Problems from Natural Language Demonstrations in Textbooks

Mrinmaya Sachan **Eric P. Xing**
School of Computer Science
Carnegie Mellon University
{mrinmays, epxing}@cs.cmu.edu

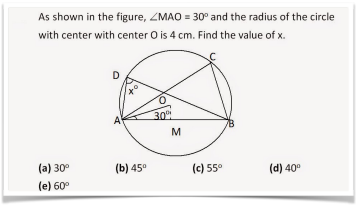
Abstract

Humans as well as animals are good at *imitation*. Inspired by this, the *learning by demonstration* view of machine learning learns to perform a task from detailed example demonstrations. In this paper, we introduce the task of *question answering using natural language demonstrations* where the question answering system is provided with detailed demonstrative solutions to questions in natural language. As a case study, we explore the task of learning to solve geometry problems using demonstrative solutions available in textbooks. We collect a new dataset of demonstrative geometry solutions from textbooks and explore approaches that learn to interpret these demonstrations as well as to use these interpretations to solve geometry problems. Our approaches show improvements over the best previously published system for solving geometry problems.

1 Introduction

Cognitive science emphasizes the importance of *imitation* or *learning by example* (Meltzoff and Moore, 1977; Meltzoff, 1995) in human learning. When a teacher signals a pedagogical intention, children tend to imitate the teacher's actions (Buchsbaum et al., 2011; Butler and Markman, 2014). Inspired by this phenomenon, the *learning by demonstration* view of machine learning (Schaal, 1997; Argall et al., 2009; Goldwasser and Roth, 2014) assumes training data in the form of example demonstrations. A task is demonstrated by a teacher and the learner generalizes from these demonstrations in order to execute the task.

In this paper, we introduce the novel task of *question answering using natural language demonstrations*. Research in question answering has traditionally focused on learning from question-answer pairs (Burger et al., 2001). However, it is well-established in the educational psychology literature (Allington and Cunningham, 2010; Felder et al., 2000) that children tend to learn better and faster from concrete illustrations and demonstrations. In this paper, we raise the question – "Can we leverage demonstrative solutions for questions as provided by a teacher to improve our question answering systems?"



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 1: Above: An example SAT style geometry problem with the text description, corresponding diagram and (optionally) answer candidates. Below: A logical expression that represents the meaning of the text description and the diagram in the problem. *GEOS* derives a weighted logical expression where each predicates also carry a weighted score but we do not show them here for clarity.

As a case study, we propose the task of learning to solve SAT geometry problems (such as the one in Figure 1) using demonstrative solutions to these problems (such as the one in Figure 2). Such demonstrations are common in textbooks as they help students learn how to solve geometry problems effectively. We build a new dataset of demonstrative solutions of geometry problems and show that it can be used to improve *GEOS* (Seo et al., 2015), the state-of-the-art in solving geom-

*Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 251–261,
Vancouver, Canada, August 3-4, 2017.

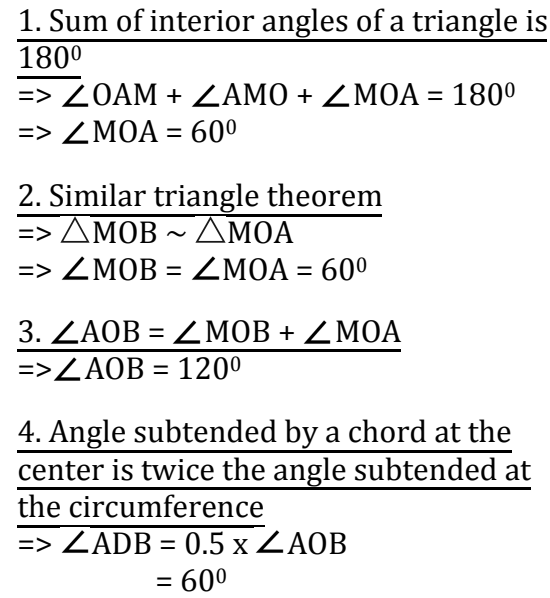


Figure 2: An example demonstration on how to solve the problem in Figure 1: (1) Use the theorem that the sum of interior angles of a triangle is 180°and additionally the fact that ∠AMO is 90°to conclude that ∠MOA is 60°. (2) Conclude that △MOA ∼ △MOB (using a similar triangle theorem) and then, conclude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

etry problems.

We also present a technique inspired from recent work in situated question answering (Krishnamurthy et al., 2016) that jointly learns how to interpret the demonstration and use this interpretation to solve geometry problems. We model the interpretation task (the task of recognizing various states in the demonstration) as a semantic parsing task. We model state transitions in the demonstration via a deduction model that treats each application of a theorem of geometry as a state transition. We describe techniques to learn the two models separately as well as jointly from various kinds of supervision: (a) when we only have a set of question-answer pairs as supervision, (b) when we have a set of questions and demonstrative solutions for them, and (c) when we have a set of question-answer pairs and a set of demonstrations.

An important benefit of our approach is 'interpretability'. While *GEOS* is uninterpretable, our approach utilizes known theorems of geometry to deductively solve geometry problems. Our approach also generates demonstrative solutions (like Figure 2) as a by-product which can be provided to students on educational platforms such as MOOCs to assist in their learning.

We present an experimental evaluation of our approach on the two datasets previously introduced in Seo et al. (2015) and a new dataset collected by us from a number of math textbooks in India. Our experiments show that our approach of leveraging demonstrations improves *GEOS*. We also performed user studies with a number of school students studying geometry, who found that our approach is more *interpretable* as well as more *useful* in comparison to *GEOS*.

2 Background: GEOS

GEOS solves geometry problems via a multi-stage approach. It first learns to parse the problem text and the diagram to a formal problem description compatible with both of them. The problem description is a first-order logic expression (see Figure 1) that includes known numbers or geometrical entities (e.g. 4 cm) as constants, unknown numbers or geometrical entities (e.g. O) as variables, geometric or arithmetic relations (e.g. *isLine*, *isTriangle*) as predicates and properties of geometrical entities (e.g. *measure*, *liesOn*) as functions. The parser first learns a set of relations that potentially correspond to the problem text (or diagram) along with confidence scores. Then, a subset of relations that maximize the joint text and diagram score are picked as the problem description.

For diagram parsing, *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 in our work too.

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

Finally, *GEOS* uses a numerical approach to check the satisfiablity of literals, and to answer the multiple-choice question. While this solver is grounded in coordinate geometry and indeed works well, it has some issues: *GEOS* requires an explicit mapping of each predicate to a set of constraints over point coordinates. For example, the predicate *isPerpendicular*(AB, CD) is mapped to the constraint $\frac{y_B - y_A}{x_B - x_A} \times \frac{y_D - y_C}{x_D - x_C} = -1$. These con-

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

Table 1: Examples of geometry theorems as horn clause rules.

straints can be non-trivial to write and often require manual engineering. As a result, *GEOS*'s constraint set is incomplete and it cannot solve a number of SAT style geometry problems. Furthermore, this solver is not interpretable. As our user studies show, it is not natural for a student to understand the solution of these geometry problems in terms of satisfiability of constraints over coordinates. A more natural way for students to understand and reason about these problems is through deductive reasoning using well-known axioms and theorems of geometry. This kind of deductive reasoning is used in explanations in textbooks. In contrast to *GEOS* which uses supervised learning, our approach learns to solve geometry problems by interpreting natural language demonstrations of the solution. These demonstrations illustrate the process of solving the geometry problem via stepwise application of geometry theorems.

3 Theorems as Horn Clause Rules

We represent theorems as horn clause rules that map a premise in the logical language to a conclusion in the same language. Table 1 gives some examples of geometry theorems written as horn clause rules. The free variables in the theorems are universally quantified. The variables are also typed. For example, *ABC* can be of type *triangle* or *angle* but not *line*. Let $\mathscr{T}$ be the set of theorems. Formally, each theorem $t \in \mathscr{T}$ maps a logical formula $l_t^{(pr)}$ corresponding to the premise to a logical formula $l_t^{(co)}$ corresponding to the conclusion. The demonstration can be seen as a program – a sequence of horn clause rule applications that lead to the solution of the geometry problem. Given a current state, theorem t can be applied to the state if there exists an assignment to free variables in $l_t^{(pr)}$ that is true in the state. Each theorem application also has a probability associated with it; in our case, these probabilities are learned by a trained model. The state diagram for the demonstration in Figure 2 is shown in Figure 3. Now, we describe the various components of our *learning from demonstrations* approach: a *semantic parser* to interpret the demonstration and a *deductive solver* that learns to chain theorems.

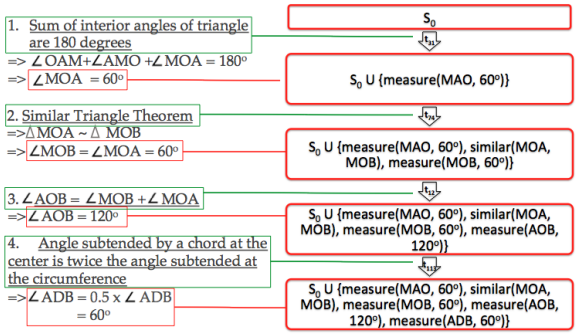


Figure 3: State sequence corresponding to the demonstration in Figure 2. Theorems applied are marked in green and the state information is marked in red. Here S_0 corresponds to the state derived from question interpretation and each theorem application subsequently adds new predicates to the logical formula corresponding to S_0. The final state contains the answer: measure(ADB, 60°). This annotation of states and theorem applications is provided only for illustrative purposes. It is not required by our model.

4 Approach

4.1 Interpretation via Semantic Parsing

We first describe a semantic parser that maps a piece of text (in the geometry question or a demonstration) to a logical expression such as the one shown in Figure 1. Our semantic parser uses a part-based log-linear model inspired from the multi-step approach taken in *GEOS*, which, in-turn is closely related to prior work in relation extraction and semantic role labeling. However, unlike *GEOS*, our parser combines the various steps in a joint model. Our parser first maps words or phrases in the input text x to corresponding concepts in the geometry language. Then, it identifies relations between identified concepts. Finally, it performs relation completion to handle implications and coordinating conjunctions. 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 relation completion is performed by using a similar rule-based approach as in GEOS. 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 (100) for inference. That is, in each step, we only expand the ten 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. Finally, 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.

This semantic parser is used to identify program states in demonstrations as well as to map geometry questions to logical expressions.

4.1.1 State and Axiom Identification

Given a demonstrative solution of a geometry problem in natural language such as the one shown in Figure 2, we identify theorem applications by two simple heuristics. Often, theorem mentions in demonstrations collected from textbooks are labeled as references to theorems previously introduced in the textbook (for example, "Theorem 3.1"). In this case, we simply label the theorem application as the referenced theorem. Sometimes, the theorems are mentioned verbosely in the demonstration. To identify these mentions, we collect a set of theorem mentions from textbooks. Each theorem is also represented as a set of theorem mentions. Then, we use an off-the-shelf semantic text similarity system (Šarić et al., 2012) and check if a contiguous sequence of sentences in the demonstration is a paraphrase of any of the gold theorem mentions. If the degree of similarity of a contiguous sequence of sentences in the demonstration with any of the gold theorem mentions is above a threshold, our system labels the sequence of sentences as the theorem. The text similarity system is tuned on the training dataset and the threshold is tuned on the development set. This heuristic works well and has a small error ($< 10\%$) on our development set.

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

4.2 Deductive Solver

Our deductive solver, inspired from Krishnamurthy et al. (2016), uses the parsed state and axiom information (when provided) and learns to score the sequence of axiom applications which can lead to the solution of the problem. Our solver uses a log-linear model over the space of possible axiom applications. Given a set of theorems $\mathscr{T}$ and optionally demonstration d, we assume $\mathbf{T} = [t_1, t_2, \ldots t_k]$ to be a sequence of theorem applications. Each theorem application leads to a change in state. Let s_0 be the initial state determined by the logical formula derived from the question text and the diagram. Let $\mathbf{s} = [s_1, s_2, \ldots s_k]$ be the sequence of program states after corresponding theorem applications. The final state s_k contains the answer to the question. We define the model score of the deduction as:

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

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

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

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

4.3 Joint Semantic Parsing and Deduction

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

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

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

Here, $\boldsymbol{\theta} = \{\boldsymbol{\theta}_p, \boldsymbol{\theta}_{ex}\}$. f_p and f_s represent the factors for semantic parsing and deduction. $f_p(p|\{q,a,d\};\boldsymbol{\theta}_p) \propto \exp\left(\boldsymbol{\theta}_p^T \boldsymbol{\phi}(p,\{q,a,d\})\right)$ and $f_s(\mathbf{s}|\mathbf{T},d,;\boldsymbol{\theta}_s) \propto \prod_{i=1}^{k} \exp\left(\boldsymbol{\theta}_{ex}^T \boldsymbol{\psi}(s_{i-1},s_i,t_i,d)\right)$ as defined in Sections 4.1 and 4.2. Next, we describe approaches to learn the joint model with various kinds of supervision.

4.4 Learning from Types of Supervision

Our joint model for parsing and deduction can be learned using various kinds of supervision. We provide a learning algorithm when (a) we only have geometry question-answer pairs as supervision, (b) when we have geometry questions and demonstrations for solving them, and (c) mixed supervision: when we have a set of geometry question-answer pairs in addition to some geometry questions and demonstrations. To do this, we implement two supervision schemes (Krishnamurthy et al., 2016). The first supervision scheme only verifies the answer and treats other states in the supervision as latent. The second scheme verifies every state in the program. We combine both kinds of supervision when provided. Given supervision $\{q_i,a_i\}_{i=1}^n$ and $\{q_i,a_i.d_i\}_{i=1}^m$, we define the following L2 regularized objective:

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

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

4.5 Features

Next, we define our feature set: $\boldsymbol{\phi}_1$, $\boldsymbol{\phi}_2$ for learning the semantic parser and $\boldsymbol{\psi}$ for learning the deduction model. Semantic parser features $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ are inspired from *GEOS*. The deduction model features $\boldsymbol{\psi}$ score consecutive states in the deduction s_{i-1}, s_i and the theorem t_i which when applied to s_{i-1} leads to s_i. $\boldsymbol{\psi}$ comprises of features that score if theorem t_i is applicable on state s_{i-1} and if the application of t_i on state s_{i-1} leads to state s_i. Table 2 lists the feature set.

5 Demonstrations Dataset

We collect a new dataset of demonstrations for solving geometry problems from a set of grade 6-10 Indian high school math textbooks by four publishers/authors – *NCERT*[1], *R S Aggarwal*[2], *R D Sharma*[3] and *M L Aggarwal*[4] – a total of $5 \times 4 = 20$ textbooks as well as a set of online geometry problems and solutions from three popular educational portals: *Tiwari Academy*[5], *School Lamp*[6] and *Oswaal Books*[7] for grade 6-10 students in India. Millions of students in India study geometry from these books and portals every year and these materials are available online. We manually

[1] http://epathshala.nic.in/e-pathshala-4/flipbook/
[2] http://www.amazon.in/Books-R-S-Aggarwal/
[3] http://www.amazon.in/Books-R-Sharma/
[4] http://www.amazon.in/Books-Aggarwal-M-L/
[5] http://www.tiwariacademy.com/
[6] http://www.schoollamp.com
[7] http://www.oswaalbooks.com

$\boldsymbol{\phi}_1$	Lexicon Map	Indicator that the word or phrase maps to a predicate in a lexicon created in *GEOS*. *GEOS* derives correspondences between words/phrases and geometry keywords and concepts in the geometry language using manual annotations in its training data. For instance, the lexicon contains ("square", *square*, *IsSquare*) including all possible concepts for the phrase "square".
	Regex for numbers and explicit variables	Indicator that the word or phrase satisfies a regular expression to detect numbers or explicit variables (e.g. "5", "AB", "O"). These regular expressions were built as a part of *GEOS*.
$\boldsymbol{\phi}_2$	Dependency tree distance	Shortest distance between the words of the concept nodes in the dependency tree. We use indicator features for distances of -3 to 3. Positive distance shows if the child word is at the right of the parentâĂŹs in the sentence, and negative otherwise.
	Word distance	Distance between the words of the concept nodes in the sentence.
	Dependency edge	Indicator functions for outgoing edges of the parent and child for the shortest path between them.
	Part of speech tag	Indicator functions for the POS tags of the parent and the child
	Relation type	Indicator functions for unary / binary parent and child nodes.
	Return type	Indicator functions for the return types of the parent and the child nodes. For example, return type of *Equals* is boolean, and that of *LengthOf* is numeric.
$\boldsymbol{\psi}$	State and theorem premise predicates	Treat the state s_{i-1} and theorem premise $l_{t_i}^{(pr)}$ as multi-sets of predicates. The feature is given by $div(s_{i-1}\|\|l_{t_i}^{(pr)})$, the divergence between the two multi-sets. $div(A,B)$, the divergence between multi-sets A and B is given by $\sum_k \frac{\min(A_k,B_k)}{B_k}$ which measures the degree to which the elements in A satisfy the pre-condition in B.
	State and theorem premise predicate-arguments	Now treat the state s_{i-1} and theorem premise $l_{t_i}^{(pr)}$ as two multi-sets over predicate-arguments. The feature is given by $div(s_{i-1}\|\|l_{t_i}^{(pr)})$, the divergence between the two multi-sets.
	State and theorem conclusion predicates	Now treat the state s_i and theorem conclusion $l_{t_i}^{(co)}$ as two multi-sets over predicate-arguments. The feature is given by $div(s_i\|\|l_{t_i}^{(co)})$, the divergence between the two multi-sets.
	State and theorem conclusion predicate-arguments	Now treat the state s_i and theorem conclusion $l_{t_i}^{(co)}$ as two multi-sets over predicate-arguments. The feature is given by $div(s_i\|\|l_{t_i}^{(co)})$, the divergence between the two multi-sets.
	State and theorem conclusion predicates	Treat the state s_i and theorem conclusion $l_{t_i}^{(co)}$ as two distributions over predicates. The feature is the total variation distance between the two distributions.
	State and theorem conclusion predicate-arguments	Now treat the state e_i and theorem conclusion $l_{t_i}^{(co)}$ as two distributions over predicate-arguments. The feature is the total variation distance between the two distributions.
	Product Features	We additionally use three product features: $\boldsymbol{\psi}_1\boldsymbol{\psi}_3\boldsymbol{\psi}_5$, $\boldsymbol{\psi}_2\boldsymbol{\psi}_4\boldsymbol{\psi}_6$ and $\boldsymbol{\psi}_1\boldsymbol{\psi}_2\boldsymbol{\psi}_3\boldsymbol{\psi}_4\boldsymbol{\psi}_5\boldsymbol{\psi}_6$

Table 2: The feature set for our joint semantic-parsing and deduction model. Features $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ are motivated from *GEOS*

marked chapters relevant for geometry in these books and then parsed them using Adobe Acrobat's *pdf2xml* parser. Then, we manually extracted example problems leading to a total of 2235 geometry problems with demonstrations. We also annotated 1000 demonstrations by labeling the various states and theorem applications. We manually collected a set of theorems of geometry by going through the textbooks, and wrote them as horn clause rules. A total of 293 unique theorems were collected. Then, we marked contiguous sentences in the demonstration texts as one of these 293 theorems or as states. An example annotation for the running example in Figures 1 and 2 is provided in Figure 3. Note that the annotation of states and theorem applications is not used in training our models and is only used for testing the accuracy of the programs induced by our model.

6 Experiments

We use three geometry question datasets for evaluating our system: practice and official SAT style geometry questions used in *GEOS*, and an additional dataset of geometry questions collected from the aforementioned textbooks. We selected a total of 1406 SAT style questions across grades 6-10. This dataset is approximately 7.5 times the size of the datasets used in Seo et al. (2015). We split the dataset into training (350 questions), development (150 questions) and test (906 questions) with equal proportion of grade 6-10 questions. We also annotated the training and development set questions with ground-truth logical forms. *GEOS* used 13 types of entities, 94 functions and predicates. We added some more entities, functions and predicates to cover other more complex concepts in geometry not covered in *GEOS*. Thus, we obtained a final set of 19 entity types and 115 functions and predicates. We use the training set to train our semantic parser with expanded set of entity types, functions and predicates. We used Stanford CoreNLP (Manning et al., 2014) for linguistic pre-processing. We also adapted the *GEOS* solver to the expanded set of entities, functions and predicates for comparison purposes. We call this system *GEOS++*.

6.1 Quantitative Results

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

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

Table 3: Scores of various approaches on the SAT practice (P) and official (O) datasets and a dataset of questions from the 20 textbooks (T). We use 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. O.S. represents our system trained on question-answer (QA) pairs, demonstrations, or a combination of QA pairs and demonstrations.

In the first setting, we only use question-answer pairs as supervision. We compare our semantic parsing and deduction model to *GEOS* and *GEOS++* on practice and official SAT style geometry questions from Seo et al. (2015) as well as the dataset of geometry questions collected from the 20 textbooks (see Table 3). On all the three datasets, our system outperforms *GEOS* and *GEOS++*. Especially on the dataset from the 20 textbooks (which is a harder dataset and includes more problems which require complex reasoning supported by our deduction model), *GEOS* and *GEOS++* do not perform very well whereas our system achieves a very good score.

Next, we only use demonstrations to train our joint model (see Table 3). We test this model on the aforementioned datasets and compare it to *GEOS* and *GEOS++* trained on respective datasets. Again, our system outperforms *GEOS* and *GEOS++* on all three datasets. Especially on the textbook dataset, this model trained on demonstrations has significant improvements as our semantic parsing and deduction model trains the deduction model as well and learns to reason about geometry using axiomatic knowledge.

Finally, we train our semantic parsing and deduction model on a combination of question answer-pairs and demonstrations. This model trained on question-answer pairs and demonstrations leads to further improvements over models trained only question-answer pairs or only on demonstrations. These results (shown in Table 3) hold on all the three datasets.

We tested the correctness of the parses and the

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

Table 4: Precision, Recall and F1 scores of the parses induced by *GEOS* and our models when only the parsing model or the joint model is used.

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

Table 5: Accuracy of the programs induced by various versions of our joint model trained on question-answer pairs, demonstrations or a combination of the two. We provide results when we use the deduction model or the joint model.

deductive programs induced by our models. First, we compared the parses induced by our models with gold parses on the development set. Table 4 reports the Precision, Recall and F1 scores of the parses induced by our models when only the parsing model or when the joint model is used and compares it with *GEOS*. We conclude that both our models perform better as compared to *GEOS* in parsing. Furthermore, our joint model of parsing and deduction further improves the parsing accuracy. Then, we compared the programs induced by the aforementioned models with gold program annotations on the textbook dataset. Table 5 reports the accuracy of programs induced by various versions of our models. Our models when trained on demonstrations induces more accurate programs as compared to the semantic parsing and deduction model when trained on question-answer pairs. Moreover, the semantic parsing and deduction model when trained on question-answer pairs as well as demonstrations achieves an even better accuracy. Our joint model of parsing and deduction induces more accurate programs as compared to the deduction model alone.

6.2 User Study on Interpretability

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

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

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

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

7 Related Work

Solving Geometry Problems: Standardized tests have been recently proposed as 'drivers for progress in AI' (Clark and Etzioni, 2016). These tests are easily accessible, and measurable, and hence have attracted several NLP researchers. There is a growing body of work on solving standardized tests such as reading comprehensions (Richardson et al., 2013, inter alia), science question answering (Clark, 2015; Schoenick et al., 2016, inter alia), algebra word problems (Kushman et al., 2014; Roy and Roth, 2015, inter alia), geometry problems (Seo et al., 2014, 2015) and pre-university entrance exams (Fujita et al., 2014; Arai and Matsuzaki, 2014).

While the problem of using computers to solve geometry questions is old (Feigenbaum and Feldman, 1963; Schattschneider and King, 1997; Davis, 2006), NLP and vision techniques were first used to solve geometry problems in Seo et al. (2015). While Seo et al. (2014) only aligned geometric shapes with their textual mentions, Seo

et al. (2015) also extracted geometric relations and built *GEOS*. We improve *GEOS* by building an axiomatic solver that performs deductive reasoning by learning from demonstrative problem solutions.

Learning from Demonstration: Our work follows the *learning from demonstration* view of machine learning (Schaal, 1997) which stems from the work on social learning in developmental psychology (Meltzoff and Moore, 1977; Meltzoff, 1995). *Learning from demonstration* is a popular way of learning policies from example state to action mappings in robotics applications. *Imitation learning* (Schaal, 1999; Abbeel and Ng, 2004; Ross et al., 2011) is a popular instance of *learning from demonstration* where the algorithm observes a human expert perform a series of actions to accomplish the task and learns a policy that "imitates" the expert with the purpose of generalizing to unseen data. Imitation learning is increasingly being used in NLP (Vlachos and Clark, 2014; Berant and Liang, 2015; Augenstein et al., 2015; Beck et al., 2016; Goodman et al., 2016a,b). However, all these models focus on learning respective NLP models from the final supervision e.g. semantic parses or denotations. However, we provide a technique to learn from demonstrations by learning a joint semantic parsing and deduction model. Another related line of work is Hixon et al. (2015) who acquire knowledge in the form of knowledge graphs for question answering from natural language dialogs and (Goldwasser and Roth, 2014) who propose a technique called *learning from natural instructions*. *Learning from natural instructions* allows human teachers to interact with an automated learner using *natural instructions*, allowing the teacher to communicate the domain expertise to the learner via natural language. However, this work was evaluated on a very simple Freecell game with a very small number of concepts (3). On the other hand, our model is evaluated on a real task of solving SAT style geometry problems.

Semantic Parsing: Semantic parsing is the NLP task of learning to map language to a formal meaning representation. Early semantic parsers learnt the parsing model from natural language utterances paired with logical forms (Zelle and Mooney, 1993, 1996; Kate et al., 2005, inter alia). However, recently indirect supervision, such as denotations (Liang et al., 2011; Berant et al., 2013, inter alia) and natural language directions for robot navigation (Shimizu and Haas, 2009; Matuszek et al., 2010; Chen and Mooney, 2011, inter alia) are being used to train these semantic parsers. In most of the above examples, the execution model is fairly simple (e.g. execution of a SQL query in a database, or binary feedback for interaction of the robot with the environment). However, our work uses demonstrations such as those given in textbooks for learning a semantic parser. Furthermore, our work learns the semantic parser along with the execution model. In our case, the execution model is a program sequence constructed from a set of theorem applications. Thus, our work provides a way to integrate semantic parsing with probabilistic programming. This integration has been pursued before for science diagram question-answering on food-web networks (Krishnamurthy et al., 2016) – which is closely related to our work. Technically, our deductive solver and the approach of learning from different kinds of supervision are the same as the execution model in Krishnamurthy et al. (2016). While Krishnamurthy et al. (2016) only has two program encodings, our work involves a much larger number of programs. We also provide an approach for learning from demonstrations.

8 Conclusion

We described an approach that learns to solve SAT style geometry problems using detailed demonstrative solutions in natural language. The approach learns to jointly interpret demonstrations as well as how to use this interpretation to deductively solve geometry problems using axiomatic knowledge. Our approach showed significant improvements over the best previously published work on a number of datasets. A user-study conducted on a number of school students studying geometry found our approach to be more *interpretable* and *useful* than its predecessors. In the future, we would like to extend our work in other domains such as science QA (Jansen et al., 2016) and use our work to assist student learning on platforms such as MOOCs.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the following research grants: NSF IIS1447676, ONR N000141410684 and ONR N000141712463.

References

Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, page 1.

R.L. Allington and P.M. Cunningham. 2010. Children benefit from modeling, demonstration, and explanation .

Noriko H Arai and Takuya Matsuzaki. 2014. The impact of ai on education–can a robot get into the university of tokyo? In *Proc. ICCE*. pages 1034–1042.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57(5):469–483.

Isabelle Augenstein, Andreas Vlachos, and Diana Maynard. 2015. Extracting relations between non-standard entities using distant supervision and imitation learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 747–757.

Daniel Beck, Andreas Vlachos, Gustavo Paetzold, and Lucia Specia. 2016. SHEF-MIME: word-level quality estimation using imitation learning. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*. pages 772–776.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1533–1544.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics* 3:545–558.

Daphna Buchsbaum, Alison Gopnik, Thomas L Griffiths, and Patrick Shafto. 2011. ChildrenâĂŹs imitation of causal action sequences is influenced by statistical and pedagogical evidence. *Cognition* 120(3):331–340.

John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, et al. 2001. Issues, tasks and program structures to roadmap research in question & answering (q&a) .

Lucas P Butler and Ellen M Markman. 2014. Preschoolers use pedagogical cues to guide radical reorganization of category knowledge. *Cognition* 130(1):116–127.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*. pages 859–865.

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

Peter Clark and Oren Etzioni. 2016. My computer is an honor student - but how intelligent is it? standardized tests as a measure of ai. In *Proceedings of AI Magazine*.

Tom Davis. 2006. Geometry with computers. Technical report.

Edward A Feigenbaum and Julian Feldman. 1963. *Computers and thought*. The AAAI Press.

Richard M Felder, Donald R Woods, James E Stice, and Armando Rugarcia. 2000. The future of engineering education ii. teaching methods that work. *Chemical Engineering Education* pages 26–39.

Akira Fujita, Akihiro Kameda, Ai Kawazoe, and Yusuke Miyao. 2014. Overview of todai robot project and evaluation framework of its nlp-based problem solving. *World History* 36:36.

Dan Goldwasser and Dan Roth. 2014. Learning from natural instructions. *Machine Learning* 94(2):205–232.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016a. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016b. Ucl+ sheffield at semeval-2016 task 8: Imitation learning for amr parsing with an α-bound. *Proceedings of SemEval* pages 1167–1172.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*. pages 851–861. http://aclweb.org/anthology/N/N15/N15-1086.pdf.

Peter Jansen, Niranjan Balasubramanian, Mihai Surdeanu, and Peter Clark. 2016. What's in an explanation? characterizing knowledge and inference requirements for elementary science exams. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 2956–2965. http://aclweb.org/anthology/C/C16/C16-1278.pdf.

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

Jayant Krishnamurthy, Oyvind Tafjord, and Aniruddha Kembhavi. 2016. Semantic parsing to probabilistic programs for situated question answering. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. The Association for Computational Linguistics, pages 160–170.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 590–599.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Cynthia Matuszek, Dieter Fox, and Karl Koscher. 2010. Following directions using statistical machine translation. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, pages 251–258.

Andrew N Meltzoff. 1995. Understanding the intentions of others: re-enactment of intended acts by 18-month-old children. *Developmental psychology* 31(5):838.

Andrew N. Meltzoff and M. Keith Moore. 1977. Imitation of facial and manual gestures by human neonates. *Science* 198(4312):75–78. https://doi.org/10.1126/science.198.4312.75.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. pages 627–635.

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

Stefan Schaal. 1997. Learning from demonstration. In M. I. Jordan and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, MIT Press, pages 1040–1046.

Stefan Schaal. 1999. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences* 3(6):233–242.

Doris Schattschneider and James King. 1997. *Geometry Turned On: Dynamic Software in Learning, Teaching, and Research*. Mathematical Association of America Notes.

Carissa Schoenick, Peter Clark, Oyvind Tafjord, Peter D. Turney, and Oren Etzioni. 2016. Moving beyond the turing test with the allen AI science challenge. *CoRR* abs/1604.04315. http://arxiv.org/abs/1604.04315.

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

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: combining text and diagram interpretation. In *Proceedings of EMNLP*.

Nobuyuki Shimizu and Andrew R. Haas. 2009. Learning to follow navigational route instructions. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*. pages 1488–1493.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics* 2:547–559.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*. Association for Computational Linguistics, Montréal, Canada, pages 441–448. http://www.aclweb.org/anthology/S12-1060.

John M. Zelle and Raymond J. Mooney. 1993. 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.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*.