Solving Hard Coreference Problems with Word Vectors

David Lu, 6.806

(Code available from https://github.mit.edu/uldivad9/David-Lu-6.806/tree/master/non-code)

Introduction

Coreference is an extremely important problem of Natural Language Processing involving disambiguating the subject of some kind of indeterminate reference (such as a pronoun). For example, determining who 'he' is in the sentence "*Mary lent Joe a book that he hadn't read*" is a problem of coreference. Note that this coreference problem is relatively easy to solve – we know that 'Mary' is a female name, and we know that 'he' must resolve to a male person. However, harder coreference problems often cannot be solved using these syntactic rules. For example, in this sentence:

Ex: *Despite its small size, the ant managed to drag the leaf.*

The coreference in this sentence cannot be resolved without knowledge of the world (what is 'size', what is 'small') and inductive ability ('despite' indicates that we are somewhat surprised about the contents of the sentence – thus it makes more sense that 'the ant' is being described as small). Humans perform extraordinarily well on these kinds of coreference problems, but current natural language systems are far behind.

In this paper, we outline an approach to solving difficult coreference problems using the semantic information contained in word vector embeddings. We show that word vector information alone is enough to resolve a significant number of coreference problems, and compare a word-vector-only approach against several state-of-the-art coreference resolution systems.

Dataset

The Winograd dataset[1], gathered by Rahman and Ng (2012), consists of 1886 sentences with instances of pronoun resolution that cannot be solved with syntactical rules. A single data point comes in the following four-line format:

```
The bee landed on the flower because it had pollen.
it
The bee,the flower
the flower
```

The first line consists of the sentence. The second line states the pronoun to be resolved. The third line lists the two *candidate entities* that the pronoun might refer to, and the fourth line lists the correct candidate entity.

[1] Available from http://www.hlt.utdallas.edu/~vince/data/emnlp12/

Each of the data points has exactly two candidate entities, which are both listed. Thus a natural baseline method would be to randomly select between the two candidate entities, which gives an accuracy of 50%.

Word Vectors

Word vectors are a frequently-used tool in natural language processing, where words are represented as high-dimensional vectors based on co-occurrence. Word vectors also have some well-known and interesting properties – for example, analogous pairs of words (man/woman versus king/queen) have similar word vector differences.

The main tool we use in this paper is *word vector cosine similarity*. This is roughly a measure of semantic relatedness, such that words that are in some way related to each other have a higher similarity.

Throughout the entirety of our project, we used a pretrained, 300-dimension word vector model based on a Google News corpus[2]. We utilized the Python gensim package[3] with this model.

Intuition

Let us consider the following sentence from the Winograd dataset:

Ex: *The pen is mightier than the sword because it can only stab things.*

Here we have a coreference problem where 'it' refers to one of the two candidate entities 'the pen' or 'the sword'. One of the clues that helps a human solve this problem is the fact that swords are more associated with 'stabbing things' than pens are. We can replicate this kind of connection using word vector similarity! Consider if we pulled out the keywords 'pen' and 'sword', along with the keywords 'stab' and 'things'. Then, we compare the 'relatedness' of the keyword 'pen' to the keywords 'stab' and 'things', getting some kind of overall relatedness score. Next, we do the same thing between the keyword 'sword' and the keywords 'stab' and 'things' for a second score. Intuitively, the relatedness score of the 'sword' would be higher, and on that basis we could decide that 'it' referred to 'the sword'. We will use this method (more explanation of exactly how keywords are determined and relatedness is computed will appear later on).

However, this method cannot apply to all sentences:

Ex: *George scored against Thomas in the shootout, so he won the game.*

Here, the two candidate entities 'George' and 'Thomas' have little to no semantic information tied to them. In order to solve this problem, we need to look at the relationship between the entities – namely, the fact that George 'scored against'

[2] Available at https://code.google.com/p/word2vec/

[3] Available at https://radimrehurek.com/gensim/index.html

Thomas. Scoring has a relationship with winning, and so if George is the one who scores, it makes sense that George is the one who wins. Hence, we will take the word 'scored' and consider it a keyword of 'George'. This allows us to represent (albeit crudely) relationships between entities using our keyword-centric approach.

Implementation

Our system takes in as input a datapoint formatted like the Winograd dataset – four lines consisting of sentence, pronoun, candidate entities, correct entity. First off, we part-of-speech tag the words and detect named entities (done through the NLTK toolkit[4]).

We now create one set of keywords for each candidate entity. For each candidate entity, we extract keywords from the text of the entity (i.e. 'the sword') only if the candidate entity was not recognized as a named entity (such as 'George' or 'Thomas'). Assuming the candidate entity is not a name, then we take all words from the candidate entity that are either a noun, an adjective, or a verb, unless the word is in a predefined list of words to ignore. The list of words to ignore consists of tenses of 'to be' along with several other common words that provide no meaningful information.

Next, we consider the described relationship between the two candidate entities defined by the chunk of the sentence between the two candidate entities. We extract the keywords from this chunk just as before. All keywords are added to the keyword set of the candidate entity that comes first in the sentence, as X VERB Y usually means that VERB can be attributed to X. However, if the VERB follows either a negation (not) or a word that indicates passivity (was), then it is added to the keyword set of the second candidate entity.

Finally, we look through the sentence to locate the pronoun to be resolved, and take out the clause or phrase that the pronoun belongs to. For example, in "*The pen is mightier than the sword because it can only stab things.*", this clause would be '*can only stab things*'. We perform the same keyword-extracting operation as before on this clause to produce a set of 'pronoun keywords'.

We now have three sets of keywords – one for candidate entity 1, one for candidate entity 2, and one for the pronoun. We calculate the average similarity between the candidate entity 1 keywords and the pronoun keywords by taking the average word vector similarity of all pairs (word1, word2) where word1 is from the candidate entity 1 keywords and word2 is from the pronoun keywords. This gives us a similarity number between 0 and 1. Similarly, we calculate the average similarity between the candidate entity 2 keywords and the pronoun keywords. Our system chooses the candidate entity whose similarity to the pronoun keywords is the highest, and returns that as the answer to the coreference.

Results

We tested the system on the Winograd test set. The word-vector-only system improved significantly over the baseline.

[4] Available at http://www.nltk.org/

IlliCons[5]	**Word Vectors**	Rahman and Ng[6]	KnowComb[7]
53.26	58.28	73.05	76.41

Table 1: Performance results of the system discussed in this paper as compared to several other coreference systems.

However, it is still far below the performance of the best coreference systems for Winograd-style problems. Of course, this is partially due to the huge difference in complexity between the word vector system described in this paper and the other systems. Both the Rahman and Ng and KnowComb systems are comprised of an ensemble of various statistical and other methods, whereas our system only uses one relatively simple method.

Conclusion

We outlined a method to solve difficult coreference problems using word vectors, and showed its performance against several state-of-the-art coreference systems. Ultimately, word vectors are useful for resolving coreference, but they alone are not enough. The semantic information provided by word vectors is not enough on its own. While word vectors are good for measuring word associations and similarities, we often need to do more than that to resolve coreferences. For example, consider the sentence:

Ex: "*Bob helped Joe because he wanted to help.*"

It is very hard to see how word vector similarities would be able to resolve this sentence. For this sentence, we need a method that recognizes that 'wanting to help' is a good reason to help someone, which is a very different kind of information than what word vectors provide. Indeed, a predicate-based schema like that of Peng et al. (2015) seems more appropriate for this kind of problem.

Nonetheless, our results show that word vectors can contribute significantly towards resolving coreference. We hope to see coreference systems adopting word vectors as one powerful part of an ensemble of methods.

[5] See Chang et al., 2013
[6] See Rahman and Ng, 2012
[7] See Peng et al., 2015

References

K. Chang, R. Samdani, and D. Roth. 2013. A constrained latent variable model for coreference resolution. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 601–612. Association for Computational Linguistics.

A. Rahman and V. Ng. 2012. Resolving complex cases of definite pronouns: the Winograd schema challenge. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 777–789. Association for Computational Linguistics.

H. Peng, D. Khashabi, and D. Roth. 2015. Solving Hard Coreference Problems. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies, pages 809-819. http://cogcomp.cs.illinois.edu/papers/PengKhRo15.pdf