SemEval 2019 Task 10: Math Question Answering

Mark Hopkins
Department of Mathematics
Reed College
hopkinsm@reed.edu

Ronan Le Bras and **Cristian Petrescu-Prahova**
Allen Institute for Artificial Intelligence
{ronanlb,cristipp}@allenai.org

Gabriel Stanovsky and **Hannaneh Hajishirzi** and **Rik Koncel-Kedziorski**
Allen School of Computer Science and Engineering
University of Washington
{gabis, hannaneh,kedzior}@uw.edu

Abstract

We report on the SemEval 2019 task on math question answering. We provided a question set derived from Math SAT practice exams, including 2778 training questions and 1082 test questions. For a significant subset of these questions, we also provided SMT-LIB logical form annotations and an interpreter that could solve these logical forms. Systems were evaluated based on the percentage of correctly answered questions. The top system correctly answered 45% of the test questions, a considerable improvement over the 17% random guessing baseline.

1 Overview

Over the past four years, there has been a surge of interest in math question answering. Research groups from around the globe have published papers on the topic, including MIT (Kushman et al., 2014), University of Washington (Hosseini et al., 2014; Koncel-Kedziorski et al., 2015), National Institute of Informatics, Japan (Matsuzaki et al., 2014), University of Illinois (Roy and Roth, 2015), Microsoft Research (Shi et al., 2015; Upadhyay and Chang, 2016), Baidu (Zhou et al., 2015), Arizona State University (Mitra and Baral, 2016), KU Leuven (Dries et al., 2017), Carnegie Mellon University (Sachan et al., 2017), Tencent (Wang et al., 2017), and DeepMind (Ling et al., 2017).

Math question answering has several attractive properties that have rekindled this interest:

1. It is easy to evaluate. Usually there is a single correct answer for a given question, either numeric or multiple-choice.

2. In order to achieve robust results, systems require some (explicit or implicit) semantic representation of the question language. Consider the first question in Figure 1. The correct answer (11) has a subtle relationship to the other quantities mentioned in the text (3 and 15). There is no obvious shortcut (like word association metrics on a bag-of-words representation of the question) to guessing 11.

3. Math questions exhibit interesting semantic phenomena like cross-sentence coreference and indirect coreference (e.g. in the geometry question from Figure 1, "the third side" refers to a triangle introduced in a previous sentence).

Closed-vocabulary algebra: Suppose 3x + y = 15, where x is a positive integer. What is the difference between the largest possible value of y and the smallest possible value of x, assuming that y is also a positive integer?

Open-vocabulary algebra: At a basketball tournament involving 8 teams, each team played 4 games with each of the other teams. How many games were played at this tournament?

Geometry: The lengths of two sides of a triangle are $(x-2)$ and $(x+2)$, where $x > 2$. Which of the following ranges includes all and only the possible values of the third side y? (A) $0 < y < x$ (B) $0 < y < 2x$ (C) $4 < y < 2x$

Figure 1: Example math questions from different genres.

This task sought to unify the somewhat divergent research efforts and to address certain recognized data issues that have developed in the nascent

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 893–899
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

```
man buys an article for 10 % less than its value and sells it for 10 % more than its value . His gain or loss percent is ?
man can row upstream at 10 kmph and downstream at 20 kmph , and then find the speed of the man in still water ?
man can row upstream at 25 kmph and downstream at 35 kmph , and then find the speed of the man in still water ?
man can row upstream at 30 kmph and downstream at 40 kmph , and then find the speed of the man in still water ?
man is 15 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man is 20 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man is 21 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man is 24 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man is 28 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man is 36 years older than his son . In two years , his age will be twice the age of his son . The present age of his son is :
man walks at speed of 8 km / h 300 mtrs length train crosses man in 30 sec from back , find speed of train ?
maximun number of identical pieces ( of same size ) of a cake by making only 3 cuts ?
```

Figure 2: Sample of questions of the AQuA dataset.

	Closed Algebra	**Open Algebra**	**Geometry**	**Total**
# test questions	476	216	335	1082
# training questions	1068	353	701	2778

Table 1: Data resources for the three subtasks. Note that the fourth column ("Total") also includes a small minority of questions that do not fall into the three major categories.

phase of this subfield. We discuss these issues in the next section.

2 Existing Resources

Existing datasets are either scraped from the web and filtered (Kushman et al., 2014; Hosseini et al., 2014; Roy and Roth, 2015; Shi et al., 2015) or crowdsourced (Ling et al., 2017).

The scraped datasets have been observed to be narrow in semantic scope, and to exhibit considerable lexical overlap from question to question (Koncel-Kedziorski et al., 2015; Roy and Roth, 2016). Also, they tend to be curated to showcase proposed models (e.g. by requiring every quantity present in the semantic representation to be explicitly mentioned in the question).

DeepMind (Ling et al., 2017) provided a public, large-scale dataset called AQuA, consisting of approximately 100,000 questions. This dataset was created by using crowdsourcing to augment a nucleus of web-scraped questions. Unfortunately, the result is extremely redundant and noisy. Figure 2 shows a typical excerpt from the sorted list of AQuA questions. Note the amount of repeated boilerplate and the low quality of the language. While AQuA may prove useful for training, it is inappropriate as an evaluation set.

3 Resource: Train/Test Data from Math SAT practice tests

Over the course of the Euclid project (Hosseini et al., 2014; Seo et al., 2014, 2015; Koncel-Kedziorski et al., 2015; Hopkins et al., 2017) at the Allen Institute for Artificial Intelligence, we curated a sizable collection of practice exams for Math SAT study guides. These were originally used as training and test in a paper that appeared at EMNLP (Hopkins et al., 2017). At the time, this dataset consisted of 648 training questions (12 practice tests) and 1082 test questions (21 practice tests). For this task, we expanded the training set to include 2778 training questions (over 50 practice tests).

Because the data is a compilation of SAT practice exams in their entirety, its distribution of topics corresponds to (at least one authority's idea of) the breadth of knowledge expected of an incoming college student. It is curated by impartial third parties (the study guide publishers) and is thus not a priori biased towards any particular model. The language is high quality and diverse.

3.1 Data Collection Process and Format

First, practice exams were scanned from physical printed copies of SAT study guides by Kaplan, McGraw-Hill, and the Princeton Review, among others. We also processed official PDF practice exams issued by the College Board. Then, trained workers manually encoded each exam as a LaTeX document, attempting to preserve the original formatting as much as possible. PDFs generated from the LaTeX documents were compared against the original scans to ensure quality and corrections were made as necessary.

Finally, the LaTeX documents were automatically converted into the JSON format shown in Figure 3. Diagrams from the exams are stored as Portable Network Graphics (PNG) files in a com-

```
{
  "id": 846,
  "exam": "Kaplan Test Prep Practice Test 9",
  "sectionNumber": 2,
  "sectionLength": 20,
  "originalQuestionNumber": 18,
  "question": "In the figure above, if the slope
      of line l is \\(-\\frac{3}{2}\\),
      what is the area of triangle AOB?",
  "answer": "E",
  "choices": {
    "A": "24",
    "B": "18",
    "C": "16",
    "D": "14",
    "E": "12",
  },
  "diagramRef": "Kaplan_Test9_5.png",
  "tags": ["geometry"]
}
```


Figure 3: JSON representation of a math SAT question.

mon directory. If a question refers to a diagram, then this is captured by the "diagramRef" field. Not all questions are multiple-choice. For non-multiple choice questions, the "answer" field contains the answer itself (as a string) rather than the choice key.

3.2 Subtasks

The Math SAT contains three broadly discernible subcategories (examples of which are provided in Figure 1):

1. **Closed-vocabulary algebra (approximately 44% of the questions):** Algebra word problems described with a circumscribed mathematical vocabulary. Note that the language and semantics can still be quite involved (see the first example in Figure 1).

2. **Open-vocabulary algebra (approximately 20% of the questions):** Algebra word problems described with an open-ended vocabulary, often involving real-world situation descriptions.

3. **Geometry (approximately 31% of the questions):** In contrast to the algebra subdomains, these often involve diagrams and thus require methods that perform joint reasoning over language and images, e.g. (Seo et al., 2014).

As part of the digitization process described in the previous section, we have also tagged the questions based on this categorization. A small minority (approximately 5%) of questions do not fall into any of these categories.

This categorization provides the basis for three subtasks: (1) closed algebra QA, (2) open algebra QA, and (3) geometry QA. Note that the algebra subtasks do not involve diagrams (algebra questions involving diagrams are classified into the small "other" category). Table 1 shows the number of questions collected, organized by subtask.

4 Additional Resource: Logical Forms for Closed Algebra

To make it easier for researchers to build systems, we provided logical form annotations for a majority of the training questions in the closed algebra subtask, as well as an engine that solves these logical forms. The logical form language was introduced in (Hopkins et al., 2017). Figure 4 shows an example logical form for the question "The sum of a two-digit number and its reverse is 121. What is the number?" The logical form language adopts

```
(declare-const q Number)
(assert (= (Count (Digits q)) 2))
(assert (= 121 (Sum q (Reverse q))))
(assert (Query q))
```

Figure 4: Logical form annotation for the question "The sum of a two-digit number and its reverse is 121. What is the number?"

the popular SMT-LIB syntax (Barrett et al., 2017), to facilitate language expansion and the building of alternate engines.

As part of the SemEval task, we provided:

- Logical form annotations for 50% of the closed algebra training data.
- Thorough documentation of the logical form language.
- An interpreter that can solve all provided annotations.

The logical form interpreter is an extended version of the one described in (Hopkins et al., 2017). This interpreter is written in Scala. We provide Scala, Java, and Python APIs.

Note: the logical form annotations and interpreter were provided to help lower the barrier to entry, but participants were not required to use them.

5 Evaluation Methodology and Baseline

For each subtask, the main evaluation metric was simply question accuracy, i.e. the number of correctly answered questions. We provided a Python script that took as input a list of JSON datum { id: <id>, response: "<response>" }, where <id> is the integer index of a question and <response> is the guessed response (either a choice key or a numeric string). Its output was the number of correct responses divided by the total number of questions in the subtask.

While the main evaluation metric included no penalties for guessing, we also computed a secondary metric that implements the actual evaluation metric used to score these SATs. This metric is the number of correct questions, minus 1/4 point for each incorrect guess. We include this metric to challenge participants to investigate high-precision QA systems.

For each subtask, we provided a simple Python baseline that reads in a JSON file containing the evaluation questions, and randomly guesses a choice key for each multiple choice question, and "0" for each numeric-answer question.

To train their systems, participants were permitted to use the following public resources: (a) the provided SAT training data and annotations, (b) data collected in MAWPS (Koncel-Kedziorski et al., 2016), (c) AQuA. Participants were also welcome to use standard public corpora for training word vector representations, language models, etc.

6 Evaluation

Table 2 shows the teams (and their affiliations) that submitted systems that beat the baseline in at least one task. Tables 3, 4, 5, and 6 compares the performance of these systems. The AiFu systems (Ding et al., 2019) outperformed the other entries by a wide margin.

6.1 The AiFu System

The AiFu system (Ding et al., 2019), like other math question answering systems designed for complex domains (Shi et al., 2015; Hopkins et al., 2017), followed the architecture in Figure 5. First, a natural language question is transformed into a logical form via a "translator" (semantic parser), then this logical form is given to a symbolic solver (after a suitable format conversion).

Like the previous semantic parsing approaches, the AiFu semantic parser is engineered manually, using the training set as a guide.

AiFu also incorporates a neural network-based system trained to guess the answer to a multiple choice question based on its question word sequence. Although this system does not work well independently, it boosts the performance of the overall system when used as a fallback for questions that go unanswered by the symbolic system.

The AiFu team submitted two variants of their system, referred to in the comparison tables as AiFu 1 and AiFu 2.

6.2 The ProblemSolver System

The ProblemSolver system (Luo et al., 2019) combines two approaches.

The first approach is a neural sequence-to-sequence translator that maps a question, e.g. "If $x + 345 = 111$, what is the value of x", to a response, e.g. "-234". Because the provided data is insufficiently large for training the sequence-

Team Name	Affiliation	Citation
AiFu	iFLYTEK Research, Shanghai Research Center for Brain Science and Brain-Inspired Intelligence, Fudan University, Massey University, University of Science and Technology of China	(Ding et al., 2019)
ProblemSolver	University of Tuebingen, Germany	(Luo et al., 2019)
FAST	National University of Computer and Emerging Sciences, Pakistan	

Table 2: Teams whose entries exceeded baseline performance on at least one subtask. FAST (and the two anonymous systems) did not provide system description papers.

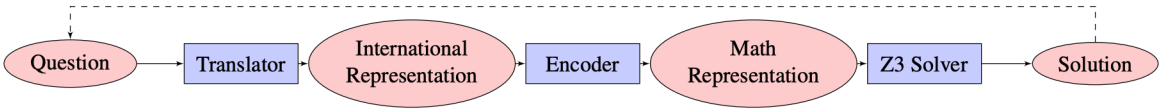

Figure 5: The architecture of AiFu. Figure taken from (Ding et al., 2019).

Team	Accuracy	Penalized
AiFu 1	.454 (1)	.368 (1)
AiFu 2	.376 (2)	.280 (2)
Anonymous 1	.208 (3)	.089 (3)
Anonymous 2	.196 (4)	.074 (4)
FAST	.173 (5)	.007 (6)
baseline	*.170* (6)	*.043* (5)
ProblemSolver	.149 (7)	-.021 (7)

Table 3: Results on the overall task (only showing entries that exceeded baseline performance on at least one subtask). System rank on each metric is shown in parentheticals.

Team	Accuracy	Penalized
AiFu 1	.706 (1)	.658 (1)
AiFu 2	.632 (2)	.576 (2)
Anonymous 2	.196 (3)	.075 (3)
Anonymous 1	.187 (4)	.064 (4)
FAST	.183 (5)	.020 (5)
ProblemSolver	.157 (6)	-.012 (7)
baseline	*.146* (7)	*.015* (6)

Table 4: Results on the closed-vocabulary algebra subtask (only showing entries that exceeded baseline performance on at least one subtask). System rank on each metric is shown in parentheticals.

to-sequence model, they use data augmentation methods to increase the data size to over 600K questions.

The second approach is an adaptation of the arithmetic tree approach of (Roy and Roth, 2015)

Team	Accuracy	Penalized
AiFu 1	.251 (1)	.145 (1)
AiFu 2	.247 (2)	.140 (2)
Anonymous 2	.247 (2)	.140 (2)
Anonymous 1	.196 (4)	.079 (4)
baseline	*.174* (5)	*.052* (5)
FAST	.146 (6)	-.025 (6)
ProblemSolver	.146 (6)	-.025 (6)

Table 5: Results on the open-vocabulary algebra subtask (only showing entries that exceeded baseline performance on at least one subtask). System rank on each metric is shown in parentheticals.

Team	Accuracy	Penalized
AiFu 1	.265 (1)	.145 (1)
Anonymous 1	.216 (2)	.099 (2)
baseline	*.212* (3)	*.095* (3)
FAST	.159 (4)	-.009 (6)
Anonymous 2	.152 (5)	.023 (4)
ProblemSolver	.152 (6)	-.018 (7)
AiFu 2	.134 (7)	-.003 (5)

Table 6: Results on the geometry subtask (only showing entries that exceeded baseline performance on at least one subtask). System rank on each metric is shown in parentheticals.

to Math SAT question answering.

The combination of these techniques provides a minor improvement over the random guessing baseline.

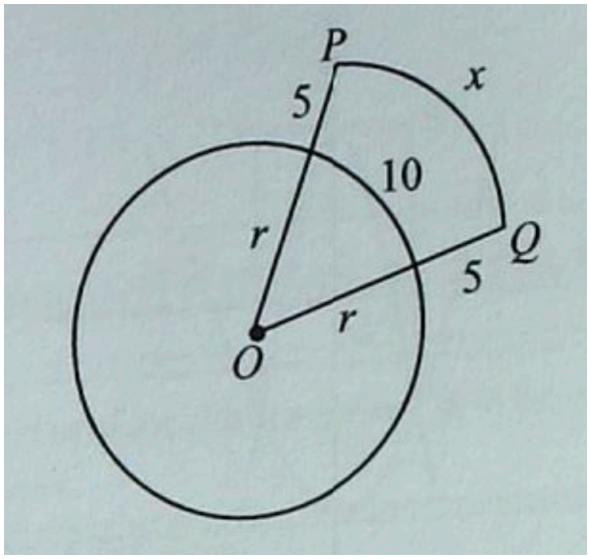

Figure 6: Example diagram accompanying the question "In the circle above, the length of an arc is 10, and there are two radii shown extended by 5 units outside the circle. If the length of arc Q is x, what must x equal?"

7 Discussion

Lately, math question answering seems to have bifurcated into two distinct approaches:

1. Simple, elegant machine learning approaches that work mainly on narrowly-scoped datasets with considerable redundancy.

2. Engineering-heavy, rule-based approaches that significantly outperform ML approaches on more realistic datasets, but are laborious to scale to new domains.

This SemEval task provides additional examples of these two approaches. As NLP researchers, we could focus on narrow-scope datasets for the time being[1], improving the performance of scalable ML approaches on these datasets. However the challenges presented by more difficult datasets, like the Math SAT data provided in this task, are intriguing and important. For instance:

How do we synthesize information that comes from heterogenous sources (e.g. from text and diagrams)?

Many of the geometry questions require a generalized notion of coreference resolution that spans language and vision. Figure 6 shows an example diagram that accompanies the question "In the circle above, the length of an arc is 10, and there are two radii shown extended by 5 units outside the circle. If the length of arc Q is x, what must x equal?". It remains an open question how to reliably resolve textual references like "the circle above" and "two radii shown" with diagram components. (Seo et al., 2014, 2015) provide a starting point for this area, but their dataset consisted of less than 100 diagrams. Hopefully our larger resource can help spur research into this research question.

How can machine learning be leveraged to reduce (or eliminate) the burden of engineering semantic parsers for complicated domains?

Given that the only techniques that have so far found success on Math SAT question answering (Hopkins et al., 2017; Ding et al., 2019) have involved semantic parsers with engineered rules, it suggests that one path forward might be to use machine learning to facilitate the engineering or elicitation of such rules for low-resource QA domains.

How do we create ML systems for diverse datasets for which we do not (and will never have) millions of training instances?

Despite the huge industry surrounding the Math SAT, it was still challenging to find and digitize over 50 distinct practice exams. Having millions of instances is not feasible. We argue that there will always be a long tail of domains for which we do not have millions of training instances, and we need ways to induce performant systems on this scale of data.

8 Conclusion

We have digitized over 70 SAT practice exams as a resource for driving research in math (and low-resource) question answering. We have also provided logical form annotations for approximately half of the closed-vocabulary algebra questions in the training data. The top system in our competition, AiFu, correctly answered 45% of the test questions, compared to a random guessing baseline of 17%. Our data and logical forms are available at https://github.com/allenai/semeval-2019-task-10, subject to the terms and conditions specified in that repository.

[1]Indeed, this seems to be the prevailing opinion of the hundredsome task participants who abandoned the task after obtaining the data.

References

Clark Barrett, Pascal Fontaine, and Cesare Tinelli. 2017. The SMT-LIB Standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa. Available at www.SMT-LIB.org.

Keyu Ding, Yifan Liu, Yi Zhou, Binbin Deng, Chaoyang Peng, Dinglong Xue, Qinzhuo Wu, Qi Zhang, and Enhong Chen. 2019. Aifu at semeval-2019 task 10: A symbolic and sub-symbolic integrated system for sat math question answering. In *SemEval-2019 Task 10*.

Anton Dries, Angelika Kimmig, Jesse Davis, Vaishak Belle, and Luc De Raedt. 2017. Solving probability problems in natural language. In *International Joint Conference on Artificial Intelligence*.

Mark Hopkins, Cristian Petrescu-Prahova, Roie Levin, Ronan Le Bras, Alvaro Herrasti, and Vidur Joshi. 2017. Beyond sentential semantic parsing: Tackling the math sat with a cascade of tree transducers. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 795–804.

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

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *TACL*, 3:585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157.

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

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Xuefeng Luo, Alina Baranova, and Jonas Biegert. 2019. Problemsolver at semeval-2019 task 10: Sequence-to-sequence learning and expression trees. In *SemEval-2019 Task 10*.

Takuya Matsuzaki, Hidenao Iwane, Hirokazu Anai, and Noriko H Arai. 2014. The most uncreative examinee: A first step toward wide coverage natural language math problem solving. In *AAAI*, pages 1098–1104.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *ACL*.

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

Subhro Roy and Dan Roth. 2016. Unit dependency graph and its application to arithmetic word problem solving. *arXiv preprint arXiv:1612.00969*.

Mrinmaya Sachan, Avinava Dubey, and Eric P. Xing. 2017. From textbooks to knowledge: A case study in harvesting axiomatic knowledge from textbooks to solve geometry problems. In *EMNLP*.

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

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

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*.

Shyam Upadhyay and Ming-Wei Chang. 2016. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. *CoRR*, abs/1609.07197.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *EMNLP*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *EMNLP*.