- Assignment 1 -

Information Retrieval

May 13, 2019

Word Embeddings

1

1.1 Warm up

1. The pretrained word embedding I used for this assignment was *word2vec*. The *word2vec* word embedding was created in 2013 by a team of researchers at Google. Word2Vec is a 2 layer neural network that takes a set of documents and learns word embeddings based on math on mathematical similarities between words in documents. In the case of the Word2Vec instance used for the pretrained word embedding the words are represented as a vector of length 300.

2. The List of most similar words to "King","London", "Good", and "Apple", can be seen in Table 1.

Table 1: Most Similar Words

Word	King	London	Good	Apple
1st	Jackson	EURASIAN_NATURAL_RESOURCES_CORP.	Bad	Apple_AAPL
2nd	Prince	Londons	good	Apple_Nasdaq_AAPL
3rd	Tupou_v	Islamabad_Slyvia_Hui	Decent	Apple_NASDAQ_AAPL
4th	KIng	Wandsworth	Better	Apple_Computer
5th	e_mail_robert.king_@	Canary_Wharf	LAKE_WYLIE_Largemouth_Bass	IPhone

3. We can see from the scatter plot that the word embeddings are clustered around the right side of the diagram. Most of the outliers are on the far left of the diagram.

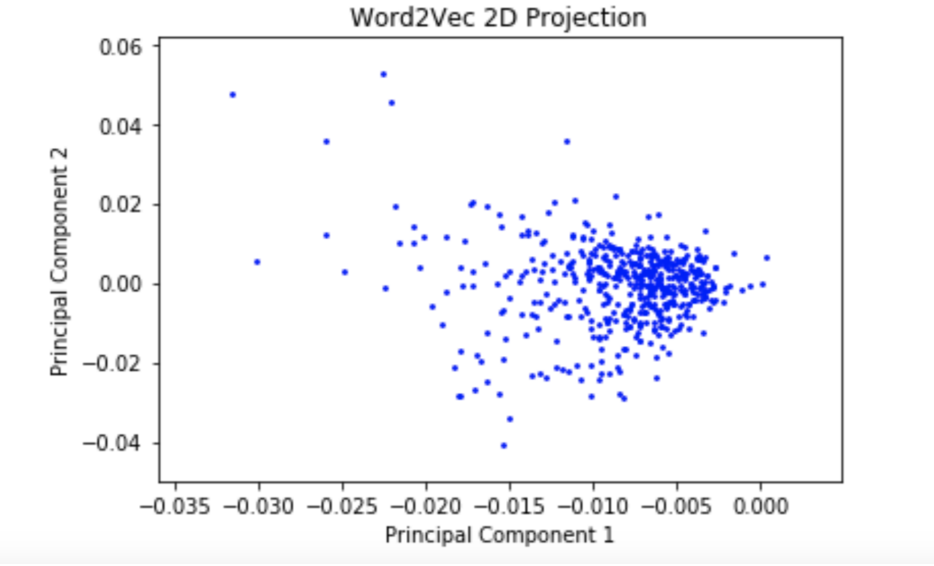

Figure 1: Projection of word embeddings on first two principal components

1.2 Main Task

One of the cool features of the word2vec word embedding is that we can get meaniningful answers using the vector offset method. If A is to B as C is to D. Then we can take try and attain the word embedding of D, by taking the value of

$$V = B + C - A$$

Then we can take the vector that is nearest to V and it is possible that we get a good guess for D if not the answer we were actually looking for. Using this method on the Google Analogy Test Set we get the following results in table 2.

Table 2: Analagy Results

Type	Semantic	Syntactic
Test Accuracy	0.1735	.2253

2 Word Embeddings for Text Classification

Word embeddings can be used to aid in classification tasks such as sentimenet analysis. The IMDB dataset consists of 25,000 training reviews and test reviews, with half of the data in both sets being being positive reviews, and the other half being negative reviews.

Document Representation

Word Embedding map words to vectors. For the task of sentiment analysis we have to classify a list of words based on which other documents are most similar. I elected to use the average word embedding as a way to represent the document. The average word embedding is the sum of the word embeddings of the words normalized by the length of the document (Number of words in the document).

Classifier

Once the Documents were mapped to their average word embedddings, I tried two different classifiers Random Forests and Linear Support Vector Machines.

Table 3: Classification Results

Classifier	Random Forest	Linear SVM
Test Accuracy	0.7185	0.8191

Results

The results of training the two classifiers on the average embedding representation can be seen in Table 1. The Linear SVM does almost 0.10 better. The classification is fairly good considering that we are essentially classifying the normalized average of word embedding.

3 Extend Word Embeddings Models to the State of the Art

In this section we test Contextually Propogated Term Weights for Document classification. We compare the results with the popular TF-IDF algorithm.

TD-IDF

Using KNN after embedding documents with K-Neireght Neighbors we get the results seen in table 4.

Table 4: Classification Results

TD-IDF	Micro F1-score	Macro F1-score
Test Accuracy	0.8401	0.7862

The results are fairly good. We can use this as a baseline for comparing the relative performance of CPTW.

CPTW

Unfortunately my implementation of CPTW was not efficient, and the preproccsing of one document to the embedding described in the algorithm takes approximately 13 second. Since there are over 8,000 documents this ends up taking $13 * 8,000$ seconds, which is over 28 hours. I therefor do not have the results for the CPTW implementation yet. To be Continued.