Final Project Checkpoint
Formalized Algebra from Text

Adam Wiemerslage and Rehan Ahmed

April 14, 2018

1 Abstract

In this project, we are proposing to solve algebraic word problems in a way similar to Kushman et al (Kushman et al., 2014). However, rather than relying on a rule based approach (Mukherjee and Garain, 2008), or a mapping function to a set of templates (as in Kushman et al), we are proposing a slightly different method. The method that we are proposing is much closer to the solution in (Hosseini et al., 2014). In this method, we will learn a mapping from the input sequence of words, to the set of verbs and their syntactic dependents, that include some type of quantifier, via a dependency parse of the text. We will then categorize the verb expression to be equivalent to some command, expression, or set of commands and expressions in the semantics of a programming language that we define for solving basic algebraic problems.

The categorization task will be based on a semantic representation in a vector space model, using fasttext (Bojanowski et al., 2016). Each command or expression, will then have its arguments filled by one of the syntactic dependents of the verb, which will then map to the arguments in the described in the semantics of our target language. The categorization parameters are learned over the same data that (Hosseini et al., 2014) use, and will be assessed in order to compare results directly. The target language will be the simple **IMP-1** language.

2 Architecture

The general workflow of our system is pictured in figure 1

3 Example

Example 1. *Pooja has 3 apples. She eats one apple. How many apples does Pooja have now?*

We attempt to translate the sample problem's text into the following commands:

Pooja has 3 apples	$->$	$a_p := 3$
She eats one apple	$->$	$a_p := a_p - 1$
How many apples does Pooja have now?	$->$	$\texttt{print}\ a_p$

And since these are sequence of steps we end up with:

$$a_p := 3; a_p := a_p - 1; \texttt{print}\ a_p$$

More formally:

$$\texttt{seq}(\texttt{set}[a_p](3); \texttt{seq}(\texttt{set}[a_p](\texttt{minus}(a_p; 1)); \texttt{print}(a_p)))$$

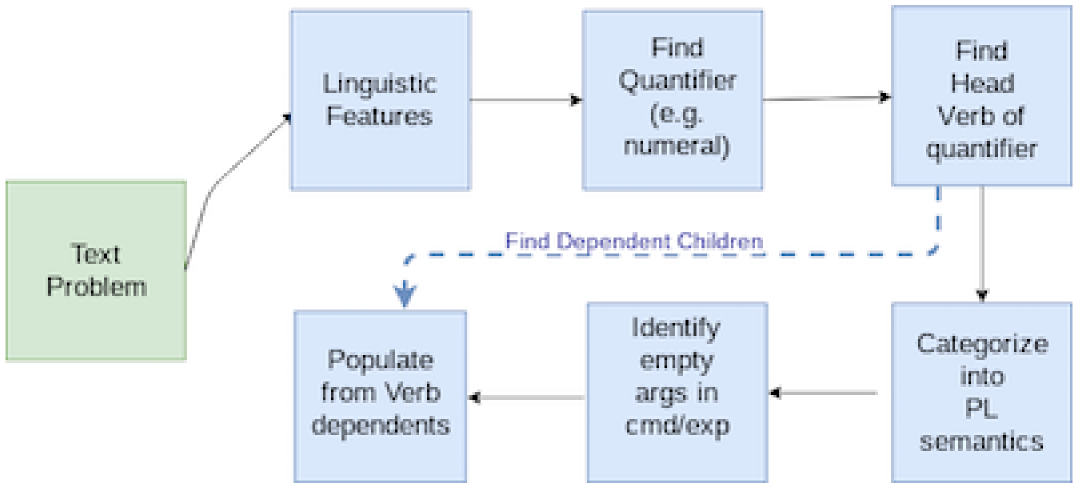

Figure 1: High level overview of the architecture

4 Language: IMP-1

$\boxed{e : \tau}$

$$\frac{}{\texttt{addr}[a] : \mathsf{float}} \qquad \frac{}{\mathsf{float}[\mathsf{f}] : \mathsf{float}} \qquad \frac{e_1 : \mathsf{float} \qquad e_2 : \mathsf{float}}{\texttt{plus}(e_1; e_2) : \mathsf{float}} \qquad \frac{e_1 : \mathsf{float}\ e_2 : \mathsf{float}}{\texttt{minus}(e_1; e_2) : \mathsf{float}}$$

$$\frac{e_1 : \mathsf{float}\ e_2 : \mathsf{float}}{\texttt{times}(e_1; e_2) : \mathsf{float}} \qquad \frac{e_1 : \mathsf{float}\ e_2 : \mathsf{float}}{\texttt{divide}(e_1; e_2) : \mathsf{float}}$$

$\boxed{c : ok}$

$$\frac{e : \mathsf{float}}{\texttt{set}[a](e)\ \mathsf{ok}} \qquad \frac{}{\texttt{skip}\ \mathsf{ok}} \qquad \frac{c_1\ \mathsf{ok} \quad c_2\ \mathsf{ok}}{\texttt{seq}(c_1; c_2)\ \mathsf{ok}} \qquad \frac{e : \mathsf{float}}{\texttt{print}(e)\ \mathsf{ok}}$$

Syntax chart:

Typ	τ	$::=$	`num`	`num`	numbers
			`bool`	`bool`	booleans
Exp	e	$::=$	$\texttt{addr}[a]$	a	addresses (or “assignables”)
			$\texttt{num}[n]$	n	numeral
			$\texttt{bool}[b]$	b	boolean
			$\texttt{plus}(e_1;e_2)$	$e_1 + e_2$	addition
			$\texttt{times}(e_1;e_2)$	$e_1 * e_2$	multiplication
			$\texttt{eq}(e_1;e_2)$	$e_1 == e_2$	equal
			$\texttt{le}(e_1;e_2)$	$e_1 <= e_2$	less-than-or-equal
			$\texttt{not}(e_1)$	$!e_1$	negation
			$\texttt{and}(e_1;e_2)$	$e_1 \mathbin{\&\&} e_2$	conjunction
			$\texttt{or}(e_1;e_2)$	$e_1 \mathbin{\|\|} e_2$	disjunction
Cmd	c	$::=$	$\texttt{set}[a](e)$	$a := e$	assignment
			`skip`	`skip`	skip
			$\texttt{seq}(c_1;c_2)$	$c_1;\, c_2$	sequencing
			$\texttt{if}(e;c_1;c_2)$	$\texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2$	conditional
			$\texttt{while}(e;c_1)$	$\texttt{while}\ e\ \texttt{do}\ c_1$	looping
			$\texttt{print}(e)$	$\texttt{print}\ e$	printing
Addr	a				

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

M. J. Hosseini, H. Hajishirzi, O. Etzioni, , and N. Kushman. Learning to solve arithmetic word problems with verb categorization. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, 2014.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. Learning to automatically solve algebra word problems. In *the Annual Meeting of the Association for Computational Linguistics*, 2014.

Anirban Mukherjee and Utpal Garain. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29, 2008.