Reverse Dictionary: Searching Words by their Definitions

Nitish Shah, Shikha Verma and Piyush Doke
Team 8

November 26, 2018

1 Introduction and Motivation

Many times we find ourselves unable to recall the most appropriate word for describing the idea/concept we have in our minds. This might just be a recall problem or might be due to our lack on knowledge in that language. And it turns out to be a very common problem for people who produce language. In such cases, dictionaries might not be the most perfect solution as it is easy to find the meaning of a particular word but it is almost impossible to find a word given its meaning. Hence, to address this issue we propose the idea of building a reverse dictionary.

2 Overview and Approach

A reverse dictionary is search engine which finds the semantically most equivalent word given its meaning (as an input to the program). And the approach that we follow is yet interestingly simple. The program is divided into two main sections. The first section of the program is to model the vocabulary and the second section implements searching through the dictionary data-set. Below we have discussed these sections in a greater detail.

First, we use a trained vectorization model to convert all the words in the vocabulary of the language into vectors (of a few hundred dimensions). And these vectors are then plotted into the hyperspace such that words with similar meanings are in close proximity to each other. Next we iterate over all the definitions present in the dictionary to find the most similar ones to our input phrase. This step involves checking similarity between two sentences. Basically, we calculate the Euclidean distance that we have to travel in the hyperspace, for converting all the words of one sentence to the words of the other one.

Using both the above steps we try to find semantically the most similar definitions, hence semantically the most appropriate word by querying for the corresponding key in the dictionary data-set.

Related work

Implementing a Reverse Dictionary, based on word definitions, using a Node-Graph Architecture [1] used one method which makes a graph using the dictionary definitions and then upon query, searches the graph for a word and computes similarity based on the depth. This method is not scalable, as the paper says. Another recent method is using a Recurrent Neural Network or a Long Short Term Memory Neural network, like done by [6] and training them with dictionary definition to output the desired word. These models take a lot of time to train, but give good results. Other methods use the semantic similarity between two words, to get the similarity beteween the query sentence and all the dictionary definitions and then report the K nearest words whose definition was the most similar to the query. Similarity between words is computed using word embeddings, like GloVe: Global Vectors for Word Representations [4] or Word2Vec [7] (both of which encode contextual and hence semantic similarity between words in their embeddings) or using WordNet [8] as is used in[5] by measuring the distance and depth from the root between disambiguated words of the query in the wordnet graph. These similarity measures are then used to measure the similarity between sentences. We use one such measure to make a reverse dictionary.

3 Methods

3.1 Word Embeddings

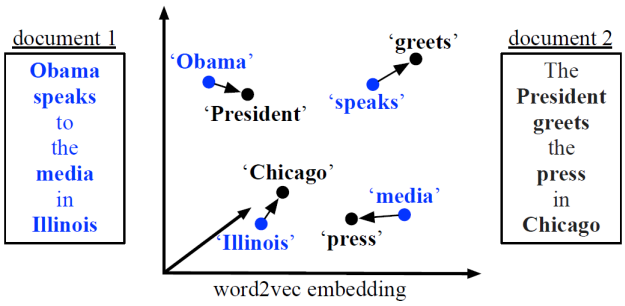

Figure 1. An illustration of the word mover's distance. All non-stop words (bold) of both documents are embedded into a word2vec space. The distance between the two documents is the minimum cumulative distance that all words in document 1 need to travel to exactly match document 2.

Hence, more the sentences are similar, lesser will be the distance between them. Therefore, using this we find semantically the most similar definitions in the dictionary data-set.

Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing the context of a word in a document, and also semantic/syntactic similarities. Word embeddings are nothing but the vector representations of the words. Where the vectors are plotted in a few hundred dimensional hyperspace with the property that, words of similar semantics are in closer proximity to each other in the hyperspace.

In our approach, we use Word2Vec [7], which is one of the most popular techniques to learn word embeddings using a shallow neural network. We use a pre-trained Word2Vec model which is developed by Google and is trained on Google News Dataset. The main reason behind choosing this pre-trained model was because news articles contain formal texts which are good for checking definition similarities.

Query	Result	Target Word
to feel happiness or pleasure	ecstasy, euphoria	enjoy
the word for painting music theatre sculpture and other creative activities	artwork	art
to move your mouth or jaw up and down in break up or soften food	crush	chew
to think into the future and feel that is is likely that a particular event will happen	enlighten	expect
to take the material from something and making it into something new rather than throwing it away	revise	recycle
to decide who someone is or what something is after thinking about it	identify	identify

3.2 Similarity Measure

Checking the similarity between sentences is the second major component to our approach. To check for similarity, we propose using the method - Word Mover's Distance (WMD) [9]. The Word Mover's Distance measures the dissimilarity between two text documents as the minimum amount of distance that the embedded words of one document need to travel to reach the embedded words of another document.

3.3 Getting the Word from its concept

We use WMD to chech the similarity between the query and all the definitions in dictionary dataset. We return the words whose definitions are most similar to the query. i.e. We return some k nearest neighbours of the query.

4 Experimental Analyses

Datasets

We use the dictionaries and the training data that was made publically availabe by [6] which has around 850 thousand different definitions of around 67 thousand different words. This dataset is compiled using dictionary definitions from five electronic resources: Wordnet, The American Heritage Dictionary, The Collaborative International Dictionary of English, Wiktionary and Webster's. The training data was made by asking some native English speakers to describe some predecided words and confirming the quality of each description by checking if someone who didn't write the description was able to guess the target word.

Results

Calculating WMD is slow. Estimate time to complete one query on the whole dictionary turned out to be about 5 minutes. Due to this, for obtaining results in a reasonable time, we constrained the program to check only the words which start with the target word's first letter. Out of the 30 description words we tested, only 6 concepts managed to find similar meaning meaning words in the first 10 results and one target word actually appeared in the results (the six similar words are shown in the table below). The rest of the words were not semantically similar to the query. This is because often the definitions of a dictionaries try to be as accurate as possible, whereas human descriptions of words are not. The long definitions, extra descriptions of words in the same sentence, all these tend to increase the dissimilarity between the query and dictionary definitions, even though their meanings are the same.

The six relevant results are shown in table above.

5 Discussion and Future Directions

We have shown a method to make a reverse dictionary. Heuristics can be implemented to compute WMD for the dictionary definitions as used in [9], which describes a prefetch and prune algorithm. Different embeddings could be used trained on different data to improve this algorithm. A better similarity measure could be implemented that incorporates word order along with word similarity. The results, though poor, seem promising.

References

[1] Sushrut Thorat and Varad Choudhari: *Implementing a Reverse Dictionary, based on word definitions, using a Node-Graph Architecture*. Available from World Wide Web: (http://www.aclweb.org/anthology/C16-1263)

[2] D. Gaya: *A study of building a Reverse Dictionary*. Available from World Wide Web: (http://www.ijsrp.org/research-paper-0715/ijsrp-p4394.pdf).

[3] *A Beginner's Quide to Word2Vec and Neural Word Embeddings*. Available from World Wide Web: (https://skymind.ai/wiki/word2vec).

[4] Jeffery Pennington, Richard Socher and Christopher D. Manning: *GloVe: Global Vectors for Word Representations*. Available from World Wide Web: (https://nlp.stanford.edu/projects/glove/).

[5] Atish Pawar, Vijay Mago: *Calculating the Similarity Between Words and Sentences Using a Lexical Database and Corpus Statistics*. Available from World Wide Web: (https://arxiv.org/pdf/1802.05667.pdf).

[6] Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016b. Learning to Understand Phrases by Embedding the Dictionary. Trans- actions of the Association for Computational Linguistics 4 (2016), 17–30. Available on: https://arxiv.org/abs/1504.00548v4

[7] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In Proceedsings ofWorkshop at ICLR, 2013a.

[8] G. A. Miller, "Wordnet: a lexical database for english," Communications of the ACM, vol. 38, no. 11, pp. 39–41, 1995.

[9] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In International Conference on Machine Learn- ing. 957–966.