Reasoning with Sets to Solve Simple Word Problems Automatically

Sowmya S Sundaram
Indian Institute of Technology, Madras
Chennai 600036
sowmya@cse.iitm.ac.in

Deepak Khemani
Indian Institute of Technology, Mandi
Himachal Pradesh 175005
khemani@iitmandi.ac.in

Abstract

A system, Magi, is proposed, which analyses simple addition/subtraction arithmetic word problems expressed in English, represents them in the form of schemas and sets, reasons with set cardinalities and presents the final answer in English phrases. It also provides simple explanations. This work presents a study of the features of a knowledge-based system used for solving such a task. It has been evaluated and has been found to perform better than current knowledge-based systems for similar problems.

1 Introduction

Natural language understanding is one of the key elements of human intelligence. Hence, it has attracted the attention of a sizeable population of researchers of artificial intelligence. The first published work in this field (Bobrow, 1964) attempted to solve word problems presented to a computer in English. The appeal of solving word problems lies in the fact that semantic understanding is required to map the word problem to a mathematical framework.

Consider the following example.
Input: Keith has 20 books . Jason has 21 books . How many books do they have together?
Output: Altogether 41 books

Here, the system has to map the word 'they' to 'Keith' and 'Jason'. This is an example of co-reference resolution. Next, the notion of associating '20 books' to 'Keith' and '21 books' to 'Jason' has to be captured. These relevant details are also extracted. The last piece of information required is the word 'together' that signifies what is the goal of the problem. In this work, these details are extracted by using the Stanford Core NLP (Manning et al., 2014) suite of tools extensively. Other approaches include semantic parsing (Shi et al., 2015), learning equation co-efficients (Kushman et al., 2014), learning expression trees (Koncel-Kedziorski et al., 2015) and so on. In this work, as the principle was to build as precise a system as possible, we've used a rule based approach. The motivation was that if this tool was used by a student, she should be able to see the trace of the solution.

In order to know what are the elements that are to be extracted from the word problem, some model of word problems must be encoded into the system. This is the role of knowledge representation. Here, knowledge representation is in the form of schemas that are templates for solving problems. They describe common categories of word problems. The schema for the above problem is 'combine' which describes that the answer is the sum of entities in question. Internally, this idea is represented as sets for closer coupling to the semantics of the problem.

t_0
Jason has B books
Keith has A books
———————
A 20
B 21

The next step is reasoning. The schema 'combine' directs that the sum of the 'books' owned by 'Jason' and 'Keith' is required. The answer is computed by adding the cardinalities of A and B. The final answer, '41' is presented as 'Altogether 41 books'. The last step of generating the answer is facilitated by the schema as well.

The challenges in this problem solving process are high. This is because natural language processing is difficult and often ambiguous or

S Bandyopadhyay, D S Sharma and R Sangal. Proc. of the 14th Intl. Conference on Natural Language Processing, pages 188–196,
Kolkata, India. December 2017.

may rely on implicit details. Magi resolves some of the ambiguities by reasoning about implicit events and making some assumptions. There are some ambiguities in schema identification as well which have been partially addressed using some heuristics. While computing elementary problems, numerical efficiency of computers is much more. The challenging task is the introduction of language and representing the information extracted from the natural language.

2 Related Work

As mentioned, the work that pioneered the field of natural language understanding was (Bobrow, 1964). The program, STUDENT, was able to process sentences that followed a specified template and could handle addition, subtraction, division, multiplication and exponentiation.

Schemas were introduced by (Fletcher, 1985) based on cognitive theory. It specified three schemas - combine, compare and change. After this, (Dellarosa, 1986) proposed ARITHPRO which encoded some inheritance. For example, dolls and balls are toys. If a problem described dolls, balls and clothes and the task was to find total toys, ARITHPRO would pick only relevant entities.

Schemas were used relatively recently in (Bakman, 2007). It could solve multi-step problems and could ignore extraneous information. However, the system did not scale well as the complexity of natural language increased. This is a common trait running through all these knowledge-based systems. A major stumbling block was the complexity of natural language understanding. Many systems worked on a subset of natural language called Controlled Natural Language to resolve ambiguities.

After this system, as mentioned in (Mukherjee and Garain, 2008), without a common standard dataset to compare different algorithms, the interest in this field died down. Also, the extensive human effort involved in curating rules for these systems was not encouraging.

In recent times, there has been a resurgence of interest for this type of problems. In (Kushman et al., 2014), word problems were solved by building an empirical model that matched the numerals in a word problem to co-efficients in a template. Their domain was the set of word problems that could be solved by a set of linear equations. They achieved a commendable accuracy. (Zhou et al., 2015) improved this work by using quadratic programming on a simpler and more efficient model. Another work (Hosseini et al., 2014) used a state representation for arithmetic word problems. It used machine learning to identify the characteristic operation signalled by a verb. They provided three datasets of varying difficulty that have been used for evaluation in this paper against their fully knowledge-based variant of the code. (Shi et al., 2015) solves algebraic word problems which reason about numbers and their relations. It uses semantic parsing with a custom-built language for their chosen domain. The work presented in (Koncel-Kedziorski et al., 2015) learns a model that maps natural language to expression trees. They could solve single-variable word problems effectively. As the narrative of the problem became longer, the search space grew exponentially. A knowledge based system could potentially solve problems of arbitrary length provided the sentences could be processed by it. In (Mitra and Baral, 2016), there is a notion of the categories of word problems where a model learns to identify which category and the alignment of the numbers to the template of the word problem. It is the learning version of this work. The drawback it faces is the heavy annotation required for learning such an alignment. A recent work, (Ling et al., 2017) uses deep learning to solve general word problems and provide explanations for the same. It solves problems which are in a competitive exam style, with possible answer options. It develops a language model and a mathematical model simultaneously. Since the problem setting is slightly different, its performance on existing datasets is not available.

The knowledge based systems are precise but can attempt few real-world problems as their natural language processing is limited. On the other hand, empirical systems can tackle a wide gamut of problems but they are not as precise as knowledge based systems. This work hopes to maximise the trade-off between the two methodologies by using statistically trained parsers and a well-defined representation system.

3 The Process

As the dependency parser provided by (Manning et al., 2014) is not robust for long sentences, the given English word problem is first simplified and

then passed to the co-reference resolver and the dependency parser. The simplification is based on a set of rules derived from the part-of-speech tags and the constituency parser. The parsers extract relevant information for each sentence. The sentences are then ordered in increasing order of time by using the tense of the sentences. After this, sets are created for each numerical entity. Then, relationships between these sets are established by schemas using the extracted information. Finally, after reasoning with the set cardinalities, the answer is displayed along with the trace and explanation. This process is explained in detail with a running example

Algorithm 1: The Problem Solving Process

```
Input: Word Problem: p
Output: String: expl, Number: ans
WordProblem p1 = simplify(p)
List<Steps> extractedInfo =
  extract(p1.sentences())
List<Steps> orderedInfo =
  rearrange(extractedInfo)
time = 0
questionSet = ∅
story = ∅
for each step in orderedInfo do
    if event(step) then
        time = time + 1
    end
    story = story.add(step,time)
    story = story.apply(step.schema)
    if step.isQuestion = true then
        questionSet =
          story.get(step,time).value
    end
end
solve(story.sets)
expln = explain(story)
ans = questionSet.cardinality
Print ans
Print expln
```

4 Natural Language Understanding

The steps involved in natural language understanding are explained briefly below.

- Resolve unknown entities - For example, if the problem had sentences like, 'There are 5 trees in a park. Park workers cut 2 of them', this is converted to 'There are 5 trees in a park. Park workers cut 2 trees.' This is to ease the task of the parsers.

- Simplifying sentences - Most long sentences are split into simpler sentences. For instance, 'Sally got 4 erasers and 3 pencils' is split as 'Sally got 4 erasers. Sally got 3 pencils.'

- Resolving co-references - This has been done by using the 'decoref' annotator provided by (Manning et al., 2014).

- Rule based information extraction - a set of rules that work on the output given by the dependency parser to extract the details of each sentence .

- The retrieved information is then ordered based on the tense of each sentence.

4.1 An Example

Let us see an example to illustrate the various points described above. Consider the problem, 'Molly owns the Wafting Pie Company. This morning, her employees used 816 eggs to bake pumpkin pies. If her employees used a total of 1339 eggs today, how many eggs did they use in the afternoon?'.

4.1.1 Preprocessing

For this problem, the first step is to resolve the pronoun 'her'. After this step, our system changes the input to, 'Molly owns the Wafting Pie Company. This morning, *Molly's* employees used 816 eggs to bake pumpkin pies. If *Molly's* employees used a total of 1339 eggs today, how many eggs did *Molly's employees* use in the afternoon?'.

4.1.2 Simplification

As the sentences are relatively complex, simplifying them brings much better results. At the end of simplification, the problem is changed to : 'Molly owns the Wafting Pie Company. Molly's employees used 816 eggs to bake pumpkin pies. Molly's employees used a total of 1339 eggs today. How many eggs did Molly's employees use in the afternoon?'. This is achieved by examining the Part-Of-Speech(POS) tag of every sentence, identifying the verb, and extracting the relevant noun phrase and verb phrase.

4.1.3 Information Extraction

Some rules are used to get the information required for representation. Each sentence's analysis is enlisted below.

- 'Molly owns the Wafting Pie Company' - this sentence is ignored because there is no number involved. There are some exceptions to this rule. If the sentence contains words like 'some', that information is encoded.

- 'Molly's employees used 816 eggs to bake pumpkin pies' - this sentence is converted by Magi as

owner1	:	Molly's employees
owner2	:	(none)
verb	:	use
entity	:	egg
value	:	816
keyword	:	use
procedure	:	reduction
tense	:	past
isQuestion	:	false
isAggregator	:	false

  The 'owner1' and 'owner2' fields suggest who are the participants. Here, only 'Molly's employees' are relevant. If the question was 'Sally gave 4 kites to Sam', then the two owners would be 'Sally' and 'Sam' respectively. This is extracted by a set of rules. In this case, the subject of sentence(denoted by the 'nsubj' tag) is taken as the first owner. There are a set of keywords and their associated procedures stored - this is explained in more detail in the next section. If the sentence contains one of the keywords, it is retrieved from the sentence and stored along with the corresponding procedure's name. The tense is stored by analysing the POS tag. It is later used for ordering. The field 'isQuestion' signifies whether this step contains information that pertains to the answer to be retrieved. On the other hand, 'isAggregator' states whether the sentence contains any word that imply combination, such as 'total', 'altogether', etc.

- 'Molly's employees used a total of 1339 eggs today.' - a similar process leads to the following data to be stored.

owner1	:	Molly's employees
owner2	:	(none)
verb	:	use
entity	:	egg
value	:	1339
keyword	:	use
procedure	:	reduction
tense	:	past
isQuestion	:	false
isAggregator	:	true

- 'How many eggs did Molly's employees use in the afternoon?' -

owner1	:	Molly's employees
owner2	:	(none)
verb	:	use
entity	:	egg
value	:	(empty)
keyword	:	use
procedure	:	reduction
tense	:	past
isQuestion	:	true
isAggregator	:	false

  By setting the 'isQuestion' flag, the system is now equipped with the insight that the answer required is the number of eggs Molly's employees used.

5 Knowledge Representation

5.1 Schemas

Schemas are templates that suggest how a problem should be solved. They were applied to solve math word problems first by (Fletcher, 1985). He used three schemas - Combine, Change and Compare. Let us consider the 'Compare' schema. A typical example is 'Rachel has 3 pencils. Tom has 3 pencils more than Rachel. How many pencils does Tom have?'. The schema, 'Compare', and its instantiation is given below:
(owner1) has (X) (object)
(owner2) has (Y) (object) more than (owner1)
(owner2) has (Z) (object)
Z = X + Y
(owner1) = 'Rachel', (owner2) = 'Tom', (object) = 'pencil', (X) = 3, (Y) = 3.

Schemas have a structure that can be mapped to the sentences given in the sentence along with an

equation connecting the variables. If two variables are retrieved from the problem (in this example, X and Y), the value of the third variable can be computed.

The first disadvantage of this method is that it is too rigid. All word problems are not expressed in a format that is easy to map to this format. If the question was changed as 'Rachel has 3 pencils. Tom has 3 more. How many does he have?', this particular schema would fail as it does not exactly match the natural language input expected. The second issue is that these three schemas are inadequate to describe all types of problems. For example, the problem 'Samantha has 8 cookies. She ate 3 of them. How many does she have now?', would not fit in any of the above schemas. The 'change' schema is tailored to capture transfer of 'object' from one person to another. Hence, though it seems applicable, it is not so.

To counter these challenges, (Bakman, 2007) describes his system 'ROBUST' that can handle a larger number of schemas. Some ideas were inspired by Script Applier Mechanism (SAM) by (Schank and Abelson, 1975) which captured semantics by grouping words from a dictionary into categories. Similarly, instead of a single keyword for schemas, ROBUST mapped a lot of keywords to a single schema. For example, 'eat', 'destroy', 'kill' were keywords for the 'termination' schema. ROBUST concentrated on the various possibilities of 'change' schema. It also used schemas iteratively until all possible equations were applied in order to handle multi-step problems. It showed significant improvements over existing systems.

5.2 Schemas and Time

From ROBUST's emphasis on the 'change' schema, the next natural step is to capture information about time. By explicitly assigning timestamps to sentences, the search for schema instantiation is made more focussed.

5.3 Schemas and Ambiguity

A classic example of ambiguity can be seen in the problem 'Samantha ate 8 cookies. Anne ate 4 cookies more than Samantha. How many cookies did Anne eat?'. Here, the correct schema to be used is 'Compare'. However, due to the word 'ate', the 'termination' schema is also applicable. If the 'termination' schema is applied, since there is no information about the cookies any of them had before or after, it cannot be instantiated. To address this, the schemas are modified such that the verb is variable and it can reason about any verb. The narrative is represented in the following manner.

t_0
Samantha : eat : 8 : cookies
t_1
Anne : eat : 8 + 4 : cookies

Hence, the template-matching is relaxed and more problems can be solved.

5.4 Schemas and Sets

After introducing time, its related concepts and reasoning about events while applying schemas, there are still some problems which cannot be addressed. Consider, 'There are 70 students in a class. If 65 students are present, how many are absent?'.

These problems have no events, or tell-tale keywords for helping the system solve problems. The schema of combination usually implies an aggregation over different owners. This a case of set completion, where the 65 students are a subset of the 70 students in class and the set of students who are present is disjoint from the set of absentees. Hence, the representation is shifted to schemas with descriptions as sets.

Revisiting 'Rachel has 3 pencils. Tom has 3 pencils more than Rachel. How many pencils does Tom have?', the 'compare' schema which has been specialised as 'compare-plus' in Magi is stored as :
(owner1) (verb) (X) (object)
(owner2) (verb) (Y) (object) more than (owner1)
(owner2) (verb) (Z) (object)
$|Z| = |X \cup Y|, X \cap Y = \emptyset$
(owner1) = 'Rachel', (owner2) = 'Tom', (verb) = 'has', (object) = pencil, $|X| = 3, |Y| = 3$.

Coming back to the set-completion scenario, the narrative is represented as :
t_0
class : has : A : students
class : has : B : present students

A 70
B 65

While parsing the sentence, the behaviour of

antonyms is recorded and the case for subset completion is set to be true, if the antonyms are appropriately situated. If it is true, the statements $B \subseteq A, C \subseteq A, B \cap C = \emptyset$ are added along with 'class : has : C : absent students'. Antonyms have been computed from `https://www.thesarus.com`.

5.5 Magi's Schemas

The schemas used by Magi are described in Table 1. The procedures are programming directives and are more flexible than traditional schemas. Implicitly, all sets are considered disjoint unless set completion is involved. Even though the description says 'owner1' and 'owner2', Magi is implemented in such a way that it can reason about different entities owned by the same owner if required.

Schema	Procedure	Relations
combine	Sum over all relevant entities	$\|D\| = \|A \cup B..\|$
comparePlus	owner1 has A items, owner2 has B items more, owner2 has C items	$\|C\| = \|A \cup B\|$
compareMinus	owner1 has A items, owner2 has B items less, owner 2 has C items	$\|C\| = \|A - B\|$
increase	owner1 had A items, owner1 got B items more, owner1 has C items now	$\|C\| = \|A \cup B\|$
reduction	owner1 had A items, owner1 lost B items, owner1 has C items now	$\|C\| = \|A - B\|$
set-completion	A,B,C	$B \subseteq A, C \subseteq A$

Table 1: Flexible Schemas used by Magi

6 Reasoning

In the straight-forward situation, reasoning is simply a case of solving the equations relating set cardinalities based on the axioms of set theory. However, to address a larger type of problems, some common sense rules have been added.

6.1 Handling Implicit Events

Consider the problem, 'Last week Tom had $74. He washed cars over the weekend and now has $86. How much money did he make washing cars?'. The word 'wash' is not a keyword, hence it is not registered as an event. When the narrative is being constructed, the first statement will record that Tom has 74 dollars. As no event has occurred, the timer is not incremented. After that, the system encounters that Tom has 86 dollars at the same time. Since this is not possible, it introduces an event, 'Tom gets 86 - 74 dollars'. This is illustrated below:

t_0
Tom : has : A : dollars
t_1
Tom : get : C : dollars
t_2
Tom : has : B : dollars

A 74
B 86
C 12

The statements involved are $|C| = |B| - |A|$.

6.2 Assumption of Initial Conditions

Most schema-based systems fail due to some missing information. For example, ROBUST would fail to solve 'Jane bought 10 cookies. She ate 3 cookies. How many does she have now?'. This is because, it would try to find some value as the initial number of cookies Jane owned. Magi sets initial values as $\emptyset$.

6.3 Reasoning about Events

Ideas from 'Event Calculus' described in (Shanahan, 1999) have been used to construct the narrative. For example, circumscription is used in the problem, 'Sam grew 4 watermelons, but the rabbits ate 3 watermelons. How many watermelons does Sam have?' to reason that the 3 watermelons are actually a subset of Sam's watermelons. This idea was also employed by (Hosseini et al., 2014). Also, common sense law of inertia was implemented to state that entities that were not affected by an event, continue to persist across time steps.

6.4 Reasoning and Natural Language

Sometimes, reasoning is performed using the extracted information presented for representation. For example, the situation where 'Sam buy games for $35' actually implies that the event is 'Sam spent 35 dollars'. Such rules are also enforced.

6.5 Heuristics

Due to the complexity of processing natural language and the limited rules available, the numerals in the problem may not be correctly assigned to the templates required for a schema properly. Hence, some heuristics are used to improve performance. As expected, they are not sound. One consistent heuristic is, if Magi

retrieves a value that is already given in the question, then search is repeated. Another heuristic is, if the system is unable to represent as desired, but has recognized that the question needs aggregation, it simply returns the sum of all entities.

7 Natural Language Generation

Since one of the driving factors behind this work is to facilitate the understanding of students, an attempt has been made to explain the answer obtained in natural language. The trace of the problem is recorded as the problem is solved and then an explanation is generated. The quality of generation is quite low at this point in time but the intermediate representation is provided. This can be taken up by a generation expert and designed.

One of the successful examples is illustrated below.

7.1 Problem

John had 7 apples. Mary has gave some apples to John. Now, John has 10 apples. How many apples does Mary have?

7.2 Explanation

John has 7 apples.
Mary gives x apples.
Hence, John has 7+x apples.
Now, John has 10 apples.
Therefore, 7+x = 10
Mary gives 3 apples

7.3 Trace

John had 7 apples. Mary gave 3 apples to John. John had 10 apples. John had 10 apples. Mary had 3 apples.

This trace is concatenating the situation at every time step. Hence, the statement John had 10 apples is repeated twice. We used SimpleNLG (Gatt and Reiter, 2009) for generation.

8 Evaluation

Magi has been coded in Java 1.7 and has used the same version of (Manning et al., 2014) parser as the one used by ARIS (Hosseini et al., 2014) for a fair evaluation. The work has been compared against other knowledge based systems.

Magi has been evaluated on the three datasets, DS1, DS2 and DS3 provided by (Hosseini et al., 2014). DS1 has 134 problems. DS1 and DS3 are similar in terms of the applicable schemas. However, DS3 has more complex sentences and has extraneous information. It has 121 problems. DS2 involves the use of decimals which is difficult for parsing. Also, DS2 has a lot of problems that require set-completion, an issue whose solution was the inspiration for this representation. It has 140 problems.

	DS1	DS2	DS3	Avg
Magi	**95.52**	**80.00**	**84.30**	**86.51**
Gold ARIS	94.0	77.1	81.0	84.0
ROBUST	12.69	0.71	0	4.56
WolframAlpha	5.97	2.14	0.83	3.03

Table 2: Performance

A comparison has been presented in Table 2 with respect to three other systems. One is ROBUST (Bakman, 2007) which has been discussed before. (Hosseini et al., 2014) presented ARIS. It attempted to learn the equation categorising verbs. They also presented an algorithm for learning that information. However, by limiting themselves to verbs (change schemas), other schemas such as 'combine' and 'compare' were missed. As we have not performed any empirical method to learn the keyword-schema mapping, the system for comparison is Gold-ARIS. (Wolfram, 2015) is another system that solves math word problems on the Internet without divulging implementation details.

The increased performance over Gold ARIS is because of the use of heuristics, addressing set completion and handling implicit events. Also, simplifying the problem and performing some reasoning with the language helped reduce parser errors mentioned in (Hosseini et al., 2014). ROBUST performs relatively better with DS1 because it consists of simple sentences. As the complexity of processing English increases, the performance of both ROBUST and WolframAlpha reduces.

8.1 Analysis of Errors

8.1.1 Absence of a Keyword

Consider "A restaurant served 9 pizzas during lunch and 6 during dinner today. How many pizzas were served today?". Since there are no keywords like "altogether", the system did not recognize that the "combine" schema is to be applied. Also, it could not identify that lunch and dinner are parts

of "today".

8.1.2 No Model for Intent

In "Joan decided to sell all of her old books. She gathered up 33 books to sell. She sold 26 books in a yard sale. How many books does Joan now have?", the system couldn't represent that Joan hadn't actually sold 33 books and was only intending to sell them.

8.1.3 Issues in Extracting Entities

Consider "A ship full of grain crashes into a coral reef. By the time the ship is fixed, 49952 tons of grain have spilled into the water. Only 918 tons of grain remain onboard. How many tons of grain did the ship originally contain?". The system did not recognize that the ship had spilled tons of grain. The system represented it as "water has 49952 ton" and "water spill 918 ton". Here there are multiple entities that are interacting with each other. These facets could not be extracted by the rules designed for information extraction.

9 Discussion

While it has been presented that Magi is a good knowledge-based system, the question remains whether it is robust enough to have a recall comparable with empirical systems. This is hard to evaluate as empirical systems are usually tested by cross validation. When a human expert makes rules, she cannot subjectively claim that the rules have made solely on the basis of a section of the data. The fact that the system can achieve a high accuracy shows that it does solve a large number of problems. However, there may be a problem of over-fitting in some sense. In empirical systems, this is also possible because often there is a considerable overlap of sentence styles in the training and test examples. How these systems fare with completely unseen data would be an interesting experiment to compare these algorithms. This work is not limited to presenting a numerical answer. Rather, it attempts to illustrate what are the components required to build a product that would benefit students - namely natural language understanding, representation, reasoning and natural language generation. A loss in precision implies that it might induce confusion in a student's mind. In hindsight, the heuristics did drastically reduce precision and doing away with them is part of the future work.

9.1 Knowledge Acquisition Bottleneck

The primary reason knowledge-based systems went out of vogue for natural language processing is because of the knowledge acquisition bottleneck. In this work, the types of word problems were already established in the literature. However, two sources of knowledge acquisition bottleneck still exist - the mapping between the schemas and the keywords as well as the various rules and strategies to extract information and represent them as schemas. While a human expert can sift through the data and come up with better rules than a learning program in a simplistic domain such as this, the generalisability of this approach is questionable.

9.2 Similarities between Knowledge-Based and Empirical Systems

In this particular domain, there is often a need to encode world knowledge in some form. In empirical systems, it comes as the cost of annotation and choice of hand-coded features.

9.3 The Need for Semantics

Many works (eg. (Shi et al., 2015)) recognize the need for semantics for this class of problems. A single word could completely change the equation construction. Hence, it is imperative that there must be some model of mathematical computation.

10 Conclusion and Future Work

We have presented a knowledge-based system to explore what are the exact sources of information required in the quest for building a student-friendly application that is precise. It has been shown that to solve such problems, world knowledge has to be encoded and semantic understanding is required. Exciting developments such as deep learning (Ling et al., 2017) in natural language processing can learn the required representation as well and succeed in building an end-to-end system. However, it comes at the cost of a huge amount of data that is not available at this point in time for many mathematical problem domains. Though we have introduced some level of statistical analysis through parsers, it would be beneficial to explore semantic parsing and other approaches to map the natural language description to an underlying representation with higher precision for semantically richer domains.

References

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*.

Daniel G Bobrow. 1964. A question-answering system for high school algebra word problems. In *Proceedings of the October 27-29, 1964, fall joint computer conference, part I*, pages 591–614. ACM.

Denise Dellarosa. 1986. A computer simulation of childrens arithmetic word-problem solving. *Behavior Research Methods, Instruments, & Computers*, 18(2):147–154.

Charles R Fletcher. 1985. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5):565–571.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93. Association for Computational Linguistics.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. *ACL (1)*, pages 271–281.

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*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

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

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2):93–122.

Roger C Schank and Robert P Abelson. 1975. *Scripts, plans, and knowledge*. Yale University.

Murray Shanahan. 1999. The event calculus explained. In *Artificial intelligence today*, pages 409–430. Springer.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*.

Stephen Wolfram. 2015. Wolfram—alpha. *On the WWW. URL http://www. wolframalpha. com*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *EMNLP*, pages 817–822.