Can we find Synonyms and Antonyms from word embedding analogy?

Sourav Dutta
2576494, Language Science and Technology[*]
Saarland University, Germany
souravd@coli.uni-saarland.de

March 25, 2019

Abstract

In this project, we try to find whether it is possible to retrieve synonyms and antonyms of a given word by utilizing relationships between a pair of word embeddings already provided. Our approach to answer this question lies in establishing a *similarity analogy* between a pair of words to identify how similar the words in that pair are to each other. Using that already established similarity relation, we try to find such a related word for a new input word. However, the main objective here is to understand whether this approach helps us to find synonyms and antonyms for a given word.

Introduction

This report is divided into sections which take us step-by-step through the challenge and our approach where we try to find a solution to it. The *Introduction* section talks about the problem and the resources already available to us for use. We have described our approach to the problem in the *Methodology* section. We show or findings in the *Results* section. In *Conclusion*, we try to summarize our findings and list the possible things to work on after this in the *Future Work* section. The Jupyter Notebook file `CoLi_project_(2576494_Sourav_Dutta).ipynb` contains the complete implementation along with comprehensive comments to explain the code.

Language has always been one of the most important forms of conversation. We are all aware of the fact that characters in a script are merely used to represent the language in a written form. The smallest fundamental unit in any language that carries a meaning of its own is a **word**. Hence in order to understand the meaning of words in a text or speech, we need to process the semantic meaning

[*]This report is submitted as part of the final project for the lecture *Computational Linguistics* offered in the Winter semester (2018-2019) in Saarland University, Germany.

representations conveyed through those words. However, words are nothing but sequences of characters and they cannot be used directly in mathematical computations for any kind of operation. Furthermore, understanding the semantic content hidden behind a text or speech is one of the most challenging tasks in the field of *Natural Language Processing* today.

We try to answer these questions through **word embeddings**[1]. Word embedding is a set of language modeling and feature learning techniques where words or phrases from the vocabulary are mapped to vectors of real numbers. It involves a mathematical embedding from a word string to a continuous vector space with a much lower dimension. There exist different types of word embeddings like Word2Vec (Mikolov et. al. 2013) and GloVe (Pennington et. al. 2014). Here we have used Stanford's **GloVe** embeddings[2] which have been pre-trained with 6 billion tokens, resulting in a vocabulary of 400,000 words with 300 dimensions. For visualizing the embeddings, we have used the **t-SNE** (Maaten et. al. 2008) dimensionality reduction method. We have used the *Cosine Similarity* method in order to find similarity scores between pair of words.

Methodology

Load pre-trained GloVe embeddings

First we load the 300 dimensional pre-trained GloVe embeddings. We parse the embedding file in order to get a complete list of the *vocabulary* and a dictionary of 300-dimensional *word vectors*.

Cosine Similarity

Algorithm We use the *Cosine Similarity*[3] method to find the similarity (cosine value of the angle between the vectors) between the two input words. This relation is mathematically represented as given by equation (1) below.

$$similarity(u, v) = cos(\theta) = \frac{u.v}{||u||_2.||v||_2} \tag{1}$$

Sample results We provide a pair of words as input to this algorithm and find similarity scores between them. The results were quite logical. For instance, while the words *father* and *mother* are closely related (75%), the word *baby* was closer to *mother* (54%) than *father* (39%), which is very likely indeed. Other logically sound relationships were seen in various domains like geographical locations, country capitals, places and related activities. The cosine similarity was also able to return a higher similarity score for synonyms than that of

[1]Word embedding: `https://en.wikipedia.org/wiki/Word_embedding`

[2]GloVe: Global Vectors for Word Representation `https://nlp.stanford.edu/projects/glove/`

[3]Cosine Similarity: `https://en.wikipedia.org/wiki/Cosine_similarity`

antonyms. For instance, *love* was more similar to *affection* (58%) than to *hatred* (32%).

Calculate "Analogy"

We are able to calculate the similarity score between any two words present in our vocabulary. This similarity score is a percentage representation of the vector difference between the word vectors of the two particular words. Now using this relation between the two words as reference, we try to find such a related word for a new input word. The logical representation of our approach is given by the equation (2) below.

$$word\_4 = max\_nearest\_similarity(word\_3 - (word\_2 - word\_1)) \quad (2)$$

Theoretically, this logical conclusion that is drawn by comparing relations is known as **Analogy** or **Identity of Relation**[4]. The above equation (2) refers to this logical relation: *word_ 1 is related to word_ 2, as word_ 3 is related to word_ 4.* In simpler words, we try to find a certain word_4 which is related to word_3 in a similar way as word_2 is related to word_1. The only constraint in this approach is that we need a predefined pair of word_1 and word_2 which can be used as a reference in our algorithm. Analogy has proved to be an interesting relationship measure among words recently (Chen et. al. 2017 and Bouraoui et. al. 2018). As our main goal was to investigate if we are able to get synonyms and antonyms using this approach, we need a pair of synonyms and/or antonyms as reference inputs to our algorithm. For this purpose, we use *WordNet*.

Synonyms-Antonyms from WordNet

WordNet[5] (Miller et. al. 1990) is a large lexical database of English. Words belonging to different parts of speech including nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms called *synsets*, each expressing a distinct concept. Synsets help find conceptual relationships between words such as hypernyms, hyponyms, synonyms, antonyms etc. Here, we have used the free-to-use WordNet corpus that comes with the NLTK python library to get a list of synonyms and antonyms for any input word. The relationship between the pair of words from WordNet is used as a reference to find other words in such a similar relationship.

Examples For a better understanding, let us look at some of the examples of using WordNet to find synonyms and antonyms in Table 1. We can see that WordNet is really efficient in capturing synonyms and antonyms for a word. However, there are exceptions in certain cases, for instance, some nouns like *earthquake* may not have a proper antonym.

[4]Identity of relation: `https://en.wikipedia.org/wiki/Analogy#Identity_of_relation`
[5]More information on WordNet: `http://wordnet.princeton.edu/`

Table 1: Synonyms and antonyms from WordNet

Word	Synonyms	Antonyms
earthquake	temblor, seism, quake, ...	-
talented	talented, gifted	untalented
walk	walkway, pass, paseo, ...	ride
below	downstairs, beneath, under, ...	above, upstairs
quickly	speedily, rapidly, chop-chop, ...	slowly

Results

We have run our algorithm with word pairs from different domains to check its efficiency. The algorithm was able to find the correct expected word for the input words in most cases. However in some cases, it was not able to perform up to the mark as expected. Let us look at some of the examples which we found interesting.

Gender-based and family-based relationships The algorithm was able to understand the similarity of words which share gender-based relationship (*man* - *woman*). This also holds true for most relationships within a family.

king is related to *queen*, as *man* is related to *woman*
brother is related to *sister*, as *prince* is related to *princess*
father is related to *mother*, as *son* is related to *daughter*
grandfather is related to *grandmother*, as *father* is related to *mother*

These examples are logically very sound and this was visible when we plotted the vectors in a 2d-space as we can see in Figure 1 above (we have given the example of only one instance for visual clarity).

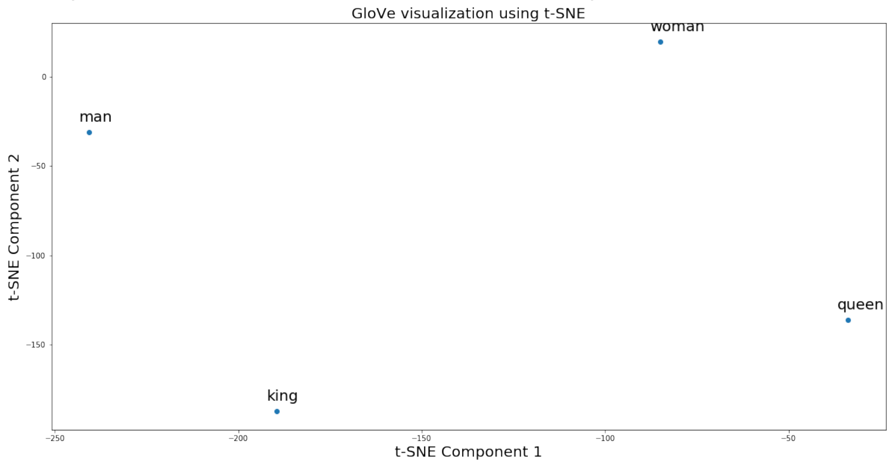
Country-capitals relationships The other kind of relationship that our algorithm could understand very well was the relationship between countries and cities, especially their capitals. Some examples are as follows:

germany is related to *berlin*, as *japan* is related to *tokyo*
germany is related to *berlin*, as *france* is related to *paris*
berlin is related to *germany*, as *delhi* is related to *india*

germany is related to *berlin*, as *australia* is related to *sydney*
germany is related to *berlin*, as *canada* is related to *canada*

We can see that country-capital relation is captured well here and it works in the other way round as well. However, there were some examples in this domain which did not work out as expected. For instance, Sydney is not the capital of Australia (maybe it captured a relation like *country's most important city?*) and it is unable to retrieve the capital city of Canada.

Figure 1: 2d-vector space representation of king, queen, man, woman.



Synonym-Antonym relationships Now let us see how this approach works for retrieving synonyms and antonyms, which was the main goal of our project. We have handpicked some of the word pairs from WordNet for both synonyms and antonyms.

intelligent is related to *clever*, as *stupid* is related to *stupid*
hot is related to *warm*, as *cold* is related to *cold*
better is related to *best*, as *worse* is related to *worst*
cow is related to *cows*, as *horse* is related to *horses*

We can see that this approach **does not perform as efficiently for synonyms** as expected. In most cases, we get the same input word as output. This is mostly because it was unable to find a new synonymous word with the closest possible vector. For example, *foolish* and *cool* would have been much more appropriate answers in the first two sentences in place of *stupid* and *cold*. This could also be the reason why WordNet includes the same word in the list of synonyms for that input word. However, we get much better results when we experiment with the **comparative-superlative forms of adjectives**. Our algorithm also seems to capture the relationships between the **singular-plural forms** of words.

live is related to *die*, as *stay* is related to *die*
feed is related to *starve*, as *live* is related to *starve*
good is related to *bad*, as *kind* is related to *bad*
better is related to *worse*, as *best* is related to *worst*

On the other hand, our approach produces relatively **much better results for antonyms**. It is arguable that we get the same input words as outputs but they are antonyms for the input words. As expected like in the case of synonyms, proper antonyms are produced for the comparative-superlative forms of adjectives.

Conclusion

In this experiment, we find related similar words given a pair of such related words from WordNet through analogy. Retrieving similar words is relatively easier. While this approach works really well for certain relationships (*family relations* and *country-capitals*), getting similar results for synonyms and antonyms is a more challenging task. We get better results for antonyms because generally the source vector-space of words have similar relations (*vector differences*) with the destination vector-space of their corresponding antonyms. On the other hand, synonyms are tougher to track in the same vector-space because even if they may **share the same semantic representation** of the input word, they may lie at a completely unrelated different co-ordinate in that vector-space. However, we can safely conclude that similar and related **words can be extracted through embeddings with respect to specific domains and relationships**.

Future Work

With these results, we can now try to narrow down the experiment to specific individual domains (like parts of speech). Once we get better results, we can probably look into embeddings with more information (like *sentence embeddings*) to find similar information on a sentence level.

Acknowledgments

I would like to sincerely thank **Prof. Alexander Koller**[6] for offering this lecture and guiding us.

References

[1] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

[2] Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

[3] Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov), 2579-2605.

[4] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. J. (1990). Introduction to WordNet: An on-line lexical database. *International journal of lexicography*, 3(4), 235-244.

[6] Prof. Alexander Koller, Professor, Saarland Universty, Germany (`http://www.coli.uni-saarland.de/~koller/`)

[5] Chen, D., Peterson, J. C., & Griffiths, T. L. (2017). Evaluating vector-space models of analogy. *arXiv preprint arXiv:1705.04416*.

[6] Bouraoui, Z., Jameel, M., & Schockaert, S. (2018). Relation induction in word embeddings revisited.