Learning to Solve Arithmetic Word Problems using Sentence Simplification

First Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Second Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

This paper presents a sentence simplification approach to learning to solve arithmetic word problems. The approach performs a thorough analysis of each sentence in the given word problem to map each sentence to a simplified syntactic pattern. The syntactic pattern is generated by parsing the sentence using a dependency parser and encapsulates all the relevant information in a sentence including the subject, verb and object with its quantity as shown in Figure 1. The pattern is then used as a feature alongside other features as described in the Features section to classify the operator of the object given in the sentence. The classifier is trained on a small dataset of sentences and their operators and is not manually annotated. An equation is generated using similar information from multiple sentences in a given word problem.

Using this approach the system learns to classify the operator with 70% and is able to solve 70% of the problems in the public dataset released by (Hosseini et al., 2014) and can be found online[1]. The system overcomes some of the parsing issues encountered in attempt to solve word problems before and needs some improvement around the relational errors.

1 Introduction

Mathematical word problems are generally a sequence of actions by a subject on an object. The challenge in solving mathematical word problems is to extract information from a sentence accurately. Converting the extracted information to an equation is a trivial part. The complexities in extracting information increases when the sentence refers to an entity encountered in the previous sentences or some other related entities such as dollars and money.

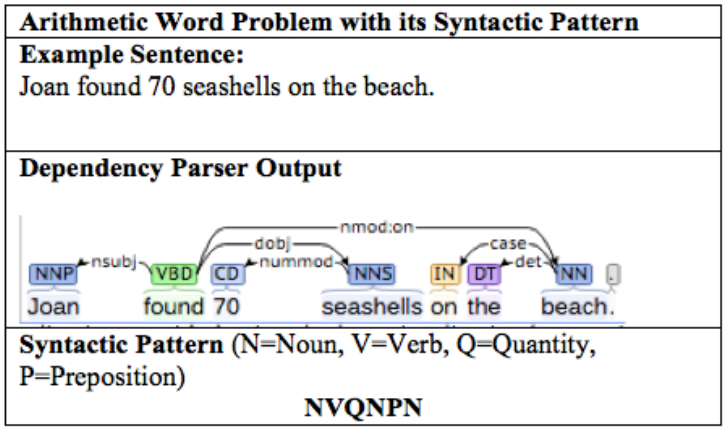

Figure 1: Example sentence of a word problem and its syntactic pattern.

When taking word problems into consideration, Algebra may be a challenging part from a child's point of view but the semantics of a sentence such as understanding of the syntax, extensive knowledge,

[1]Dataset used is available at https://www.cs.washington.edu/nlp/arithmetic.

coreference resolution and extracting information from individual sentences and combine that information effectively to produce a result.

Our system needs to parse the sentences in a word problem well to fetch useful information. This information needs to be used collectively in an effective way to generate an equation. Solving the equation if generated correctly is a trivial part. Figure 1 shows an example of how a sentence from a word problem is simplified to a syntactic pattern. This paper approaches the problem of solving arithmetic word problems by mapping individual sentences in a word problem to a syntactic pattern. Based on the syntactic pattern each sentence is simplified till it represents minified information which consists of a subject, verb, object with its quantity and preposition if there exists one. This simplified sentence is then classified to an operator and the object with its quantity is added to the equation with the classified operator. Each simplified sentence is classified to one of four operators as described in Table ?.

We gather information form the sentences by parsing the sentence using the dependency parser [2]. Based on the Part-of-speech tags and relations between the words from the sentence different entities from the sentence are extracted. Expletives, nouns and their quantities, verbs, conjunctions, adjectives and prepositions are extracted from the sentence and are related to each other based on the relations output by the dependency parser. Based on some rules these extracted entities are used collectively to form a simplified sentence and is provided as an input to the classifier. As explained above the classifier predicts an operator for the simplified sentence. Now that we have the operator we use the operator and the extracted entity in the equation. Consider the below example:

Sentence: Sara picked 6 pears and Tim picked 5 pears from the pear tree.	
Simplified Sentences: Sara picked 6 pears from the pear tree. Tim picked 5 pears from the pear tree.	
Extracted Entities	
Sara picked 6 pears from the pear tree.	Tim picked 5 pears from the pear tree.
Sara	Tim
6 pears	5 pears
Pear tree	Pear tree
Predicted Operator	
+	+
Add to Equation	
+ 6 Pears	+ 5 pears

Figure 2: Example sentence of a word problem and its simplification.

Simplifying the sentences to such format makes this less complex for the classifier as well as to generate equations. Though the training data seems to be the sentences, actually the sentences are just a way to generate syntactic patterns which are used extensively as the features to the classifier. A large number of sentences will end up having the same syntactic pattern. Hence, training is actually on syntactic patterns and not on raw text. Our system is evaluated on an existing dataset of arithmetic problems provided by (Hosseini et al., 2014).

Our system learns this classifier based on 300 individual sentences. Our data and source code are publicly available [3]. The next section describes how we extract the syntactic pattern, its usage and problem decomposition.

[2] http://nlp.stanford.edu/software/stanford-dependencies.shtml

[3] Source code and training dataset is available at `https://github.com/vishalrajpal/TrySystem`.

2 Syntactic Pattern and Problem Decomposition

In this section we describe how we use the dependency parser output to extract the syntactic pattern of a sentence and the process of simplifying sentences using the extracted syntactic pattern. We further describe the way we predict the operator of the simplified sentence using a Logistic Regression classifier. The input to our system is a problem text and we carry out multiple steps for each sentence in this problem text. Below are the steps:

2.1 Run Dependency Parser on Each Sentence

For each sentence in the problem text, we extract the relations and the part-of-speech tags using the dependency parser. Based on the output from the dependency parser we extract the Expletive if any, Nouns and their quantities, preposition, conjunction etc. and build a syntactic pattern based on the ordering of the words in the sentence as described in Figure 1. The representation of a sentence as a syntactic pattern helps us fetch information about this sentence easily such as if the sentence has a quantified noun or a conjunction, an expletive or a preposition etc. We use this information to further simplify the sentence as described in further steps.

2.2 Parse Commas in Each Sentence

In most arithmetic word problems, commas are a way to either multiple subjects interacting with the same object or a subject interacting with multiple objects. Hence, to simplify sentences comma is an important punctuation character. After extracting the syntactic pattern of the overall sentence, the sentence is looked for commas. If any of the commas exist based on some rules we decide if the text before comma belongs to the subject part of the sentence or the object part of the sentence. The goal of this step is to have simplified sentences based on parsing commas in the sentence. The current sentence is replaced by simplified sentences extracted by parsing the sentence based on a comma. Consider the below example:

Sentence: Joan picked 43 apples from the orchard, and gave 27 apples to Melanie.
Simplified Sentences: Joan picked 43 apples from the orchard. Joan gave 27 apples to Melanie.

Figure 3: Simplification of a sentence based on a comma.

2.3 Run Dependency Parser on Each Sentence

Now that we have simplified sentences extracted by parsing the sentence based on a comma, we give these simplified sentences to the dependency parser to update the syntactic pattern. The syntactic pattern is extracted in the same way as in Step 2.1. We further apply some rules to extract more simpler sentences.

2.4 Parse Conjunctions in Each Sentence

Similar to commas, in most arithmetic problems conjunctions are a way mostly to specify two subjects interacting with a single object, or a single subject interacting with two objects. Based on the updated syntactic pattern from previous step looking if a conjunction exists is trivial. If a conjunction exists we simplify the sentence based on some rules. The goal of this step is to have multiple simplified sentences based on parsing by conjunctions. The current sentence is replaced by simplified sentences extracted by parsing the sentence based on a conjunction. Below is an example where a conjunction ”and” exists:

Sentence: Joan grew 8 watermelons and 4 turnips.
Simplified Sentences: Joan grew 8 watermelons. Joan grew 4 turnips.

Figure 4: Simplification of a sentence based on a conjunction.

2.5 Run Dependency Parser on Each Sentence

At this point, the sentences are simplified at the most granular level. Since, each sentence is replaced by multiple simplified sentences their will be multiple syntactic patterns. We provide these simplified sentences to the dependency parser and based on the same rules used in the above steps we extract the useful information such as Nouns, verbs etc. Based on the information extracted the features for each sentences are extracted from the syntactic pattern and other properties as described in ?. At this point, we have a feature vector for a single sentence of the problem text. Each sentence of the problem text will be simplified to multiple sentences and each simplified sentence would have its own feature vector which would classified to an operator as discussed in the next step.

2.6 Features

2.6.1 Syntactic Patterns

All the syntactic patterns extracted from the training data are used as individual features. The syntactic pattern to which the sentence belongs is et and all the other are not in a one-hot encoding way. From the training data of 300 sentences, we extracted near to 125 syntactic patterns, which shows that multiple sentences have a similar syntactic pattern. For text data, even if we haven't seen a pattern before the other features as described below will try to map the syntactic pattern found to the existing syntactic pattern.

2.6.2 Count based Features

Based on our analysis we found that having a count of some important things in the syntactic pattern proves to be helpful in to the classifier in learning accurately. There are 4 count based features: number of nouns, number of prepositions, number of verbs and number of quantities. These features depend on how we map the sentence to a syntactic pattern and as explained above if we map the sentence to the correct syntactic pattern, extracting these kind of features is a trivial task.

2.6.3 Occurrence based Features

Relation based Occurrence Features We use the occurrence of some dependency relations as features. Particularly nmod:poss relation from the dependency parser if set can determine the subject possessing the object and nmod:of which helps to differentiate between two nouns of same kind. The relations in the sentence depict its structure and to simplify the sentences the properties of its structure prove to be helpful.

Pattern based Occurrence Features Occurrence of some specific syntactic pattern as a sub pattern of the syntactic pattern of a sentence can also be helpful. We have 2 features: NVQN syntactic pattern and QN syntactic pattern. These two patterns indicate that a sentence of such pattern is mostly suggesting addition or subtraction. Since we are trying to simplify the sentences to a similar pattern, they prove to distinctive features between quantified classes("+", "-") and other classes("=", "?").

Word based Occurrence Features Some specific words allow the classifier to predict more accurately. The word "now" is suggesting a result after some operation and hence the predicted operator should be "=". Similarly, occurrence of one of the words from "most", "some" and "several" suggests an unknown quantity in the sentence. There are different such words through which we can predict the operation and hence we use the occurrence of such words as features.

2.6.4 Word similarity based features

We identified 8 word categories from WordNet and Word2Vec which can be helpful in predicting the operator. The 8 verb categories are "verb.possession", "verb.change", "verb.communication", "verb.consumption" , "verb.contact", "verb.creation", "verb.motion" and "verb.weather". The verbs in the sentence are given as input to WordNet and Word2Vec and the result is normalized between these categories.

2.7 Training the Classifier to Predict Operator

At this point we only consider addition subtraction problems but based on our approach and analysis we believe it is generalizable to multiplication and division problems. The possible operators in our system currently are "+", "-", "=" and "?". The "+" operator and "-" are self explanatory, whereas a sentence classified as "=" is suggesting that the quantity in the sentence is a result of some operations and a sentence classified as "?" is suggesting that the sentence is asking a question about some entity.

We train four Logistic Regression binary classifiers for each of the above mentioned operators. We then use one vs all technique to combine the results from these binary classifiers. Our test data contain 55 simplified sentences. The below table consists of the precision and recall for all the 4 classes:

Operator	No. of instances	Precision	Recall
+	30	84%	70%
-	8	30%	37.5%
=	2	33.33%	50%
?	15	88.23%	100%

Table 1: Precision and Recall

Based on the results above, it is clear that operator "?" had some specific features such as presence of WH-adverb which made it easy for the classifier to predict question sentences correctly. Preposition occurrence features provided distinctive information to the classifier for operator "+". Operator "-" is entirely dependent on verb features and some preposition occurrence features such as presence of preposition "from". Whereas operator "=" is dependent on some specific word occurrence features such as now. Below is an example of some sentences classified correctly:

Sentence	Actual Operator	Classified Operator
Sandy has 24 red balloons.	+	+
He took 20 seashells from Joan.	-	-
He now has 21 left.	=	=
How much sugar is remaining?	?	?

Table 2: Example Sentences

2.8 Building and Solving Equation

Based on the operator prediction for a single simplified sentence from the classifier, the quantified noun from sentence is extracted with the predicted label. When our system encounters a sentence predicted as "?" it extracts the Noun for which the computation needs to be done. All the other sentences are looked for similarity with this entity and are merged into an coherent whole equation if they are similar. Below is an example of an equation generated by our system:

Problem Text	Equation Generated by our system
Sandy grew 6 carrots . Sam grew 3 carrots . How many carrots did they grow in total ?	+6 carrots +3 carrots = ? carrots

Table 3: Example Problem Text and Equation

3 Experimental Results

We evaluate our system on publicly available datasets of arithmetic word problems in this section.

3.1 Datasets

We evaluate our system on 3 publicly available datasets.

3.1.1 A12 Dataset

This dataset consists of 395 addition and subtraction problems, released by (Hosseini et al., 2014). They performed a 3-fold cross validation and each fold consists of problems from different sources. We evaluate our system on this dataset with the same settings and report our results.

3.1.2 IL Dataset

This dataset consists of 562 problems released by Roy:15. We only evaluate on the addition and subtraction problems for now as our system is able to solve addition and subtraction problems.

3.1.3 Commoncore Dataset

This dataset consists of 600 problems among which there are 200 addition and subtraction problems. We evaluate our system on these 200 questions and report our results.

4 Discussion

4.1 Future Work

4.2 Detailed Error Analysis

5 Conclusion

References

Mohammad Javed Hosseini, Hannaneh Hajishirzi, Oren Etzioni and Nate Kushman. 2014. *In Proceedings of the EMNLP*