CS 447 Final Project Proposal: Universal Algebraic Word Problem Solver

Liming Wang
lwang114@illinois.edu

Xinran Wang
xranw5@illinois.edu

1 Introduction

Math word problems (MWP) are math problems written in natural language. In order to solve a math word problem, machine needs to be able to detect quantities in the word description and the implicit and explicit relations between them. A machine capable of solving math word problem can serve as virtual math tutor for high school students and online computational engine to solve problems from text queries. Further, the same insights gained from building such a machine can be transferred to more advanced systems for theorem proving, scientific question answering and even automatic math modeling. Our project aims to develop an NLP algorithm capable of solving both arithmetic and algebraic word problems in a unified framework.

2 Related Work

Mathematically, machine needs to translate a problem $P(\mathbf{w}, \mathbf{q})$ described in word strings $w_1...w_{N_1}q_1...q_{N_2}$, where w_i's are the conditions of the problem and q_j's form the question, into mathematical expressions in mathematical symbols like arithmetic operators and numbers $x_1...x_M$. Clearly, some word problems may not be solvable and some may be ambiguous and can lead to more than one mathematical expressions. Further, some problems may involve quantities irrelevant to the question of the problem. Even if the problem is assumed to be solvable and no irrelevant information is provided, the correct translation of the problem requires common sense knowledge about the entities and relations described by noun phrases and verb phrases of P as well as domain-specific knowledge such as the mathematical theory behind the problem. To simplify the problem, current works focused on two types of MWPs: arithmetic word problems, whose solution involves only known variables and algebraic word problems, which requires solving a equations of unknowns.

The idea of solving MWP automatically may trace back to the 1960s, when various approach based on symbolic rules has been developed (**?**), (**?**). Most recent approaches, however, are mostly statistical. (**?**) consider only problems that involve addition and subtraction and used various transition-based approach to model various verbs and nouns and transform them into quantities and equations. One way to simplify the general MWP is to restrict the types of relations in the word description. For example, Roy and Roth (**?**) (**?**) consider a very special type of arithmetic problem that only contains number and verbs for arithmetic operations like "multiply" and "divide". Then they hand-designed a CFG to parse the sentences into "expression tree", and evaluate the corresponding arithmetic expression represented by the tree. However, this may be too restrictive and wash away many key challenges in MWPs, so they train a few classifiers to help determine the relations between the quantities from more diverse problems that requires common sense knowledge. (**?**) solve the more general algebraic word problem by considering a finite number of "templates" for the problems and the expression for P will then depend only on its template and the specific quantities that go into the template.

Our approach will be either rule-based or template-based, or both. But the basic module will consist of an event detection module that detects the entities, head verbs and quantities. Then the rule-based system will use a set of grammar to map the event to mathematical expression; the retrieval-based system will retrieve the most probable equation template from a set of templates, and then fill the template with quantities. We will not implement the downstream part of the event detec-

tion, such as dependency parsing or name entity recognition, but implement the feature extraction based on these systems; we are likely to design our own rules and/or implement the retrieval system ourselves.

3 Dataset and Resources

Roy and Subhro (**?**) publicizes their data Illinois dataset and commoncore dataset. The former contains only single-step arithmetic problems. After performing pruning by removing some problems with the same templates to keep the data balanced across templates, the dataset contains 562 problems. The latter contains multi-step arithmetic problems and has 600 problems in total. All the problems in the latter do not contain irrelevant information. Kushman et. al. (**?**) released their dataset which contains 514 algebraic problems, and each problems contain around three sentences.

Further, Kedziorski et. al. (**?**) has developed MAWP, a math word problem platform for training and evaluating MWP systems. Conveniently, It encompasses all the datasets mentioned above. There are five categories of questions: "AddSub" with only addition and subtraction, "SingleOp" with single arithmetic operation, "MultiArith" with multiple operations, "SingleEq" with a single equation, "SimulEq-S" with single or two equations and "SimulEq-L", the superset of "SimulEq-S" with multiple equations. All questions are in one single .json file. Each problem entry of the .json file contains the index, the equation, the solution and the word description. Our project will use this platform as our dataset and testbed.

All works mentioned above released their codes. Although our project will be based on the method proposed in (**?**), we will implement new learning models from scratch using deep learning frameworks like Tensorflow or Pytorch to reimplement their work if necessary.

To build upon their models, some other pretrained models in the standard NLP pipeline may be used for purposes like detecting entities, resolving coreference, finding synonyms. To this end, our models may use toolkits like NLTK or Stanford CoreNLP as building blocks.

4 Tentative Timeline

Nov. 4th (2 weeks) — understand all the papers and write the related work part of the final report; decide our design, work on the baseline
Nov. 9th — Finish the intermediate project report
Nov. 18th (2 weeks) — Finish implementing the baseline model
Dec. 2nd (2 weeks) — Potential improvement on the model; finish writing the baseline results part of the final report
Dec. 7th (1 weeks) — Finish the final report