International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

NLP based Mathematical Problem Solving

Sonal Srivastava[1], Shreya Sharma[2], Sonia Sharma[3], Manika Bhardwaj[4]

[1]*Sonal Srivastava, Dept. of Information Technology, ABESEC, UP, India*
[2]*Shreya Sharma, Dept. of Information Technology, ABESEC, UP, India*
[3]*Sonia Sharma, Dept. of Information Technology, ABESEC, UP, India*
[4]*Manika Bhardwaj, Dept. of Information Technology, ABESEC, UP, India*

Abstract - *A computer program can easily compute solutions to an equation, but it cannot understand the same problem given in the form of a word problem where the computer needs to identify which equation is to be put and also identify what information is provided in the problem text. Here, we describe a NLP based approach with which a computer can be trained to identify the topic and subtopic and ultimately the equation that is applicable to the solution of the given problem and also identify the known values in the problem. Only a handful of utility software is available today providing this functionality. The ones online provide solutions for template-based problems. This study goes beyond currently used template-based models and uses Parsing Technique for identifying the given information in the paragraph and focuses on TF-IDF measure with SVM classifier for topic classification. The current implementation of the system considers only high school level problems of Physics.*

1. INTRODUCTION

Humans are no match for a computer in terms of accuracy and speed of mathematical computations, given that the computer is provided with the mathematical problem in a strict syntax of a computer programming language. However, this accuracy and speed vanish as we drift from a computer understandable syntax to a human-understandable natural language. Machines are not well equipped with understanding problems the way humans do in a natural language. There has been a lot of research in this area to devise ways in which this can be done successfully.

There have been basically two approaches that have been widely experimented with: the symbolic method and the statistical way. Most previous symbolic approaches suffer from a major shortcoming: most of the work revolves around ad-hoc rules and pattern matching that for the small percentage of work with evaluation results available, makes it unclear whether the patterns and rules are specially designed for specific sentences in the test set.

In this paper, we present a system that automatically solves mathematical word problems using rule-based Natural Language Processing. These rules leverage the commonly followed patterns and the positioning of information in such word problems and use them to the benefit of solving them. It identifies information from the word problem regarding the known values and the unknown values. It also classifies the topic of the problem using TF-IDF and SVM algorithms, narrowing down the search for the suitable equation that eventually solves the problem.

2. RELATED WORK

Various efforts have been made for automatically solving word problems. Early researches were limited to a fixed set and type of word problems. Later, solvers became capable of solving a wide range of problems.

In 1964, a system named STUDENT was developed by [1] which was capable of solving algebra word problems of specific structures like time, rate, sequence, percentage, etc. It was the first remarkable AI approach for solving word problems. After this, in the 1980s, [2] proposed CHIPS (Concrete Human-like Inferential Problem Solver) which was based on child psychology for solving word problems.

Then, the technological progress in the field of natural language processing, machine learning and artificial intelligence marked a new beginning with the development of complex problem solvers. [3] developed a system called ROBUST which could understand word problems in an independent manner and in multiple steps with some extra information, based on propositional logic. At the same time, [4] proposed a method to categorize problem text

into two categories-"Multiplicative Compare(MC)" and "Equal Group(EG)" using classification techniques based on Part of Speech Tagging (POS).

In recent years, the first and pioneer work was done by [5] using a statistical learning approach. The system was made to learn word problems with preset equation templates and equations. These were further processed to generate answers. This research was related to the following three fields: semantic interpretation, information extraction and automatic word problem-solving.

In 2015, [6] presented a semantic parsing and reasoning approach to automatically solve mathematical word problems. They designed a new meaning representation language DOL for semantic representation of natural language text. The natural language text was represented as a semantic tree in DOL. A reasoning module was then used for deriving mathematical expressions from DOL trees and calculating problem answers by solving equation systems.

3. APPROACH

To automatically solve a problem, the system needs to figure out, somehow, the following,

1) what are the values already given in the problem
2) which is the equation that would solve the problem
3) what is that certain value of that certain type which is demanded,

In our system, all possible keywords associated with Physics Class IX NCERT problems are pre-stored as a bag of words using training data. Also, standard units of measurement are stored and mapped to the keywords. We identify and store the equations related to the topics of Laws of Motion, Force, Gravitation, Work and Energy and Sound, up to the level of high school. We also map the specific variables to these keywords that are utilized in corresponding equations. The numerical values are identified and they are associated with the statistically nearest found keyword in the sentence given that it is separated by words like 'is', 'of'. For example, "Ramu runs at the speed of 10 kmph from his apartment to the school every day. The distance between the two places being 20 km, find the time he takes for the commute." The keywords are speed, distance and time. The nearest numerical value associated with each is *'10'* for speed and *'20'* for distance. Similarly, the unit of the numerical value always occurs after the numeric value itself, eg. '*10'* is followed by *'kmph'* and *'20'* by *'km'*. In case there is no obvious keyword in the vicinity of the numeric data, we determine the type of the numeric data using its unit specified in the problem since every variable in the physical sciences has a set of units of measurement associated with it. One of the units is given the status of the standard unit of the variable. This is used when there is no specific unit in which the value of the variable unknown is demanded.

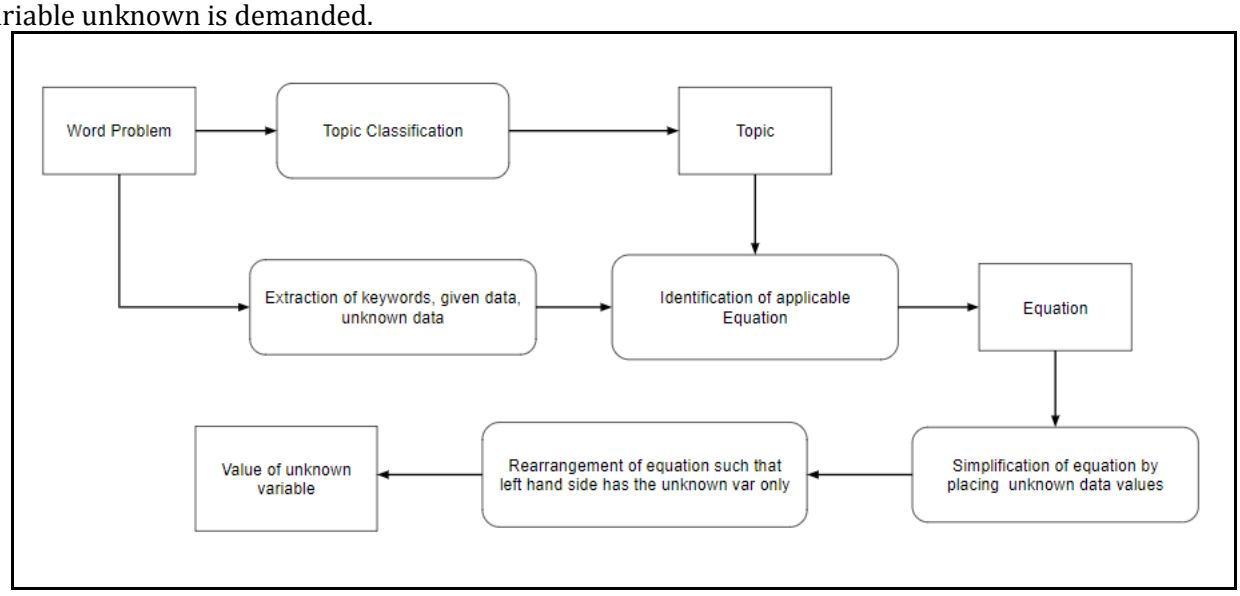

Fig. high-level flow of processes

Next, we identify the missing variable in the word problem (what is being asked to determine). The missing quantity is usually preceded by phrases containing words like 'find', 'what', 'determine', 'convert', 'when', etc. We narrow down the search of equations from our collection to a particular topic using Topic Modelling algorithms like TF-IDF. Then, we map the collection of all the known and unknown keywords and look for the equation corresponding to the topic identified for which the derived information fits in completely. We feed in the information about the known values and the unknown values and the equation found suitable into the Mathematical Equation Solver System.

3.1. Equation Solver Module

This system converts the infix equation into a more computer-friendly postfix equation according to the precedence and associativity of the occurring operators in the equation. The known values obtained from the processing of the problem text are put in their corresponding place and the obtained expression is further simplified. This simplified expression happens to have only a single unknown variable with the whole expression equating to zero.
The simplified expression is parsed and operations on the variable are reversed so that an addition operation on it when taken to the right side of the equation becomes a subtraction, a multiplication operation becomes a division operation, and so on. And hence the value of the unknown variable is arrived at.

3.2. Topic Classification Module

The process of topic classification involves these three steps: web scraping- to form the dataset with word problems of each topic, calculating TF-IDF for collected data and using SVM classifier to classify the topic of the problem entered by the user.

Web Scraping is done using Python's BeautifulSoup library to form a dataset containing problems and their corresponding topics. Relevant data is filtered out of the data extracted from the web and is stored in a csv file.

Next, for multi-class classification of the entered word problem into one of the topics, data from the csv file is transformed into a TF-IDF weighted vector. The reason behind using TF-IDF instead of raw frequencies of the token occurrences is to alter the impact of frequently occurring tokens in a certain topic and which are as a consequence, less informative than topic features.

Term Frequency (TF) measures how frequently a term occurs in a topic. Every topic has its own term frequency. Inverse Document Frequency (IDF) measures the importance of a term to the document. The IDF is computed once for all the topics in the document.

$$TF(t) = \frac{(Number\ of\ times\ term\ t\ appears\ in\ a\ topic)}{(Total\ number\ of\ terms\ in\ the\ topic)}$$

$$IDF(t) = log(\frac{(Total\ number\ of\ topics)}{(Number\ of\ topics\ with\ term\ t\ in\ it)})$$

Lastly, TF-IDF is calculated as TF multiplied by IDF. Python's sklearn library provides CountVectorizer and TfidfTransformer functions to automate these calculations. CountVectorizer converts the data into a matrix of token counts and then TfidfTransformer transforms this count matrix to a normalized TF-IDF representation.
Finally, an SVM (Support Vector Machine) classifier is used on the TF-IDF matrix and we get the topic corresponding to the problem entered by the user.

4. RESULT

We examined our system on several types and patterns of high school physics problems. The accuracy and precision of the result depended on the individual accuracy of the two modules:

4.1. Topic Classification Module

Size of dataset: 580 problems approx.

	Precision	Recall	F1-score	Support
Motion	0.89	0.81	0.85	31
Force and laws of motion	0.90	0.67	0.77	27
Gravitation	0.86	0.96	0.91	46
Work and Energy	0.88	0.95	0.92	40
Sound	0.92	0.96	0.94	49
Accuracy	-	-	0.89	193
Macro average	0.89	0.87	0.88	193
Weighted average	0.89	0.89	0.89	193

4.2. Equation Solver Module

This module is a pure mathematical component of the system. The module works correctly on all the instances tested from a set of 50 or so equations representing different operators of different precedences, no matter the position of the unknown.

4.3. Overall System

The system gives correct results for 78% of the problems we have tested. That being said bogus or incoherent text is incomprehensible to the system as is to a human. The system is however tolerant to spelling mistakes. It also conveniently adapts to the many ways a single unit of measurement may be denoted by, such as, *'kilometre per hour'* as *'km/h'*, *'km/hr'*, *'kmph'*, etc. The system expects a problem that demands only one numerical solution. A problem with more than one value asked results in only one of the values returned or error.

5. CONCLUSION

In this paper, we propose an approach to solving word problems of mathematical nature found in high school students textbooks. We have designed language processing rules that leverage the specific format used in Physics numerical word problems. TF-IDF measure and the SVM classifier are used to classify the topic of the problem. The appropriate equation is put in to arrive at the solution desired. We have achieved success in solving simple high school level problems as of now. We hope to extend our technique to handling more general math word problems and to other domains in the future.

6. REFERENCES

1. Bobrow, D.G.: Natural language input for a computer problem-solving system (1964)
2. Briars, D.J., Larkin, J.H.: An integrated model of skill in solving elementary word problems. Cognition and instruction (1984)
3. Bakman, Y.: Robust understanding of word problems with extraneous information. arXiv preprint math/0701393 (2007)
4. Centintas, S., Si, L., Xin, Y.P., Zhang, D., Park, J.Y.: Automatic text categorization of mathematical word problems. In: FLAIRS Conference (2009)
5. Kushman, N., Zettlemoyer, L., Barzilay, R., Artzi, Y.: Learning to automatically solve algebra word problems (2014)
6. Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, Yong Rui: Automatically solving number word problems by semantic parsing and reasoning (2015)