4th Grade Arithmetic Word Problem Solver

Submission Date: 18/12/2016 **Group no: 6**
Students: **1)Kunal Shah (110974215)** **2) Rashmi Makheja (110920870)** **3) Sanket Dige (110815356)**

A. Introduction

i. Problem Definition :

Being fascinated by the field of Natural Language Processing (NLP), we aimed to study the state of the art applications of NLP and get acquainted with Parts Of Speech (POS) tagging. We choose the topic of 4th grade arithmetic Question answering. An advanced real world system for solving mathematics problems is WolframAlpha[1]. However, our focus was on studying and experimenting with the different techniques discussed in various papers.

We present four basic models, discussed in detail below. At an I/O level, each model will take a basic arithmetic word problem, solve it and output the answer. For example, You can ask our model the following question:

"Jack ate 10 apples and gave 10 apples to Jill. How many apples did Jack have?"

The model will give output : ***20***

An another question could be:

" There were 100 trees in the garden. 10 trees fell down. How many trees are there now in the garden?"

The result will be ***90***

ii. Motivation :

To prepare a child for future complex algorithms, it is important to teach him the basics of mathematics and language. Similarly, to create an high level AI agent which can solve bigger problems or improve the search engines, it is necessary to start with the basics. A 4th grade arithmetic question involves simple equations and the english used is comparatively less ambiguous. An advanced question answering system can help students to study the step by step solving of different problems and also greatly increase the accuracy of search engine results. For example, when we wish to search something on Google, we often have to try different phrases to get the desired results. An improved NLP oriented question answering algorithm can help solve this gap.

iii. Contributions:

As part of this project we tried to apply the classifiers and POS tagging to solve arithmetic word problems.

Model 1 - Logistic Regression and POS tagging : Classify data into positive or negative problems, and perform operations based on this classification.

Model 2 - Slight variation in Model 1: Added a filtering step to Model 1. Use of POS tagging to classify data.

Model 3 - A brute force approach: Used individual sentences to classify the verbs into three classes: assign, transfer positive, and transfer negative. Classifier is only used to create a list of verbs with their respective classification. Test problem is scanned for verbs and solved using this list.

Model 4 - Recursive brute force approach : The classifier is now also used to predict the sentences in test problem into four classes: assign, transfer positive, transfer negative and final. If there are multiple sentences in the problem, then it will take one at a time and recursively apply brute force till answer is obtained.

iv. Observations:

Each model tries to overcome the limitations of previous model.

Model 1 - Logistic Regression and POS tagging : It works in cases where the test problem has cardinals that contribute to the final output. Also, the nouns used in the training set should not be highly probable in any one of the class. This is discussed in detail in 'Description' Section.

Model 2 - Slight variation in Model 1: It removes the dependency on nouns. But will still fail if the problem contains cardinals that are not important and do not contribute to final answer.

Model 3 - A brute force approach: This model overcomes the drawback of above models, but works only for cases where the problem will have one variable and three cardinals.

Model 4 - Recursive brute force approach : So far, this model is our best approach. It overcomes the limitations of all the models. It achieves 60% accuracy in solving the problems.

B. Description

Model 1 - Logistic Regression and POS tagging :

Before starting with NLP directly, we thought of first analyzing what might a 4th grade student or for that matter we as adults think when given a simple word problem. We generally observe the quantities given and try to figure out what to do with these numbers: add them

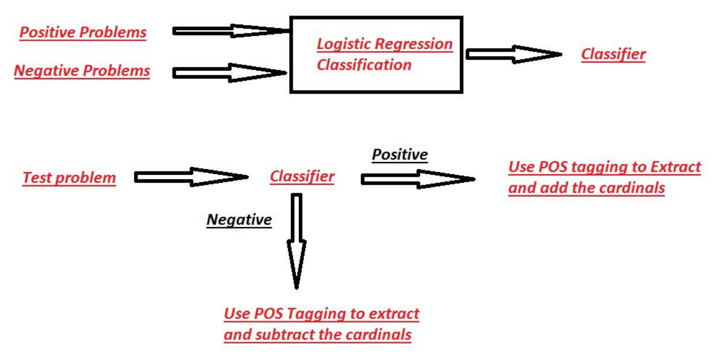
[1] https://www.wolframalpha.com/

or subtract them. This is the exact observation that we make our classifier learn so that we know whether to add the cardinals or subtract them.

Pseudo code:

```
1. Divide the training set into positive and negative classes.
    a. Positive class: the questions that require addition.
    b. Negative class: the questions that require subtraction.
2. Using logistic Regression and selecting only 35% of the features, train the classifier.
3. Each test dataset problem is classified into positive or negative class using classifier.
4. Using the nltk library POS tagging, extract the cardinal numbers.
5. If the problem is positive, add the extracted cardinals, else subtract the cardinals.
6. Output the result.
```

Model 1 - Logistic Regression and POS Tagging


Positive Problems
Negative Problems
Logistic Regression Classification
Classifier
Test problem
Classifier
Positive
Use POS tagging to Extract and add the cardinals
Negative
Use POS Tagging to extract and subtract the cardinals

Limitation:

Used complete sentence to train classifier and predict the test data (Detailed explanation in Model 2)

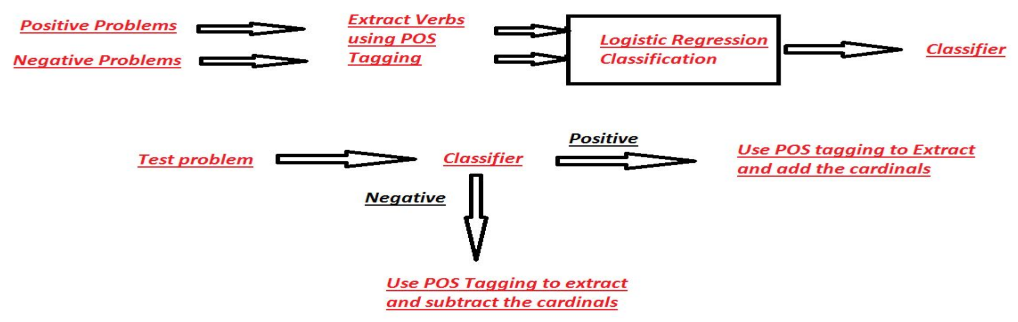
Model 2 - Slight variation in Model 1:

Model 1 used the complete sentence to classify and predict the sentence. However, we know that in this particular scenario, only the verbs or the actions of the subject should be used to classify the data. This is because noun/subject/object do not contribute to the classification. The word problems make use of words like "John", "Mary", "apples", "oranges", etc to personify the underlying equation which is to be solved. If the training set uses name "John" for all the positive dataset class and "Mary" for negative, then the classifier will probably classify the test problem as positive if it has "John" in its question irrespective of the nature of question.
To solve this problem, we added filtering step to our model 1. This filtering step will use the POS tagging to extract the verbs from each problem in the training dataset. We train our classifier based on the verbs and the sequence of verbs(1-3 gram) to classify into positive or negative.
For example,

"Jack had 10 apples and gave 5 apples to Jill. How many apples did Jack have now?"(Negative)

Filtering step will give --> *'had', 'gave', 'had, gave' : Negative*
Rest of the algorithm remains same as Model 1.

Model 2 - Variant of Model 1


Positive Problems
Negative Problems
Extract Verbs using POS Tagging
Logistic Regression Classification
Classifier
Test problem
Classifier
Positive
Use POS tagging to Extract and add the cardinals
Negative
Use POS Tagging to extract and subtract the cardinals

Limitation:

The approach used in Model 1 and Model 2 performs sufficiently good for basic problems. However, it fails majorly because of using cardinals which are nothing but garbage for solution. Secondly, it doesn't support two different operations in one problem (Discussed in Model 3).

Model 3 - A brute force approach:

Consider the following example:

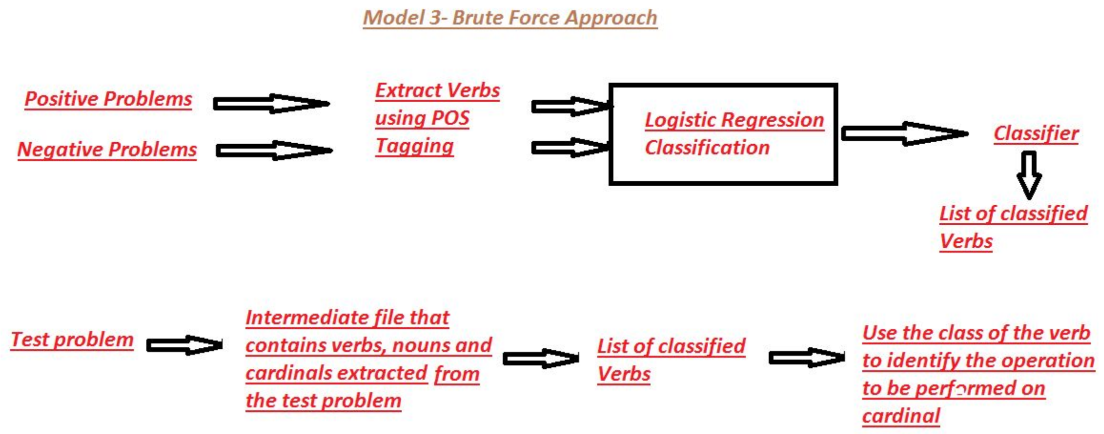
"Jack had 3 pencils. Jill gave 1 pencil to Jack. Jack gave 2 pencils to Mary. How many pencils did Jack have now?" (Positive)

In this example, the above approach will give result in incorrect output. This is because, the model is not smart enough to handle to different operations in one single problem. It will either add or subtract the cardinals but not both. To improve accuracy, we now try to make a well informed model but with extremely simple brute force approach.

We use the filtering step of Model 2 to train classifier.

Pseudo code:

```
1. Use the POS tagging to extract verbs from the training set.
2. Train classifier to classify verbs into three classes :
    a. Assign
    b. Transfer Negative
    c. Transfer Positive
3. Generate a list of verbs and their class.
4. Generate a intermediate file for test data. Extract noun, verb, cardinal from each test sentence and store it in file.
5. Using the list of verbs generated in step 3, classify the POS tags. Based on this classification, perform operation on
   cardinals.
6. Output the result.
```


Model 3- Brute Force Approach
Positive Problems
Negative Problems
Extract Verbs using POS Tagging
Logistic Regression Classification
Classifier
List of classified Verbs
Test problem
Intermediate file that contains verbs, nouns and cardinals extracted from the test problem
List of classified Verbs
Use the class of the verb to identify the operation to be performed on cardinal

Example:

Training : ***"John had 3 oranges. His dad gave him 3 oranges. How many oranges did he have now."***

"Rahul cut 3 trees. Now 3 trees are left. How many trees were there earlier?"

Verb classification will be as follows:

"John had 3 oranges" **===> had ===> assign** || ***"His dad gave him 3 oranges"*** **===> gave ===> transfer positive**

"*Rahul cut 3 trees.* " ===> cut===> transfer negative || ***" Now 3 trees are left"*** **===> are===> final**

Test :

"Jessica had 10 pencils and she gave 5 to Rahul. How many does she have now"

===> [('had', 'VBD', -7), ('10', 'CD', 10.0), ('pencils', 'NNS', '-5'), ('gave', 'VBD', -7), ('5', 'CD', 5.0), ('does', 'VBZ', -7), ('have', 'VBZ', -7)]

===> Since 'had' is classified as 'assign' and 'gave' is classified as 'transfer negative', it will do ===> 10 +(-5) = 5

Limitation:

Only works for problems with two sentences(3 cardinals).

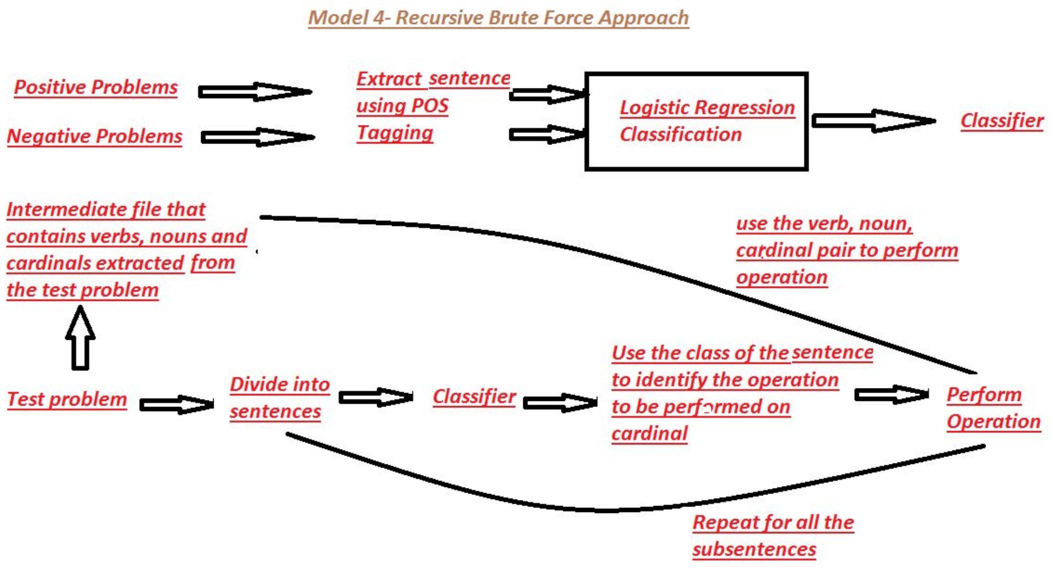
Gives poor performance.

Model 4 - Recursive brute force approach:

As previous model is a brute force model, it will fail for those question whose verbs are totally new to the classifier. Also, it expects every verb to be either of the 4 verb categories - assignment, transfer positive, transfer negative or final.
Thus, this model will consider the entire statement creating a trigram feature vector while training for the above 4 categories. Then, based on the test input sentence, it will find which category this sentence belong and create its corresponding equation form.

Pseudo Code:

```
1. Use the entire sentence tokenized into 3gram vectorizer output as the training set.
2. Train classifier to classify verbs into four classes :
    a. Assign
    b. Transfer Negative
    c. Transfer Positive
    d. Final
3. Generate a pickle file 'trained_model.pkl' for this trained verb categories from input sentences.
4. Based on the test questions, split question in each sentence, and create an intermediate file 'data_vector1' using the
   trained_model.pkl. This intermediate file basically represents the equation forms in tuples.
5. Solve this intermediate file 'data_vector1' using the predict.py script.
6. In predict.py script variables are assigned according to prediction present in the intermediate file. This assignment is
   supported with multiple values for a particular variable which imitates behavior of recursive operations.
7. Output the result.
```


Model 4- Recursive Brute Force Approach
Positive Problems
Negative Problems
Extract sentence using POS Tagging
Logistic Regression Classification
Classifier
Intermediate file that contains verbs, nouns and cardinals extracted from the test problem
use the verb, noun, cardinal pair to perform operation
Test problem
Divide into sentences
Classifier
Use the class of the sentence to identify the operation to be performed on cardinal
Perform Operation
Repeat for all the subsentences

Example:
Training : ***"John had 3 oranges. His dad gave him 3 oranges. How many oranges did he have now."***
"Rahul cut 3 trees. Now 3 trees are left. How many trees were there earlier?"

Verb classification will be as follows:

"John had 3 oranges" **=> feature vector => assign || *"His dad gave him 3 oranges"* => feature vector => transfer positive**
"*Rahul cut 3 trees.* " => feature vector => transfer negative || *" Now 3 trees are left"* => feature vector => final

Test :
"Jessica had 10 pencils. She gave 5 to Rahul. She gave 2 to Jen. How many does she have now"
===> [('had', 'VBD', assign), ('10', 'CD', 10.0), ('pencils', 'NNS', '-5'), ('gave', 'VBD', transfer negative), ('5', 'CD', 5.0), ('gave', 'VBD', transfer negative), ('2', 'CD', 2.0), ('does', 'VBZ', assign), ('have', 'VBZ', assign)]
===> 'had' is predicted as 'assign' and 'gave' is predicted as 'transfer negative' by classifier. Since 'had' is classified as 'assign' and 'gave' is classified as 'transfer negative', it will do ===> 10 +(-5) +(-2) = 3
"John cut 1 apple to eat. Now he is left with 3 . How many apples were there"
===> ['cut', 'VBZ', transfer negative], ('1', 'CD', 1.0), ('apple', 'NN', -5), ('eat', 'VBZ', assign), ('is', 'VBZ', final), ('apples', 'NNS', -5), ('3', 'CD', 3.0), ('were', 'VBZ', -7)]
===> 'cut' is classified as 'transfer negative' and 'are' is classified as 'final'. Therefore the answer will be ===> 1 + 3 = 4

Limitation :

This model does not support the sentences which contains two cardinal values for a single action.
Miscalculates for sentences which involve two variables with cardinal values but equation is linear variable.

C. Evaluation:

We have used the *dataset*[2] provided by Professor. We then divided the dataset as per the classes required by our models.
The *evaluation measure* that we used is the number of problems the model was able to predict and solve correctly.
Each model has it's own pros and cons. However, the accuracy is more in model 4.
Models 1 - 3 restrict the kind of problems that the system can solve. Therefore, if the test dataset conforms to these restrictions, these models will work fine and will be able to solve almost all the problems if correctly predicted. However, model 4 tries to get rid of these restrictions. A summarized evaluation of the type of questions that each model can solve or not is given in table below. However the accuracy also largely depends on the prediction(i.e. F1 score) of the classifier.

Underlying Equation of the question	Model 1	Model 2 (Better Predictor)	Model 3	Model 4
X= 10 + 20 Single operation, either addition or subtraction	Solves	Solves	Solves	Solves
X = 10 + 20 + 30 Or X = 30 - 20 - 5 More than two operations, but same operator	Solves	Solves	Fails	Solves
X = 5 - 2 + 1 Two different operations in one equation	Fails	Fails	Solves	Solves
X = 5 + 4 + 3 - 10 More than two operations, mixed operators.	Fails	Fails	Fails	Solves
Y = 2 (Doesn't contribute to the result i.e. value of x) X = 10 + 20 - 5 Any type of equation with additional variable and cardinal which doesn't contribute to result.	Fails	Fails	Fails	Fails

Model 1 and Model 2 behave poorly because they simply classify the problems into positive or negative and accordingly add or subtract all the cardinals present in the problem. This prevents the model from solving equations that involve mixed operations.
On the other hand, Model 3 supports at most 2 mixed operations or one addition/subtraction. This is because the model is a brute force approach built on three variables. Model 4 overcomes all these limitations, by recursively applying brute force to allow multiple mixed operations.
However, all four models, have one major drawback and that is they fail to separate important and unnecessary cardinals. The models will give inaccurate answer due to addition or subtraction of extra information(unnecessary cardinal). If the unnecessary cardinal is expressed in words, then this will no more be an issue.

D] Conclusions:

Using the basic understanding of classifiers and NLP POS tagging, we were able to build four models with incremental approach. Although each model suffers from its own limitations, the model 4 can be further improved to increase accuracy. We can also extend these models to support more operations like multiplication and division. We worked with very limited data as our focus was on learning the NLP POS Tagging and its applications in our model. With increased and correctly classified learning sets, we can make our classifiers work even better. The variance and bias are also applicable in this scenario but can be handled.

E. References:

1]http://textminingonline.com/dive-into-nltk-part-iii-part-of-speech-tagging-and-pos-tagger
2]http://ssli.ee.washington.edu/~hannaneh/algebra-emnlp14.pdf
3]https://www.cs.washington.edu/nlp/arithmetic

[2] http://ai2-website.s3.amazonaws.com/data/arithmeticquestions.pdf