Natural Language Processing for Solving Simple Word Problems

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

Deepak Khemani
Indian Institute of Technology, Madras
Chennai 600036
khemani@iitm.ac.in

Abstract

This paper describes our system which solves simple arithmetic word problems. The system takes a word problem described in natural language, extracts information required for representation, orders the facts presented, applies procedures and derives the answer. It then displays this answer in natural language. The emphasis of this paper is on the natural language processing(NLP) techniques used to retrieve the relevant information from the English word problem. The system shows improvements over existing systems.

1 Introduction

The aim of this work is to solve a mathematical problem involving addition/subtraction which is given in English. It uses the principles of NLP to extract information from the text. Then, it uses some pre-stored routines to calculate the answer. To illustrate, a sample input and the corresponding output is shown below.

Input: John has 3 apples. He gave 1 apple to Mary. How many apples does John have now?

Output: John has 2 apples.

The system extracts the information that, to use the parlance of (Bakman, 2007), an owner "John" has 3 apples. The natural language processor maps the word "give" to the knowledge that it must happen between two owners, where the first owner loses something and the second owner gains it. This information is coded in the form of schemas which are templates describing different scenarios. For example, the above scenario is known as a "Transfer-In-Ownership" or "Change-Out" which says John loses one apple and Mary gains it.

The issues in this simple example are many. The pronoun, "He" must be mapped to "John". The order of the sentences matter. If the problem was "John gave 1 apple to Mary. He has 3 apples now. How many apples did John have?", then the answer is completely different.

To elaborate, the problem solving process begins by processing the question that is expressed in natural language. After processing the question, the information required by the knowledge representation is extracted. This is a knowledge-based natural language processing system. Hence, the domain knowledge provided by the underlying representation can also help clear ambiguities faced by the natural language processor.

For example, one of the heuristics used is described as follows. If the problem is, "There are 2 pencils in the drawer. Tim placed 3 pencils in the drawer. How many pencils are now there in total?", the natural language processor does not understand whether the question is about the pencils with Tim or the ones in the drawer. However, after representing the knowledge, it is clear that the "drawer" has a change in state and more probably, the question is about the ones in the drawer.

The given word problem is split into constituent sentences. Every sentence is analysed and the information required by the representation is extracted using Stanford Core NLP Suite (Manning et al., 2014) in a manner that is described in detail in the following sections. The answer is again displayed in natural language.

For evaluation, the datasets provided by (Hosseini et al., 2014) are used. The proposed system shows an improvement because of the exploitation of knowledge representation and using keywords other than verbs.

2 Related Work

Such a problem was first attempted by Bobrow as part of his PhD dissertation in 1964. The

D S Sharma, R Sangal and E Sherly. Proc. of the 12th Intl. Conference on Natural Language Processing, pages 394–402,
Trivandrum, India. December 2015.

system described in (Bobrow, 1964) could solve an algebra problem given in a subset of natural language. A common trait of most systems that use this approach is to work with a subset of the natural language, most commonly via a Controlled Natural Language. This work attempts to extract information from the ambiguous English sentences themselves. The latest in this line of work for mathematical algebraic word problems, without using empirical methods, is in (Bakman, 2007) which proposed a method that improved upon (Dellarosa, 1986). Dellarosa used "schemas" to solve addition/subtraction problems in 1986 with some improvements over (Fletcher, 1985) for classifying entities like "dolls" are "toys". Schemas are templates for problem solving. (Bakman, 2007) extends schemas to handle extraneous information and can solve multi-step problems unlike its predecessors. In a review of such knowledge-based systems by (Mukherjee and Garain, 2009), it was remarked that there are no datasets to compare performances. The recent empirical methods that are cited below provided datasets that can now be used for evaluation. For example, (Kushman et al., 2014) proposed a word problem solver which uses statistical analysis to solve word problems. They handled a more complex class of word problems which involve solving a set of simultaneous linear equations. More recently, (Hosseini et al., 2014) attempted the same addition/subtraction word problem framework where they concentrated on classifying the verbs in a given problem into semantic classes. They were able to show remarkable improvement over (Kushman et al., 2014). They used a state representation to depict temporal ordering. However, the system could not reason about existing scenarios like questions that involved comparisons. There is some development in this direction by the mathematical modelling conglomerate WolframAlpha (2015).

3 Architecture

The architecture of the system is described in Figure 1. The given problem is first simplified such that every sentence in the new problem has exactly one verb. Then, linguistic information is extracted from each sentence and passed on for the Knowledge representation. For example, "Ruth has 3 apples" will be translated as "owner = Ruth; entity.name = apples, entity.value = 3; verb = has".

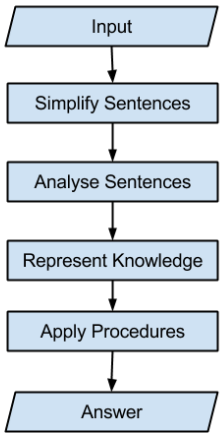

Figure 1: Architecture of Word Problem Solver

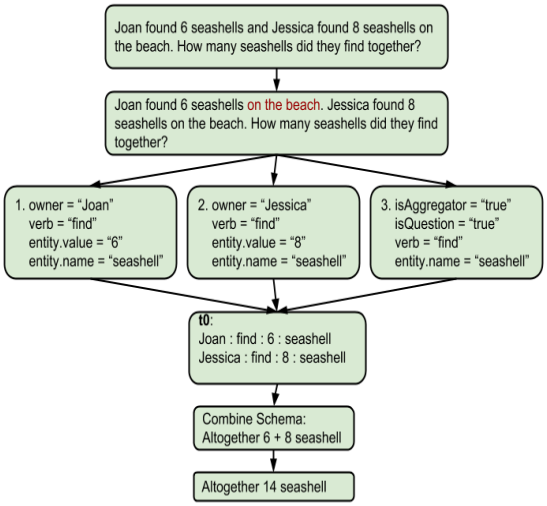

Figure 2: Example of the Problem Solving Process

During the knowledge representation phase, all the sentences are taken into consideration the events are ordered according to time. Then, the problem is solved using the procedures stored in it. These units are explained in more detail in the following sections. An example that depicts how a problem is solved is shown in Figure 2.

4 Knowledge Representation

4.1 Schemas

A popular representation idea is to use schemas. Schemas are templates that describe how entities interact. In this discussion, the schemas described by (Bakman, 2007) are explained. For instance, "John had 3 apples. He forfeited 1 apple. How

many apples does he have now?" will trigger the "Transfer in Ownership" schema because of the keyword "forfeit". This situation is described below.
(owner) had (X) (object)
(owner) (transfer-in-ownership) (Y) (object)
(owner) has (Z) (object)

To elaborate, each sentence is examined sequentially until a keyword is encountered. In the second sentence, the word "forfeit" is such a word that maps to the "transfer-in-ownership" schema. This is matched against the second statement in the schema description and the owner is identified as "John" and Y = 1 and object as "apple". Now the problem is re-examined searching for sentences with "John" and "apple" to match.
(owner) = John, (X) = 3, (Y) = 1, (object) = apple, (transfer-in-ownership) = forfeit
Y + Z = X.
The equation is attached to every schema.

ROBUST, the system developed by (Bakman, 2007), was able to solve multi-step problems with extraneous information.

Disadvantages of ROBUST:

- Sometimes, due to the lack of some implicit common sense knowledge, the problem cannot be solved. To illustrate, "A farmer bought 3 apples and lost 2 of them. How many fruits does he have now?". This problem cannot be solved by schemas as there is no explicit statement saying the farmer had no apples initially.
- There is a search overhead to match sentences against all possible schemas because all the sentences are examined whenever a keyword is encountered.

4.2 Temporal Schemas

In this paper, by using time, default assumptions and heuristics, a large number of problems are solved. To illustrate, the following problem is taken.
"John buys 11 peaches. He eats 9 peaches. How many does John have now?"
The first sentence is represented as:
t_0
$John : has : peaches : x_1$
$unknown_0 : has : peaches : x_2$
t_1
$John : buy : peaches : 11$
t_2
$John : has : peaches : x_1 + 11$
$unknown_0 : has : peaches : x_2 - 11$

This represents the knowledge that the keyword "buy" signifies a "Change Out" in ownership for "John", the buyer and "$unknown_0$", the unnamed seller. As we don't know some values, they are kept as variables. The time stamp t_0 is the initial default one.

When the next sentence is read, "John eats 9 peaches", it is updated as:
t_0
$John : has : peaches : x_1$
$unknown_0 : has : peaches : x_2$
t_1
$John : buy : peaches : 11$
t_2
$John : has : peaches : x_1 + 11$
$unknown_0 : has : peaches : x_2 - 11$
t_3
$John : eat : peaches : 9$
t_4
$John : has : peaches : x_1 + 11 - 9$
$unknown_0 : has : peaches : x_2 - 11$

Now, once the system encounters the question, it retrieves the most recent value of John's peaches. Hence, it retrieves "x_1 + 11 - 9". By making an assumption that John had no peaches in the beginning, we compute "11-9", that is "2" and display the answer.

There are situations when an expression has to be substituted by a value. For example, John may have 6 apples. Then, we come to know that John has 3 more apples than Mary and we do not know how many apples Mary has (say x apples). After applying the schema, we know that John has (x+3) apples. When John has two consistent values at the same time, an equation 6 = 3 + x is created and solved. The fact that Mary has 3 apples is stored.

Sometimes, we may require a value that occurs in the first time step. By analysing the tense and narrative order, the correct value to be retrieved can be ascertained. For example, if the first two sentences are in present tense and the question is in past tense, the value in the first time step is retrieved. The reason information such as "John buys 11 apples" is stored is because sometimes, a question may be how many John bought or how many John ate.

4.3 List of Schemas

In the schemas we have designed, no sentence is explicitly matched against a template. Whenever a keyword is encountered, this information is adjusted according to its corresponding procedure. If the schema requires some information which is not available, a variable is introduced. If the value that corresponds to that variable is seen later, it is replaced in all the expressions that contain the variable. The list of schemas used is given in Table 2.

Name	Schema	Update
Change In	$Owner_1$ has X objects. $Owner_2$ has Y objects. Z objects were transferred from $Owner_2$ to $Owner_1$	$Owner_1$ has (X+Z) objects. $Owner_2$ has (X-Z) objects.
Change Out	$Owner_1$ has X objects. $Owner_2$ has Y objects. Z objects were transferred from $Owner_1$ to $Owner_2$	$Owner_1$ has (X-Z) objects. $Owner_2$ has (X+Z) objects.
Combine	$Owner_1$ had X objects. $Owner_2$ had Y objects. Together, they have Z objects.	Z = X + Y
Compare Plus	$Owner_1$ had X objects. $Owner_2$ had Y objects more than $Owner_1$.	$Owner_1$ has (X+Y) objects.
Compare Minus	$Owner_1$ had X objects. $Owner_2$ had Y objects less than $Owner_1$.	$Owner_1$ has (X-Y) objects.
Increase	$Owner_1$ had X objects. $Owner_1$ got Y objects more.	$Owner_1$ has (X+Y) objects.
Reduction	$Owner_1$ had X objects. $Owner_1$ lost Y objects.	$Owner_1$ has (X-Y) objects.

Table 1: List of Schemas

4.4 Common Sense Law of Inertia

When the number of discrete time steps increase, unaffected owners and entities are assumed to retain their value as per the common sense law of inertia (Shanahan, 1999). For example, "John has 3 apples. Mary gave 4 apples to Tom" implies that at the second time step also John has 3 apples.

5 Natural Language Processing

The natural language processing is done during the simplification and the analysis phase. The aim of the simplification phase is to make the text as unambiguous as possible and to make it amenable for analysis. After the analysis is completed, the relevant information needed for knowledge representation would have been extracted. These tasks, which are described below, extensively used the Stanford CoreNLP suite (Manning et al., 2014).

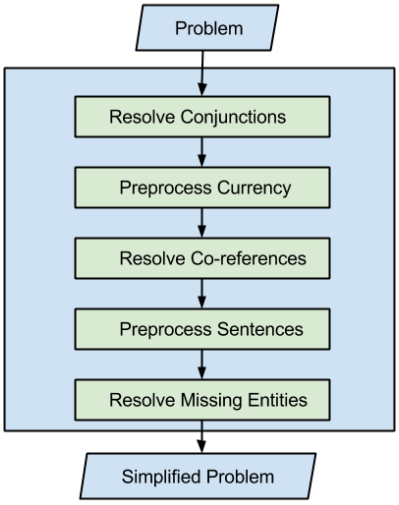

Figure 3: Simplifying the Problem

5.1 Problem Simplification

The steps involved in simplifying the problem is described in Figure 3.

5.1.1 Resolving Conjunctions

The first step of simplification is to deal with sentences that contain "and", "but", "if", and "then". To illustrate, take the example "John had 5 apples and ate 2 of them". The fact that "John" is the person who ate 2 apples can be made clearer if this sentence was split into the following sentences - "John had 5 apples. John ate 2 apples."
Algorithm:

- Separate the sentence into two parts, Part1 and Part2, by splitting at the conjunction.
- If the first part has no verb, then no processing is required. This is to handle situations like "Ram and Sita have 13 apples altogether". It does not make sense to split this sentence as "Ram has 13 apples. Sita has 13 apples.".
- Split each part into four strings, pre-verb (P), verb (V), after-verb (A), preposition- phrase (PrP). Some of these strings may be empty.
- Whenever a string of Part2 is empty, copy the corresponding string from that of Part1.
- Concatenate these four strings into a single sentence and return the two sentences.

This is depicted for a typical example in Figure 4

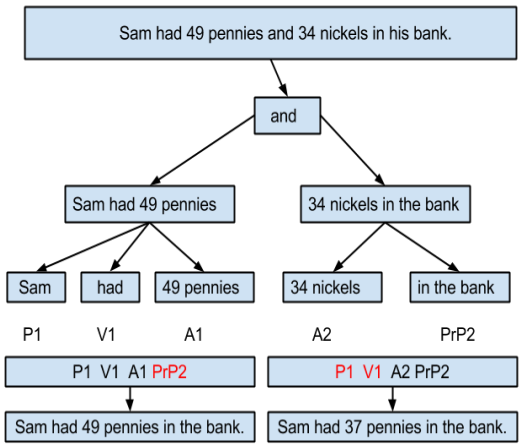

Figure 4: Resolving Conjunctions

5.1.2 Preprocessing Currency

The dependency parser provided by (Manning et al., 2014) links numbers to their corresponding entities using the "num" edge or the "number" edge. To maintain this consistency, "$5.5" is replaced by "5.5 dollars".

5.1.3 Resolving Co-references

The task of resolving co-references was described in the introduction. To reiterate, it has been used to map pronouns to the referring entity. The system uses the resolver provided by (Manning et al., 2014). Some heuristics are used to improve its current performance. These are listed below.

- If a pronoun refers to another pronoun, it is not considered. This is to avoid recursive computations.
- "They" and "their" are ignored. This is because the mapping found for these pronouns are often wrong. For example, the input "Ram has 6 apples. Sita has 7 apples. How many apples do they have in all?" results in "apples" being mapped to "they".
- A inherent ambiguity is exemplified by "Sam has 13 dimes. His dad gave him 3 dimes". Here, "him" is always mapped to the "dad" instead of "Sam". In word problems, such a pronoun mostly refers to someone else and the program uses this knowledge and maps it to "Sam" instead.

5.1.4 Preprocessing sentences

The sentences are simplified such that each sentence has exactly one verb. For example the sentence "John had 5 apples and Mary gave John 3 apples." is parsed by the Stanford Core NLP Suite(Manning et al., 2014) as follows:

Parser output (Stanford Core NLP) :

ROOT (S (S (NP (NNP John)) (VP *(VBD had)* (NP (CD 5) (NNS apples)))) (CC and) (S (NP (NNP Mary)) (VP *(VBD gave)* (NP (NNP John)) (NP (CD 3) (NNS apples)))) (. .)))

The corresponding phrase is extracted and converted into two sentences.

John had 5 apples. Mary gave John 3 apples.

Sometimes the verb phrase encompasses another verb phrase, hence this process is done recursively until every sentence has exactly one verb. Sometimes it is not possible to split sentences such as "How many apples did he have?". Here, both "did" and "have" are identified as verbs. Thus, such sentences are not simplified further.

5.1.5 Resolving Entities

This step is to address problems of the type "John has 5 apples. He gave 3 to Mary". Here, 3 is not explicitly specified as 3 apples. So, all plural entities are stored as potential entities and every time a number is not followed by a noun or an adjective, one of them is chosen. Their adjectives are also retained such that a typical entity list may have (balloons, red balloons, green balloons). The most suitable entity is chosen based on the sentence order. Most problems deal with just one type of entity.

5.2 Analysis of Simplified Problem

At the end of this phase, the following information would be available for representation.

- A list of all owners such as "John" and "basket".
- A list of all entities like "green balloons" and "red balloons".
- A list of processed sentences.

Each processed sentence would in turn have the following information.

Procedure	Keywords
Change Out	put, place, plant, add, sell, distribute, load, give
Change In	take from, get, pick, buy, borrow, steal
Increase	more, carry, find
Reduction	eat, destroy, spend, remove, decrease
Compare Plus	more than, taller than, longer than, etc.
Compare Minus	less than, fewer than, shorter than, etc.
Combine	together, in all, combined

Table 2: Schemas and their Keywords

- isAggregator - whether the sentence contains any aggregating phrases such as "altogether", "in all" etc.
- isComparator - whether the sentence contains any phrase such as "more than", "less than", "fewer", "taller" etc.
- isDifference - whether the sentence contains any phrase such as "left", "remaining" etc.
- isQuestion - whether the sentence denotes a question such as "Find the number of apples John has".
- keyword - whether the sentence contains any of the keywords listed in Table 2.
- schema - if there is a keyword, the schema linked with it
- $owner_1$ - the first owner in the sentence - "John gave 3 apples to Mary" would set $owner_1$ to "John" and $owner_2$ to "Mary".
- $owner_2$ - the second owner in the sentence
- entityName - the name of the entity in the sentence such as "apples".
- entityValue - the value connected to the entity - such as 3 in the previous example.
- tense - this is derived from the POS tag of the verb and is used for ordering the sentences in the representation.

To get this information, the dependency parser has been extensively used. Consider "Mary gave 3 apples to John."
Dependency Parser output (Stanford Core NLP) :
- gave/VBD (root)
- Mary/NNP (nsubj)
  - apples/NNS (dobj)
  - 3/CD (num)
  - John/NNP (prep-to)

From this tree, the two owners Mary and John are identified as well as the entity "3 apples". As a general rule, the subject is identified as the first owner and the noun associated with a preposition is taken as the second owner. Every number is identified as "num" and that is taken as the value of the entity with the corresponding node of the edge being taken as the entity name. The verb is identified from the Part-Of-Speech (POS) tag "VBD" and stored.

If the sentence contains "some" or "few", the value of that entity is set to some variable. Sometimes, especially for word problems that involve decimals, the parse may not correctly map the entity name to the entity value. Hence, there is a check if there is any number in the sentence that has not been assigned an entity and if so, the nearest plural noun is set as the entity's name.

For a question, the entity's value is not available. Hence, the sentence is examined and checked against the list of existing entities and owners to ascertain what the question is about.

6 Some Heuristics

A list of heuristics are used to increase the performance. However, like all heuristics, they are not precise.

- Consider, "Sam grew 41 watermelons but the rabbits ate 5 watermelons". Usually, "eat" would denote that there is a reduction in one's own entities, i.e., the rabbits should have had some watermelons. In such a situation, if that information is not available, it is mapped to a reduction in the watermelons possessed by "Sam".
- As mentioned before, if there are two possible answers due to ambiguity in the question, then the answer which involved some calculation or which has no unknown terms is preferred. Consider, "There are 2 pencils in the drawer. Tim placed 3 pencils in the drawer. How many pencils are now there in total?". So the system would correctly display "drawer has 5 pencils" instead of "Tim has unknown pencils".
- If the final answer that the system derived is simply an entity mentioned in the question

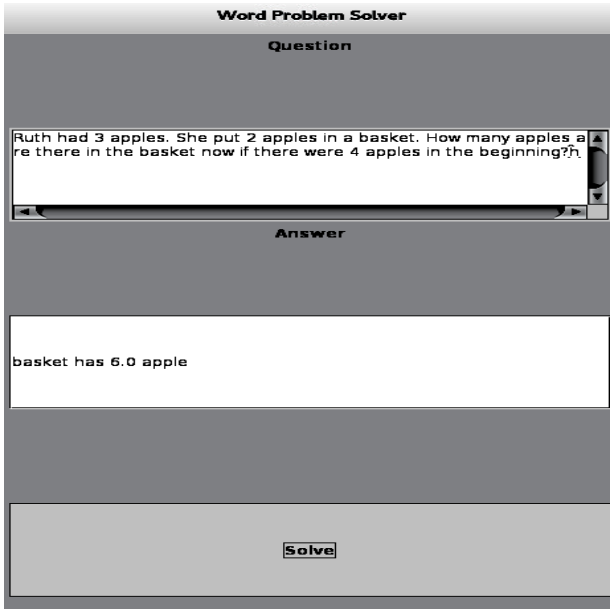

Figure 5: Screenshot of the System

and if it is known that the question has an aggregator like "altogether", then the system bypasses the representation and returns the sum of all entities.

7 Illustration

A screenshot of the system is available in Figure 5.

8 Experiments

The experiments are similar to the ones performed by (Hosseini et al., 2014). They had 3 datasets (DS1, DS2, DS3) with increasing difficulty in terms of natural language and irrelevant information. Our system's performance is compared against ARIS proposed by (Hosseini et al., 2014), ROBUST proposed by (Bakman, 2007) and the online mathematical query system (WolframAlpha, 2015). Also, as we are not learning the verb categorization, that is the keyword to schema mapping, the appropriate comparison is with "Gold ARIS" which is ARIS with perfect categorization.

	DS1	DS2	DS3	Avg
Our System	**96.27**	**80.00**	**90.08**	**88.64**
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 3: Comparison with Existing Systems

ROBUST fails almost completely with the two complex datasets as its NLP is quite weak. WolframAlpha also does not scale well. However, they generate an answer in natural language. ARIS does not present the answer in natural language though it has the means to do so. The representation used by ARIS and our system is similar but there is more reasoning involved in our system by exploiting non-verb keywords and with some heuristics. Also, ARIS exploited only one type of schema, that is "Change" schema and did not address "Combine" and "Compare" schemas completely. (Hosseini et al., 2014) analysed the errors and classified them into five types - set completion, parser issues, irrelevant information, implicit knowledge and others. Our system was able to reduce errors in these areas. The only category of errors that remains unaffected is set completion. The system is not able to solve problems such as "Sara 's high school played 12 basketball games this year. The team won most of their games. They were defeated during 4 games. How many games did they win?". By using recognizing antonyms, these error can also be reduced in a future version. The errors involving parser issues, irrelevant information and implicit knowledge are reduced by the use of heuristics. Other issues are resolved significantly because of more information available about comparisons and combinations. For example, "In March it rained 0.81 inches. It rained 0.35 inches less in April than in March. How much did it rain in April?" can be solved by our system because of the keyword "less". ARIS fails because the verb "has" does not signify any operation.

9 Discussion

With the availability of sophisticated parsers, we are able to revisit age-old language understanding problems and get much better results. While traditional knowledge based approaches gave a good base for representation, their NLP was quite limited due to the limitations in technology at that time. On the other hand, the recent empirical systems performed much better language processing but were semantically limited in comparison by not exploiting all types of word problems.

Our system also offers the answer in natural language and in real time. Though the

performance of WolframAlpha is poor, it provides visually pleasing explanations for the problems it can solve and presents the answer in natural language as well. It is also capable of analysing phrases such as "two times" that involve multiplication. One sure direction of future work is to generate good explanations with our proposed system.

10 Conclusion and Future Work

We developed a system that solves addition/subtraction word problems with considerable accuracy over the existing systems. It takes a problem, simplifies it and then extracts information such as owners, entities and keywords from every sentence. This extracted information is represented in a temporal representation and then presents the answer in natural language.

The system was developed in Java and the code is available at `https://github.com/Sowms/AdditionSolver/tree/KRexperiment`. The semantics can be improved greatly by using a more complex representation method such as Event Calculus described in (Shanahan, 1999) which is natural choice for temporal reasoning. Also, the semantics for units is not well defined. For example, consider "Marta picked 2 pumpkins. The first pumpkin weighed 4 pounds, and the second pumpkin weighed 8.7 pounds. How much did the 2 pumpkins weigh all together?". The question can refer to two entities, "pound" or "watermelon". It arbitrarily chooses "watermelon" and initially reports the answer as "Marta picked 2 watermelon". At this point, the system recognizes that this fact is already present in the problem and that there is an aggregator "altogether" and uses a heuristic to report the answer as "Altogether 14.7" instead of "12.7".

Also, more work can be done to improve the problem simplification, especially for the longer word problems that involve decimals.

To conclude, a system for solving addition/subtraction problems has been presented. It will hopefully be one of the many steps forward on a long journey towards representing knowledge that can lead to better question answering systems and can help teach the eager student.

References

WolframAlpha. Wolfram Research, Inc., Mathematica, Version 10.2, Champaign, IL (2015). http://blog.wolframalpha.com/2012/10/04/solving-word-problems-with-wolframalpha/

Princeton University "About WordNet." WordNet. Princeton University. 2010. http://wordnet.princeton.edu

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman 2014. *Learning to Solve Arithmetic Word Problems with Verb Categorization*. Proceedings of the Conference on Empirical Methods in Natural Language Processing.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay 2014. *Learning to Automatically Solve Algebra Word Problems*. Proceedings of the Conference of the Association for Computational Linguistics (ACL).

Christopher D. Manning, Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and David McClosky. 2014. *The Stanford CoreNLP Natural Language Processing Toolkit*. Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60

Anirban Mukherjee and Utpal Garain 2009. *A review of methods for automatic understanding of natural language mathematical problems* . Artificial Intelligence Review, Volume 29, Issue 2, pp 93-122

Y. Bakman. 2007. *Robust understanding of Word Problems with Extraneous Information*, http://lanl.arxiv.org/abs/math.GM/0701393

Shanahan, M. 1999. *The event calculus explained*, Wooldridge, M., & Veloso, M.(Eds.), Artificial Intelligence Today, pp. 409430. Springer Lecture Notes in Artificial Intelligence no. 1600.

D. Dellarosa. 1986. *A computer simulation of childrens arithmetic word-problem solving*, Behavior Research Methods, Instruments, and Computers Volume 18, Issue 2, pp 147-154

Charles R. Fletcher *Understanding and solving arithmetic word problems: A computer simulation.*, Behavior Research Methods, Instruments, and Computers, 17, pp 565-571.

Daniel Bobrow. 1964. *A question-answering system for high school algebra word problems.*. AFIPS 64 (Fall, part I) Proceedings of the October 27-29, 1964, fall joint computer conference, part I, pp 591-614