CS221 Prroject Report

Melvin Johnson Premkumar
Dept. of Computer Science
Stanford University
Email: melvinj@stanford.edu

Isaac Caswell
Dept. of Computer Science
Stanford University
Email: icaswell@stanford.edu

Zane Silver
Dept. of Electrical Engineering
Stanford University
Email: zsilver@stanford.edu

***Abstract*—In which we discuss various things**

I. Introduction

Our project is to detect paraphrases in Twitter, as part of the SemEval challenge for 2015. Given two sentences (Tweets), we determine whether they express the same or very similar meaning and optionally a degree score between 0 and 1. The training dataset is provided by semEval and contains about 17,790 annotated sentence pairs, and comes with tokenization, part-of-speech and named entity tags. The testing dataset consists of a further 1k examples from a different time period, annotated by an expert. SemEval also provides several baselines against which to compare our algorithm's performance. In this progress report we report the result from using Google Wordvec and Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection for this task; and discuss how to improve upon them.

We are working with Xiao Cheng, a PhD candidate in the Computer Science Dept.

II. Related Work

Extracting Lexically Divergent Paraphrases from Twitter; Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan and Yangfeng Ji

Detail the baseline paper (Isaac)

III. Model Description

We are using the MULTIP (Multi-Instance Learning Paraphrase) model for the Twitter phrase classier. This model was presented in the research paper titled "Extracting Lexically Divergent Paraphrases from Twitter". We will be implementing this model and extending its functionality (time permitting). The model description below is a summary of the paper that presents the MULTIP model.

The advantage of using the MULTIP model is that it relies on sentence level relations using a feature-based classifier. Extensive research has been undertaken to improve upon the MULTIP model to that it can correctly identify two related twitter texts. This has proven to be especially difficult because of the amount of variability in the type of tweets and the prevelence of generalized naming for entities. Specifically, the paper presenting MULTIP uses the example sentences:

- That boy Brook Lopez with a deep 3
- brook Lopez hit a 3 and i missed it

The above example is just to illustrate difficulty that arises when the named entities are generalized into different words.

Firstly, the MULTIP model relies on the at-least-one-anchor assumption. The at-least-one-anchor assumption is derived from the idea that twitter messages posted around the same time and the same topic share lexical paraphrases. This intuition allows us to extend the idea further: that most related twitter, not just those posted around the same time or location, messages will share a lexical paraphrase. The lexical paraphrases may be the same words, or different words, that are contained in two different tweets that identity the tweets are being related. In other words, it is assumed that related tweets contain an anchor, a lexical paraphrase.

The MULTIP model setups a learner that observes the labels on groups of sentence-level paraphrases. This due to the at-least-one-anchor assumption described early. This method contrasts with a learner that observes word pairs. There are now two layers in the model since the sentence-level analysis inherently relies on the word-level analysis. This two-level hierarchy is described in explicit detail in the "Extracting Lexically Divergent..." paper. In the context of this class, the two-level model is simply a factor graph with paraphrases/sentences as the upper nodes and the word-pairs as the lower, or leaf, nodes.

For each pair of sentences there is a binary variable ('y') that represents whether the two sentences are related. This is determined to be true (y=1) if there is at least one anchor found in the two sentences. We determine if at least one anchor exists in the set of word-pairs between the two sentences. The set of word-pairs is the set of unique word-pairs that can be formed from one word in each sentence. For each word-pair there is another binary variable ('z') which determines if those words are an anchor or not. Therefore, for y=1 there must be at least one z=1 in the set of word-pairs for the two sentences.

ZANE

For two sentences s_1 and s_2, y(s_2,s_1) = 1 if z_j = 1 for at least one j. z_j = 1 only if w_j = (w_{j1}, w_{j2}) is an anchor, where w_{ji} corresponds to a word in sentence s_i. For every sentence pair (s_i, s_k), there will be $|s_i|$ * $|s_k|$ word-pairs, where $|s_i|$ is the number of words in sentence s_i.

The next step now is to properly determine when two words are an anchor. This aspect will be incorporated into our different learning algorithms. WordVec has proven to be the most promising method to properly determine whether or not two words, and by extension two phrases, are lexically related.

==== describe model here ===

IV. Initial Steps: Wordvec

- brief discussion of what this is
- how did it work? (1-2 pg)
- not so great - get numbers TODO
- discussion of why

V. Current Thread of Work: Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection

- What is the algorithm?
- similar to wordvec, but extends in in these ways.
- Why is it well suited to our problem specifically?
- how did it work? (TODO: run this)

VI. Next Steps

There are several things we can do to improve the accuracy of our current model. Following is a list of the next things we will work on.

A. better normalization of the data

Twitter datasets tend to attract slang and misspellings, arbitrary capitalization and the like. An easy way to improve our model would be therefore be to normalize the data better first. We have found an external library (twitter lexicon normalization) which implements X, Y, by means of Z; we will apply this to our dataset and see what the effect on our performance is. This tool can be used with a Twitter specifc entity detector, availiable online at https://github.com/sem-io/python-twitter-spell-checking. Furthermore one could experiment with basic word stemming.

B. Expand on Xiao's comment: Open problems: better ways to identify these phrases in the sentences from parse trees? how to learn the representation of these compositional phrases without retraining all the other word vectors?

VII. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.