Solving Geometry Problems: Combining Text and Diagram Interpretation

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, Clint Malcolm
University of Washington, Allen Institute for Artificial Intelligence
`{minjoon,hannaneh,clintm}@washington.edu,{alif,orene}@allenai.org`

Abstract

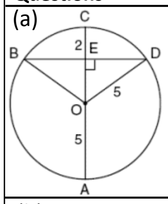
This paper introduces GeoS, the first automated system to solve unaltered SAT geometry questions by combining text understanding and diagram interpretation. We model the problem of understanding geometry questions as submodular optimization, and identify a formal problem description likely to be compatible with both the question text and diagram. GeoS then feeds the description to a geometric solver that attempts to determine the correct answer. In our experiments, GeoS achieves a 49% score on official SAT questions, and a score of 61% on practice questions.[1] Finally, we show that by integrating textual and visual information, GeoS boosts the accuracy of dependency and semantic parsing of the question text.

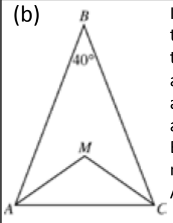
1 Introduction

This paper introduces the first fully-automated system for solving unaletered SAT-level geometric word problems, each of which consists of text and the corresponding diagram (Figure 1). The geometry domain has a long history in AI, but previous work has focused on geometric theorem proving (Feigenbaum and Feldman, 1963) or geometric analogies (Evans, 1964). Arithmetic and algebraic word problems have attracted several NLP researchers (Kushman et al., 2014; Hosseini et al., 2014; Roy et al., 2015), but *geometric* word problems were first explored only last year by Seo et al. (2014). Still, this system merely aligned diagram elements with their textual mentions (e.g., "Circle O")—it did not attempt to fully represent geometry problems or solve them. Answering geometry questions requires a method that interpert question text and diagrams in concert.

[1]The source code, the dataset and the annotations are publicly available at `geometry.allenai.org`.

Questions		Interpretations

(a) 	In the diagram at the left, circle O has a radius of 5, and CE = 2. Diameter AC is perpendicular to chord BD. What is the length of BD?	*Equals(RadiusOf(O), 5)*
		IsCircle(O)
		Equals(LengthOf(CE), 2)
		IsDiameter(AC)
		IsChord(BD)
		Perpendicular(AC), BD)
		Equals(what, Length(BD))
		correct
		a) 12 b) 10 [✓ c) 8] d) 6 e) 4

(b) 	In isosceles triangle ABC at the left, lines AM and CM are the angle bisectors of angles BAC and BCA. What is the measure of angle AMC?	*IsIsoscelesTriangle(ABC)*
		BisectsAngle(AM, BAC)
		IsLine(AM)
		CC(AM, CM)
		CC(BAC, BCA)
		IsAngle(BAC)
		IsAngle(AMC)
		Equals(what, MeasureOf(AMC))
		correct
		[✓ a) 110] b) 115 c) 120 d) 125 e) 130

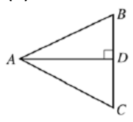
(c) 	In the figure at left, The bisector of angle BAC is perpendicular to BC at point D. If AB = 6 and BD = 3, what is the measure of angle BAC?	*IsAngle(BAC)*
		BisectsAngle(line, BAC)
		Perpendicular (line, BC)
		Equals(LengthOf(AB), 6)
		Equals(LengthOf(BD), 3)
		IsAngle(BAC)
		Equals(what, MeasureOf(BAC))
		correct
		a) 15 b) 30 c) 45 [✓ d) 60] e) 75

Figure 1: Questions (left column) and interpretations (right column) derived by GeoS.

The geometry genre has several distinctive characteristics. First, diagrams provide essential information absent from question text. In Figure 1 problem (a), for example, the unstated fact that lines BD and AC intersect at E is necessary to solve the problem. Second, the text often includes difficult references to diagram elements. For example, in the sentence "In the diagram, the longer line is tangent to the circle", resolving the referent of the phrase "longer line" is challenging. Third, the text often contains implicit relations. For example, in the sentence "AB is 5", the relations `IsLine(AB)` and `length(AB)=5` are implicit. Fourth, geometric terms can be ambiguous as well. For instance, radius can be a type identifier in "the length of radius AO is 5", or a predicate in "AO is the radius of circle O". Fifth, identifying the correct arguments for each relation is challenging. For example, in sentence "Lines AB and CD are perpendicular to EF", the parser has to determine *what* is perpendicular to EF—line AB? line

CD? Or both AB and CD? Finally, it is hard to obtain large number of SAT-level geometry questions; Learning from a few examples makes this a particularly challenging NLP problem.

This paper introduces GEOS, a system that maps geometry word problems into a logical representation that is compatible with both the problem text and the accompanying diagram (Figure 1). We cast the mapping problem as the problem of selecting the subset of relations that is most likely to correspond to each question.

We compute the mapping in three main steps (Figure 2). First, GEOS uses text- and diagram-parsing to overgenerate a set of relations that potentially correspond to the question text, and associates a score with each. Second, GEOS generates a set of relations (with scores) that corresponds to the diagram. Third, GEOS selects a subset of the relations that maximizes the joint text and diagram scores. We cast this maximization as a submodular optimization problem, which enables GEOS to use a close-to-optimal greedy algorithm. Finally, we feed the derived formal model of the problem to a geometric solver, which computes the answer to the question.

GEOS is able to solve *unseen* and *unaltered* multiple-choice geometry questions. We report on experiments where GEOS achieves a 49% score on official SAT questions, and a score of 61% on practice questions, providing the first results of this kind. Our contributions include: (1) designing and implementing the first end-to-end system that solves SAT plane geometry problems; (2) formalizing the problem of interpreting geometry questions as a submodular optimization problem; and (3) providing the first empirical results on the geometry genre, making the data and software available for future work.

2 Related Work

Semantic parsing is an important area of NLP research (Zettlemoyer and Collins, 2005; Ge and Mooney, 2006; Flanigan et al., 2014; Eisenstein et al., 2009; Kate and Mooney, 2007; Goldwasser and Roth, 2011; Poon and Domingos, 2009; Berant and Liang, 2014; Kwiatkowski et al., 2013; Reddy et al., 2014). However, semantic parsers do not tackle diagrams—a critical element of the geometry genre. In addition, the overall number of available geometry questions is quite small compared to the size of typical NLP corpora, making it challenging to learn semantic parsers directly from geometry questions. Relation extraction is another area of NLP that is related to our task (Cowie and Lehnert, 1996; Culotta and Sorensen, 2004). Again, both diagrams and small corpora are problematic for this body of work.

Our work is part of grounded language acquisition research (Branavan et al., 2012; Vogel and Jurafsky, 2010; Chen et al., 2010; Hajishirzi et al., 2011; Liang et al., 2009; Koncel-Kedziorski et al., 2014; Bordes et al., 2010; Kim and Mooney, 2013; Angeli and Manning, 2014; Hixon et al., 2015; Koncel-Kedziorski et al., 2014; Artzi and Zettlemoyer, 2013) that involves mapping text to a restricted formalism (instead of a full, domain independent representation). In the geometry domain, we recover the entities (e.g., circles) from diagrams, derive relations compatible with both text and diagram, and re-score relations derived from text parsing using diagram information. Casting the interpretation problem as selecting the most likely subset of literals can be generalized to grounded semantic parsing domains such as navigational instructions.

Coupling images and the corresponding text has attracted attention in both vision and NLP (Farhadi et al., 2010; Kulkarni et al., 2011; Gupta and Mooney, 2010; Gong et al., 2014; Fang et al., 2014). We build on this powerful paradigm, but instead of generating captions we show how processing multimodal information help improve textual or visual interpretations for solving geometry questions.

Diagram understanding has been explored since early days in AI (Lin et al., 1985; Hegarty and Just, 1989; Novak, 1995; O'Gorman and Kasturi, 1995; Bulko, 1988; Srihari, 1994; Lovett and Forbus, 2012). Most previous approaches differ from our method because they address the twin problems of diagram understanding and text understanding in isolation. Often, previous work relies on manual identification of visual primitives, or on rule-based system for text analysis. The closest work to ours is the recent work of Seo et al. (2014) that aligns geometric shapes with their textual mentions, but does not identify geometric relations or solve geometry problems.

3 Problem Formulation

A geometry question is a tuple (t, d, c) consisting of a text t in natural language, a diagram d

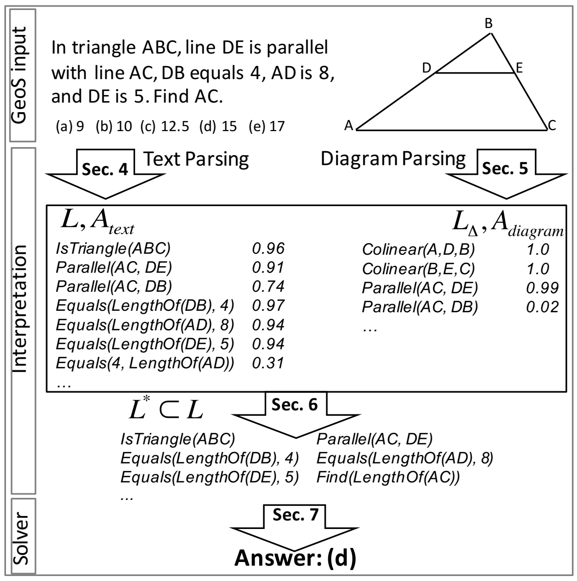


Figure 2: Overview of our method for solving geometry questions.

in raster graphics, and multiple choice answers $c = \{c_1, \dots, c_M\}$ ($M = 5$ in SAT). Answering a geometry question is to find a correct choice c_i.

Our method, GEOS, consists of two steps (Figure 2): (1) *interpreting* a geometry question by deriving a logical expression that represents the meaning of the text and the diagram, and (2) *solving* the geometry question by checking the satisfiablity of the derived logical expression. In this paper we mainly focus on *interpreting* geometry questions and use a standard algebraic solver (see section 7 for a brief description of the solver).

Definitions: We formally represent logical expressions in the geometry domain with the language Ω, a subset of typed first-order logic that includes:

- *constants*, corresponding to known numbers (e.g., `5` and `2` in Figure 1) or entities with known geometric coordinates.
- *variables*, corresponding to unknown numbers or geometrical entities in the question (e.g., `O` and `CE` in Figure 1).
- *predicates*, corresponding to geometric or arithmetic relations (e.g., `Equals`, `IsDiameter`, `IsTangent`).
- *functions*, corresponding to properties of geometrical entities (e.g., `LengthOf`, `AreaOf`) or arithmetic operations (e.g., `SumOf`, `RatioOf`).

Each element in the geometry language has either boolean (e.g., `true`), numeric (e.g., `4`), or entity (e.g., `line`, `circle`) type. We refer to all symbols in the language Ω as *concepts*.

We use the term *literal* to refer to the application of a predicate to a sequence of arguments (e.g., `IsTriangle(ABC)`). Literals are possibly negated atomic formulas in the language Ω. Logical formulas contain constants, variables, functions, existential quantifiers and conjunctions over literals (e.g., $\exists$`x, IsTriangle(x)`$\wedge$`IsIsosceles(x)`).

Interpretation is the task of mapping a new geometry question with each choice, (t, d, c_m), into a logical formula γ in Ω. More formally, the goal is to find $\gamma^* = \arg\max_{\gamma \in \Gamma} score(\gamma; t, d, c_m)$ where Γ is the set of all logical formulas in Ω and $score$ measures the interpretation score of the formula according to both text and diagram. The problem of deriving the best formula γ^* can be modeled as a combinatorial search in the space of literals L (note that each logical formula γ is represented as a conjunction over literals l_i).

GEOS efficiently searches this combinatorial space taking advantage of a submodular set function that scores a subset of literals using both text and diagram. The best subset of literals is the one that has a high *affinity* with both text and diagram and is *coherent* i.e., does not suffer from redundancy (see Section 6). More formally,[2]

$$L^* = \arg\max_{L' \subset L} \lambda \underbrace{\mathcal{A}(L', t, d)}_{\text{Affinity}} + \underbrace{\mathcal{H}(L', t, d)}_{\text{Coherence}}, \quad (1)$$

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

The affinity $\mathcal{A}$ is decomposed into text-based affinity, $\mathcal{A}_{text}$, and diagram-based affinity, $\mathcal{A}_{diagram}$. The text-based affinity closely mirrors the linguistic structure of the sentences as well as type matches in the geometry language Ω. For modeling the text score for each literal, we learn a log-linear model. The diagram-based affinity $\mathcal{A}_{diagram}$ grounds literals into the diagram, and scores literals according to the diagram parse. We describe the details on how to compute $\mathcal{A}_{text}$ in section 4 and $\mathcal{A}_{diagram}$ in section 5.

4 Text Parser

The text-based scoring function $\mathcal{A}_{text}(L, t)$ computes the affinity score between the set of liter-

[2]We omit the argument c_m for the ease of notation.

```
1. $\{l_j\}, \mathcal{A}_{text} \leftarrow$ TEXT PARSING(language $\Omega$, text-choice pair $(t, c_i)$)    (Section 4)
     (i) concept identification: initialize a hypergraph $G$ with concept nodes.
    (ii) relation identification: add a hyperedge (relation) $r_j$ between two or three related concept nodes and assign a
         weight $\mathcal{A}_{text}(r_j, t; \theta)$ based on the learned classifier.
   (iii) literals parsing: obtain all subtrees of $G$, which are equivalent to all possible literals, $\{l'_j\}$. Let $\mathcal{A}_{text}(l_j, t) =$
         $\sum_j \mathcal{A}_{text}(r_j, t; \theta)$ for all $r_j$ in the literal $l_i$.
    (iv) relation completion: obtain a complete literal $l_j$ for each (under-specified) $l'_j$, dealing with implication and
         coordinating conjunctions.
2. $L_\Delta, \mathcal{A}_{diagram} \leftarrow$ DIAGRAM PARSING(diagram image $d$, literals $\{l_j\}$)    (Section 5)
3. $L^* \leftarrow$ GREEDY MAXIMIZATION(literals $L = \{l_j\}$, score functions $\mathcal{A}_{text}$ and $\mathcal{A}_{diagram}$)    (Section 6)
     (i) initialization: $L' \leftarrow \{\}$
    (ii) greedy addition: $add(L', l_j)$ s.t. $l_j = \text{argmax}_{l_j \in L \setminus L'} \mathcal{F}(L' \cup \{l_j\}) - \mathcal{F}(L')$, where $\mathcal{F} = \lambda\mathcal{A} + \mathcal{H}$
   (iii) iteration: repeat step (ii) while the gain is positive.
4. Answer $c^* \leftarrow$ one of choices s.t. $L^* \cup L_\Delta$ are simultaneously satisfiable according to SOLVER    (Section 7)
```

Figure 3: Solving geometry questions with GEOS.

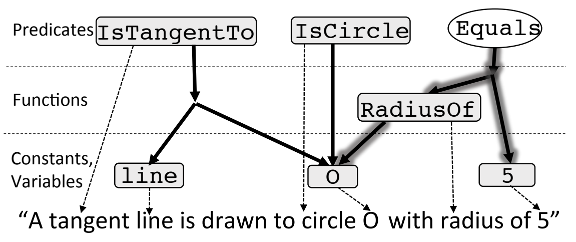


Figure 4: Hypergraph representation of the sentence "A tangent line is drawn to circle O with radius of 5".

als L and the question text t. This score is the sum of the affinity scores of individual literals $l_j \in L$ i.e., $\mathcal{A}_{text}(L, t) = \sum_j \mathcal{A}_{text}(l_j, t)$ where $\mathcal{A}_{text}(l_j, t) \mapsto [-\infty, 0]$.[3] GEOS learns a discriminative model $\mathcal{A}_{text}(l_j, t; \theta)$ that scores the affinity of every literal $l_j \in L$ and the question text t through supervised learning from training data.

We represent literals using a hypergraph (Figure 4) (Klein and Manning, 2005; Flanigan et al., 2014). Each node in the graph corresponds to a concept in the geometry language (i.e. constants, variables, functions, or predicates). The edges capture the *relations* between concepts; concept nodes are connected if one concept is the argument of the other in the geometry language. In order to interpret the question text (Figure 3 step 1), GEOS first identifies concepts evoked by the words or phrases in the input text. Then, it learns the affinity scores which are the weights of edges in the hypergraph. It finally completes relations so that type matches are satistfied in the formal language.

4.1 Concept Identification

Concepts are defined as symbols in the geometry language Ω. The concept identification stage maps words or phrases to their corresponding concepts in the geometry language. Note that a phrase can be mapped to several concepts. For instance, in the sentence "ABCD is a square with an area of 1", the word "square" is a noun referring to some object, so it maps to a variable `square`. In a similar sentence "square ABCD has an area 1", the word "square" describes the variable `ABCD`, so it maps to a predicate `IsSquare`.

GEOS builds a *lexicon* from training data that maps stemmed words and phrases to the concepts in the geometry language Ω. The lexicon is derived from all correspondences between geometry keywords and concepts in the geometry language as well as phrases and concepts from manual annotations in the training data. For instance, the lexicon contains ("square", $\{$`square`, `IsSquare`$\}$) including all possible concepts for the phrase "square". Note that GEOS does not make any hard decision on which identification is correct in this stage, and defers it to the relation identification stage (Section 4.2). To identify numbers and explicit variables (e.g. "5", "AB", "O"), GEOS uses regular expressions. For an input text t, GEOS assigns one node in the graph (Figure 4) for each concept identified by the lexicon.

4.2 Relation Identification

A relation is a directed hyperedge between concept nodes. A hyperedge connects two nodes (for unary relations such as the edge between `RadiusOf` and `O` in Figure 4) or three nodes (for binary relations such as the hyperedge between `Equals` and its two arguments `RadiusOf` and `5` in Figure 4).

We use a discriminative model (logistic regression) to predict the probability of a relation r_i being correct in text t: $P_\theta(y_i|r_i, t) = \frac{1}{1+\exp(f_{text}(r_i,t)\cdot\theta)}$, where $y_i \in \{0, 1\}$ is the label

[3]For the ease of notation, we use $\mathcal{A}_{text}$ as a function taking sets of literals or a literal.

Dependency tree distance	Shortest distance between the words of the concept nodes in the dependency tree. We use indicator features for distances of -3 to 3. Positive distance shows if the child word is at the right of the parent's in the sentence, and negative otherwise.
Word distance	Distance between the words of the concept nodes in the sentence.
Dependency tree edge label	Indicator functions for the 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`.

Table 1: The features of the unary relations. The features of the binary relations is computed in a similar way.

(a)	sentence:	"What is the perimeter of ABCE?"
	intermediate:	`∃ what, ABCE: Bridged(what, PerimeterOf(ABCE))`
	final:	`∃ what, ABCE: Equals(what, PerimeterOf(ABCE))`
(b)	sentence:	"AM and CM bisect BAC and BCA."
	intermediate:	`∃ AM, CM, BAC, BCA: BisectsAngle(AM, BAC) ∧ CC(AM, CM) ∧ CC(BAC, BCA)`
	final:	`∃ AM, CM, BAC, BCA: BisectsAngle(AM, BAC) ∧ BisectsAngle(CM, BCA)`

Figure 5: Showing the two-stage learning with the intermediate representation that demonstrates implication.

for r_i being correct in t, $f_{text}(r_i, t)$ is a feature vector of t and r_i, and θ is a vector of parameters to be learned. We define the affinity score of r_i by $\mathcal{A}_{text}(r_i, t; \theta) = \log P_\theta(y_i|r_i, t)$. The weight of the corresponding hyperedge is the relation's affinity score. We learn θ using the maximum likelihood estimation of the training data (details in Section 8), with L2 regularization.

We train two separate models for learning unary and binary relations. The training data consists of sentence-relation-label tuples (t, r, y); for instance, ("A tangent line is drawn to circle O", `IsTangent(line, O)`, 1) is a positive training example. All incorrect relations in the sentences of the training data are negative examples (e.g. ("A tangent line is drawn to circle O", `IsCircle(line)`, 0)).

The features for the unary and binary models are shown in Table 1 for the text t and the relation r_i. We use two main feature categories. Structural features: these features capture the syntactic cues of the text in the form of text distance, dependency tree labels, and part of speech tags for the words associated with the concepts in the relation. Geometry language features: these features capture the cues available in the geometry language Ω in the form of the types and the truth values of the corresponding concepts in the relation.

At inference, GEOS uses the learned models to calculate the affinity scores of all the literals derived from the text t. The affinity score of each literal l_j is calculated from the edge (relation) weights in the corresponding subgraph, i.e. $\mathcal{A}_{text}(l_j, t) = \sum_i \mathcal{A}_{text}(r_i, t; \theta)$ for all r_i in the literal l_j.

4.3 Relation Completion

So far, we have explained how to score the affinities between explicit relations and the question text. Geometry questions usually include *implicit* concepts. For instance, "Circle O has a radius of 5" implies the `Equals` relationship between "Radius of circle O" and "5". In addition, geometry questions include *coordinating conjunctions* between entities. In "AM and CM bisect BAC and BCA", "bisect" is shared by two lines and two angles (Figure 5 (b)). Also, consider two sentences: "AB and CD are perpendicular" and "AB is perpendicular to CD". Both have the same semantic annotation but very different syntactic structures.

It is difficult to directly fit the syntactic structure of question sentences into the formal language Ω for *implications* and *coordinating conjunctions*, especially due to small training data. We, instead, adopt a two-stage learning inspired by recent work in semantic parsing (Kwiatkowski et al., 2013). Our solution assumes an intermediate representation that is syntactically sound but possibly *underspecified*. The intermediate representation closely mirrors the linguistic structure of the sentences. In addition, it can easily be transferred to the formal representation in the geometry language Ω.

Figure 5 shows how implications and coordinating conjunctions are modeled in the intermediate representation. `Bridged` in Figure 5 (a) indicates that there is a special relation (edge) between the two concepts (e.g., `what` and `PerimeterOf`), but the alignment to the geometry language $\mathcal{L}$ is not clear. `CC` in Figure 5 (b) indicates that there is a special relation between two concepts that are connected by "and" in the sentence. GEOS *completes* the under-specified relations by mapping them to the corresponding well-defined relations in the formal language.

Implication: We train a log-linear classifier to identify if a `Bridged` relation (implied concept) exists between two concepts. Intuitively, the classification score indicates the likelihood that certain two concepts (e.g., `What` and `PerimeterOf`) are *bridged*. For training, positive examples are pairs of concepts whose underlying relation is under-specified, and negative examples are all other pairs of concepts that are not bridged. For instance, (`what`, `PerimeterOf`) is a positive training example for the `bridged` relation. We use the same features in Table 1 for the classifier.

We then use a deterministic rule to map *bridged* relations in the intermediate representation to the correct *completed* relations in the final representation. In particular, we map `bridged` to `Equals` if the two children concepts are of type `number`, and to `IsA` if the concepts are of type entity (e.g. `point`, `line`, `circle`).

Coordinating Conjunctions: `CC` relations model coordinating conjunctions in the intermediate representation. For example, Figure 5 (b) shows the conjunction between the two angles `BAC` and `BCA`. We train a log-linear classifier for the `CC` relations, where the setup of the model is identical to that of the binary relation model in Section 4.2.

After we obtain a list of `CC(x,y)` in the intermediate representation, we use deterministic rules to coordinate the entities x and y in each `CC` relation (Figure 5 (b)). First, GEOS forms a set $\{x, y\}$ for every two concepts x and y that appear in `CC(x,y)` and transforms every x and y in other literals to $\{x, y\}$. Second, GEOS transforms the relations with expansion and distribution rules (Figure 3 Step 1 (iv)). For instance, `Perpendicular({x,y})` will be transferred to `Perpendicular(x, y)` (expansion rule), and `LengthOf{x,y})` will be transferred to `LengthOf(x)` $\wedge$ `LengthOf(y)` (distribution rule).

5 Diagram Parser

We use the publicly available diagram parser (Seo et al., 2014) to obtain the set of all visual elements (points, lines, circles, etc.), their coordinates, their relationships in the diagram, and their alignment with entity references in the text (e.g. "line AB", "circle O"). The diagram parser serves two purposes: (a) computing the diagram score as a measure of the affinity of each literal with the diagram; (b) obtaining high-confidence visual literals which cannot be obtained from the text.

Diagram score: For each literal l_j from the text parsing, we obtain its diagram score $\mathcal{A}_{diagram}(l_j, d) \mapsto [-\infty, 0]$. GEOS grounds each literal derived from the text by replacing every variable (entity or numerical variable) in the relation to the corresponding variable from the diagram parse. The score function is the relaxed indicator function of whether a literal is true according to the diagram. For instance, in Figure 1 (a), consider the literal $l =$ `Perpendicular(AC, BD)`. In order to obtain its diagram score, we compute the angle between the lines AC and BD in the diagram and compare it with $\pi/2$. The closer the two values, the higher the score (closer to 0), and the farther they are, the lower the score. Note that the variables `AC` and `BD` are grounded into the diagram before we obtain the score; that is, they are matched with the actual corresponding lines AC and BD in the diagram.

The diagram parser is not able to evaluate the correctness of some literals, in which case their diagram scores are *undefined*. For instance, `Equals(LengthOf(AB), 5)` cannot be evaluated in the diagram because the scales in the diagram (pixel) and the text are different. For another example, `Equals(what, RadiusOf(circle))` cannot be evaluated because it contains an ungrounded (query) variable, `what`. When the diagram score of a literal l_j is undefined, GEOS lets $\mathcal{A}_{diagram}(l_j) = \mathcal{A}_{text}(l_j)$.

If the diagram score of a literal is very low, then it is highly likely that the literal is false. For example, in Figure 2, `Parallel(AC, DB)` has a very low diagram score, 0.02, and is apparently false in the diagram. Concretely, if for some literal l_j, $\mathcal{A}_{diagram}(l_i) < \epsilon$, then GEOS disregards the text score of l_i by replacing $\mathcal{A}_{text}(l_j)$ with $\mathcal{A}_{diagram}(l_j)$. On the other hand, even if the diagram score of a literal is very high, it is still possible that the literal is false, because many diagrams are not drawn to scale. Hence, GEOS adds both text and diagram scores in order to score literals (Section 6).

High-confidence visual literals: Diagrams often contain critical information that is not present in the text. For instance, to solve the question in Figure 1, one has to know that the points `A`, `E`, and `C` are colinear. In addition, diagrams include numerical labels (e.g. one of the labels in Figure 1(b) indicates the measure of the angle ABC = 40 degrees). This kind of information is confidently parsed with the diagram parser by Seo et al. (2014). We denote the set of the high-confidence literals by L_Δ that are passed to the solver (Section 7).

6 Optimization

Here, we describe the details of the objective function (Equation 1) and how to efficiently maximize it. The integrated affinity score of a set of literals L' (the first term in Equation 1) is defined as:

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

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

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

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

where N_{covered} is the number of the concept nodes used by the literals in L', and $N_{\text{redundant}}$ is the number of redundancies among the concept nodes of the literals. To account for the different scales between $\mathcal{A}$ and $\mathcal{H}$, we use the trade-off parameter λ in Equation 1 learned on the validation dataset.

Maximizing the objective function in Equation 1 is an NP-hard combinatorial optimization problem. However, we show that our objective function is submodular (see Appendix (Section 11) for the proof of submodularity). This means that there exists a greedy method that can provide a reliable approximation. GEOS greedily maximizes Equation 1 by starting from an empty set of literals and adding the next literal l_j that maximizes the gain of the objective function until the gain becomes negative (details of the algorithm and the gain function are explained in Figure 3 step 3).

7 Solver

We now have the best set of literals L^* from the optimization, and the high-confidence visual literals L_Δ from the diagram parser. In this step, GEOS determines if an assignment exists to the variables X in $L^* \cup L_\Delta$ that simultaneously satisfies all of the literals. This is known as the problem of automated geometry theorem proving in computational geometry (Alvin et al., 2014).

We use a numerical method to check the satisfiablity of literals. For each literal l_j in $L^* \cup L_\Delta$, we define a relaxed indicator function $g_j : S \mapsto z_j \in [-\infty, 0]$. The function $z_j = g_j(S)$ indicates the relaxed satisfiability of l_j given an assignment S to the variables X. The literal l_j is completely satisfied if $g_j(S) = 0$. We formulate the problem of satisfiability of literals as the task of finding the assignment S^* to X such that sum of all indicator functions $g_j(S^*)$ is maximized, i.e. $S^* = \arg\max_S \sum_j g_j(S)$. We use the basing-hopping algorithm (Wales and Doye, 1997) with sequential least squares programming (Kraft, 1988) to globally maximize the sum of the indicator functions. If there exists an assignment such that $\sum_j g_j(S) = 0$, then GEOS finds an assignment to X that satisfies all literals. If such assignment does not exist, then GEOS concludes that the literals are not satisfiable simultaneously. GEOS chooses to answer a geometry question if the literals of exactly one answer choice are simultaneously satisfiable.

8 Experimental Setup

Logical Language Ω: Ω consists of 13 types of entities and 94 function and predicates observed in our development set of geometry questions.

Implementation details: Sentences in geometry questions often contain in-line mathematical expressions, such as "If AB=x+5, what is x?". These mathematical expressions cause general purpose parsers to fail. GEOS uses an equation analyzer and pre-processes question text by replacing "=" with "equals", and replacing mathematical terms (e.g., "x+5") with a dummy noun so that the dependency parser does not fail.

GEOS uses Stanford dependency parser (Chen and Manning, 2014) to obtain syntactic information, which is used to compute features for relation identification (Table 1). For diagram parsing, similar to Seo et al. (2014), we assume that GEOS has access to ground truth optical character recognition for labels in the diagrams. For optimization, we tune the parameters λ to 0.5, based on the training examples.[4]

Dataset: We built a dataset of SAT plane geometry questions where every question has a tex-

[4]In our dataset, the number of all possible literals for each sentence is at most 1000.

	Total	Training	Practice	Official
Questions	186	67	64	55
Sentences	336	121	110	105
Words	4343	1435	1310	1598
Literals	577	176	189	212
Binary relations	337	110	108	119
Unary relations	437	141	150	146

Table 2: Data and annotation statistics

tual description in English accompanied by a diagram and multiple choices. Questions and answers are compiled from previous official SAT exams and practice exams offered by the College Board (Board, 2014). In addition, we use a portion of the publicly available high-school plane geometry questions (Seo et al., 2014) as our training set.

We annotate ground-truth logical forms for all questions in the dataset. Table 2 shows details of the data and annotation statistics. For evaluating dependency parsing, we annotate 50 questions with the ground truth dependency tree structures of all sentences in the questions. [5]

Baselines: *Rule-based text parsing* + GEOS *diagram* solves geometry questions using literals extracted from a manually defined set of rules over the textual dependency parser, and scored by diagram. For this baseline, we manually designed 12 high-precision rules based on the development set. Each rule compares the dependency tree of each sentence to pre-defined templates, and if a template pattern is matched, the rule outputs the relation or function structure corresponding to that template. For example, a rule assigns a relation *parent(child-1, child-2)* for a triplet of *(parent, child-1, child-2)* where *child-1* is the subject of *parent* and *child-2* is the object of the parent.

GEOS *without text parsing* solves geometry questions using a simple heuristic. With simple textual processing, this baseline extracts numerical relations from the question text and then computes the scale between the units in the question and the pixels in the diagram. This baseline rounds the number to the closest choice available in the multiple choices.

GEOS *without diagram parsing* solves geometry questions only relying on the literals interpreted from the text. It outputs all literals whose text scores are higher than a tuned threshold, 0.6 on the training set.

GEOS *without relation completion* solves geometry questions when text parsing does not use the intermediate representation and does not include the relation completion step.

9 Experiments

We evaluate our method on three tasks: solving geometry question, interpreting geometry questions, and dependency parsing.

Solving Geometry Questions: Table 3 compares the score of GEOS in solving geometry questions in practice and official SAT questions with that of baselines. SAT's grading scheme penalizes a wrong answer with a negative score of 0.25. We report the SAT score as the percentage of correctly answered questions penalized by the wrong answers. For official questions, GEOS answers 27 questions correctly, 1 questions incorrectly, and leaves 27 un-answered, which gives it a score of 26.75 out of 55, or 49%. Thus, GEOS's precision exceeds 96% on the 51% of questions that it chooses to answer. For practice SAT questions, GEOS scores 61%.[6]

In order to understand the effect of individual components of GEOS, we compare the full method with a few ablations. GEOS significantly outperforms the two baselines GEOS *without text parsing* and GEOS *without diagram parsing*, demonstrating that GEOS benefits from both text and diagram parsing. In order to understand the text parsing component, we compare GEOS with *Rule-based text parsing* + GEOS *Diagram* and GEOS *without relation completion*. The results show that our method of learning to interpret literals from the text is substantially better than the rule-based baseline. In addition, the relation completion step, which relies on the intermediate representation, helps to improve text interpretation.

Error Analysis: In order to understand the errors made by GEOS, we use *oracle text parsing* and *oracle diagram parsing* (Table 3). Roughly 38% of the errors are due to failures in text parsing, and about 46% of errors are due to failures in diagram parsing. Among them, about 15% of errors were due to failures in both diagram and text parsing. For an example of text parsing failure, the literals in Figure 6 (a) are not scored accurately due to missing coreference relations (Hajishirzi et al., 2013). The rest of errors are due to problems that require more complex reasoning (Figure 6 (b)).

[5]The source code, the dataset and the annotations are publicly available at `geometry.allenai.org`.

[6]Typically, 50th percentile (penalized) score in SAT math section is 27 out of 54 (50%).

Method	SAT score (%) Practice	Official
GEOS w/o diagram parsing	7	5
GEOS w/o text parsing	10	10
Rule-based text parsing + GEOS diagram	31	24
GEOS w/o relation completion	42	33
GEOS	**61**	**49**
Oracle text parsing + GEOS diagram parsing	78	75
GEOS text parsing + oracle diagram parsing	81	79
Oracle text parsing + oracle diagram parsing	88	84

Table 3: SAT scores of solving geometry questions.

	P	R	F1
Rule-based text parsing	**0.99**	0.23	0.37
GEOS w/o diagram	0.57	**0.82**	0.67
GEOS	0.92	0.76	**0.83**

Table 4: Precision and recall of text interpretation.

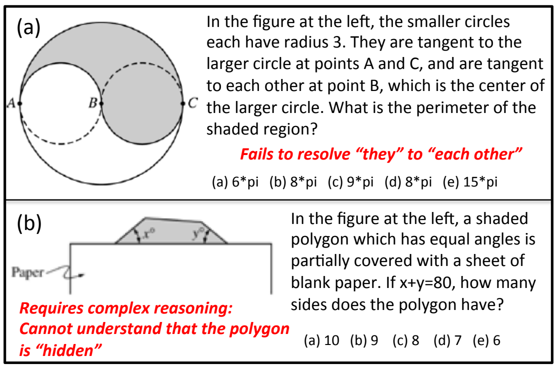
Interpreting Question Texts: Table 4 details the precision and recall of GEOS in deriving literals for geometry question texts for official SAT questions. The rule-based text parsing baseline achieves a high precision, but at the cost of lower recall. On the other hand, the baseline GEOS *without diagram* achieves a high recall, but at the cost of lower precision. Nevertheless, GEOS attains substantially higher F1 score compared to both baselines, which is the key factor in solving the questions. Direct application of a generic semantic parser (Berant et al., 2013) with full supervision does not perform well in the geometry domain, mainly due to lack of enough training data. Our initial investigations show the performance of 33% F1 in the official set.

Improving Dependency Parsing: Table 5 shows the results of different methods in dependency parsing. GEOS returns a dependency parse tree by selecting the dependency tree that maximizes the text score in the objective function from the top 50 trees produced by a generic dependency parser, Stanford parser (Chen and Manning, 2014). Note that Stanford parser cannot handle mathematical symbols and equations. We report the results of a baseline that extends the Stanford dependency parser by adding a pre-processing step to separate the mathematical expressions from the plain sentences (Section 8).

We evaluate the performance of GEOS against the best tree returned by Stanford parser by reporting the fraction of the questions whose dependency parse structures match the ground truth annotations. Our results show an improvement of 16% over the Stanford dependency parser when equipped with the equation analyzer. For example, in "AB is perpendicular to CD at E", the Stanford dependency parser predicts that "E" depends on "CD", while GEOS predicts the correct parse in which "E" depends on "perpendicular".

	Accuracy
Stanford dep parse	0.05
Stanford dep parse + eq. analyzer	0.64
GEOS	**0.78**

Table 5: Accuracy of dependency parsing.



In the figure at the left, the smaller circles each have radius 3. They are tangent to the larger circle at points A and C, and are tangent to each other at point B, which is the center of the larger circle. What is the perimeter of the shaded region?

Fails to resolve "they" to "each other"

(a) 6*pi (b) 8*pi (c) 9*pi (d) 8*pi (e) 15*pi

In the figure at the left, a shaded polygon which has equal angles is partially covered with a sheet of blank paper. If x+y=80, how many sides does the polygon have?

Requires complex reasoning: Cannot understand that the polygon is "hidden"

(a) 10 (b) 9 (c) 8 (d) 7 (e) 6

Figure 6: Examples of Failure: reasons are in red.

10 Conclusion

This paper introduced GEOS, an automated system that combines diagram and text interpretation to solve geometry problems. Solving geometry questions was inspired by two important trends in the current NLP literature. The first is in designing methods for grounded language acquisition to map text to a restricted formalism (instead of a full, domain independent representation). We demonstrate a new algorithm for learning to map text to a geometry language with a small amount of training data. The second is designing methods in coupling language and vision and show how processing multimodal information help improve textual or visual interpretations.

Our experiments on unseen SAT geometry problems achieve a score of 49% of official questions and a score of 61% on practice questions, providing a baseline for future work. Future work includes expanding the geometry language and the reasoning to address a broader set of geometry questions, reducing the amount of supervision, learning the relevant geometry knowledge, and scaling up the dataset.

Acknowledgements. The research was supported by the Allen Institute for AI, Allen Distinguished Investigator Award, and NSF (IIS-1352249). We thank Dan Weld, Luke Zettlemoyer, Aria Haghighi, Mark Hopkins, Eunsol Choi, and the anonymous reviewers for helpful comments.

References

Chris Alvin, Sumit Gulwani, Rupak Majumdar, and Supratik Mukhopadhyay. 2014. Synthesis of geometry proof problems. In *AAAI*.

Gabor Angeli and Christopher D. Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *EMNLP*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 1.

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

College Board. 2014. The college board.

Antoine Bordes, Nicolas Usunier, and Jason Weston. 2010. Label ranking under ambiguous supervision for learning semantic correspondences. In *ICML*.

SRK Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning high-level planning from text. In *ACL*.

William C. Bulko. 1988. Understanding text with an accompanying diagram. In *IEA/AIE*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*.

David Chen, Joohyun Kim, and Raymond Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *JAIR*, 37.

Jim Cowie and Wendy Lehnert. 1996. Information extraction. *Communications of the ACM*, 39(1).

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *ACL*.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *EMNLP*.

Thomas G Evans. 1964. A heuristic program to solve geometric-analogy problems. In *Proceedings of the April 21-23, 1964, spring joint computer conference*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, et al. 2014. From captions to visual concepts and back. In *CVPR*.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *ECCV*.

E.A. Feigenbaum and J. Feldman, editors. 1963. *Computers and Thought*. McGraw Hill, New York.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *ACL*.

Ruifang Ge and Raymond J. Mooney. 2006. Discriminative reranking for semantic parsing. In *ACL*.

Dan Goldwasser and Dan Roth. 2011. Learning from natural instructions. In *IJCAI*.

Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. 2014. Improving image-sentence embeddings using large weakly annotated photo collections. In *ECCV*.

Sonal Gupta and Raymond J. Mooney. 2010. Using closed captions as supervision for video activity recognition. In *AAAI*.

Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir. 2011. Reasoning about robocup soccer narratives. In *UAI*.

Hannaneh Hajishirzi, Leila Zilles, Daniel S Weld, and Luke S Zettlemoyer. 2013. Joint coreference resolution and named-entity linking with multi-pass sieves. In *EMNLP*.

Mary Hegarty and Marcel Adam Just. 1989. 10 understanding machines from text and diagrams. *Knowledge acquisition from text and pictures*.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. In *NAACL*.

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

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

Joohyun Kim and Raymond J. Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *ACL*.

Dan Klein and Christopher D Manning. 2005. Parsing and hypergraphs. In *New developments in parsing technology*. Springer.

R Koncel-Kedziorski, Hannaneh Hajishirzi, and Ali Farhadi. 2014. Multi-resolution language grounding with weak supervision. In *EMNLP*.

Dieter et. al. Kraft. 1988. *A software package for sequential quadratic programming*. DFVLR Obersfaffeuhofen, Germany.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2011. Baby talk: Understanding and generating image descriptions. In *CVPR*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *ACL*.

T Kwiatkowski, E Choi, Y Artzi, and L Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACLAFNLP*.

Xinggang Lin, Shigeyoshi Shimotsuji, Michihiko Minoh, and Toshiyuki Sakai. 1985. Efficient diagram understanding with characteristic pattern detection. *CVGIP*, 30(1).

A. Lovett and K. Forbus. 2012. Modeling multiple strategies for solving geometric analogy problems. In *CCS*.

Gordon Novak. 1995. Diagrams for solving physical problems. *Diagrammatic reasoning: Cognitive and computational perspectives*.

Lawrence O'Gorman and Rangachar Kasturi. 1995. *Document image analysis*, volume 39. Citeseer.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *EMNLP*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *TACL*, 2(Oct).

S. Roy, T. Vieira, and D. Roth. 2015. Reasoning about quantities in natural language.

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

Rohini K Srihari. 1994. Computational models for integrating linguistic and visual information: A survey. *Artificial Intelligence Review*, 8(5-6).

Adam Vogel and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *ACL*.

David J Wales and Jonathan PK Doye. 1997. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28).

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

11 Appendix: Proof of Submodularity of Equation 1

We prove that the objective function in equation (1), $\lambda\mathcal{A}(L') + \mathcal{H}(L')$ is submodular by showing that $\mathcal{A}(L')$ and $\mathcal{H}(L')$ are submodular functions.

Submodularity of $\mathcal{A}$. Consider $L' \subset L$, and a new literal to be added, $l_i \in L \setminus L'$. By the definition of $\mathcal{A}$, it is clear that $\mathcal{A}(L' \cup \{l_j\}) = \mathcal{A}(L') + \mathcal{A}(\{l_j\})$. Hence, for all $L'' \subset L' \subset L$,

$$\mathcal{A}(L'' \cup \{l_j\}) - \mathcal{A}(L'') = \mathcal{A}(L' \cup \{l_j\}) - \mathcal{A}(L')$$

. Thus $\mathcal{A}$ is submodular.

Submodularity of $\mathcal{H}$. We prove that the coverage function, $\mathcal{H}_{\text{cov}}$, and the negation of the redundancy function, $-\mathcal{H}_{\text{red}}$ are submodular independently, and thus derive that their sum is submodular. For both, consider we are given $L'' \subset L' \subset L$, and a new literal $l_j \in L \setminus L'$. Also, let K'' and K' denote the the sets of concepts covered by L'' and L', respectively, and let K_j denote the set of concepts covered by l_j.

Coverage: Since $K'' \subset K'$, $|K'' \cup K_j| - |K''| \geq |K' \cup K_j| - |K'|$, which is equivalent to

$$\begin{aligned}&\mathcal{H}_{\text{cov}}(L'' \cup \{l_j\}) - \mathcal{H}_{\text{cov}}(L'')\\ &\quad\geq \mathcal{H}_{\text{cov}}(L' \cup \{l_j\}) - \mathcal{H}_{\text{cov}}(L')\end{aligned}$$

Redundancy: Note that $\mathcal{H}_{\text{red}}(L'' \cup \{l_j\}) - \mathcal{H}_{\text{red}}(L'') = |K'' \cap K_j|$, and similarly, $\mathcal{H}_{\text{red}}(L' \cup \{l_j\}) - \mathcal{H}_{\text{red}}(L') = |K' \cap K_j|$. Since $K'' \subset K'$, thus $|K'' \cap K_j| \leq |K' \cap K_j|$. Hence,

$$\begin{aligned}&\mathcal{H}_{\text{red}}(L'' \cup \{l_j\}) - \mathcal{H}_{\text{red}}(L'')\\ &\quad\leq \mathcal{H}_{\text{red}}(L' \cup \{l_j\}) - \mathcal{H}_{\text{red}}(L'),\end{aligned}$$

By negating both sides, we derive that the negation of the redundancy function is submodular.