ASSIGNMENT 3

A PREPRINT

Didem Yaniktepe
`b21527563@cs.hacettepe.edu.tr`

May 17, 2019

1 Introduction

In this assignment, I learned the quality of the word embedding for analogy task and document classification with Doc2Vec model. Word embedding is often used as a word. In this way, word burials are sometimes called semantic vectors. I used the Word2Vec library. The latter is also a library commonly used for neural networks.

I used one of the many well-known gensim doc2vec applications to extract features from the text bodies in my data set. My goal was to obtain a logistic regression model that was eventually trained by the doc2vec feature to classify invisible documents. Therefore, the generalization ability of the model is important in this case

2 Word Analogy

2.1 Google's Vector

There are several existing models for constructing word-embedding representations. Google's word2vec is one of the most widely used implementations due to its training speed and performance.So we used ̈GoogleNews-vectors-negative300.bin ̈Word2vec is a predictive model, it is trained to predict a target word from the context of its neighboring words. The model first encodes each word using one-hot-encoding, then feeds it into a hidden layer using a matrix of weights; the output of this process is the target word. The word embedding vectors are are actually the weights of this fitted model. To illustrate, here's a simple visual:

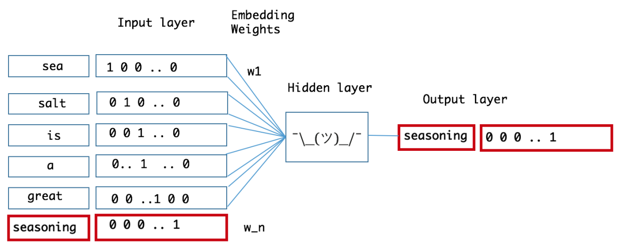

Figure 1: Embedding

2.2 Word2Vec

Normal vocabulary encoding doesn't help identify the relation between two words.We want a way to identify man and woman are opposites and apple and orange are similar because they're fruits. So we create multiple features per word. And give each feature a number. Embedding would train on large embedding of unlabelled text. A general word embedding use case would look something like this.

2.3 Word Analogy

Word analogies can be made by taking the difference of their embedding vectors. Man:Woman :: King:? eMan - eWoman = eKing - e? Compare the vector difference between man-woman and king-queen Find a word W such that the cost between the embedding differences is minimum. cosine_similarity(eW, eKing - eMan + eWoman) should measure the similarity between the two words. I used Cosine similarity for this task.

Given two posts t_a and t_b, their cosine similarity is

$$\cos(\mathbf{t_a}, \mathbf{t_b}) = \frac{\mathbf{t_a}\mathbf{t_b}}{\|\mathbf{t_a}\|\|\mathbf{b}\|} \tag{1}$$

a	b	x	y	$Cos_{S}im$
London	England	Baghdad	Mosul	0.648
London	England	Baghdad	Iraq	0.727
London	England	Oslo	Norway	0.506
London	England	Tokyo	Japan	0.574
Cairo	Egypt	Hanoi	Vietnam	0.9754
Cairo	Egypt	Hanoi	Laos	0.9746
boy	girl	father	mother	0.953
boy	girl	father	daughter	0.956

Given the analogy a is to b as x is to y,let $Cos_{S}im = ({}_{y},{}_{x} -{}_{a}+{}_{b})$

As shown in Table, both distance measures yield a greater value for the correct answer, Iraq, regardless of the embedding dimension. This disagrees with the results in, but the computation involves different word embeddings.Even in the example “boy is to girl as father is to mother”, where the greater distance, gender, should be relatively.

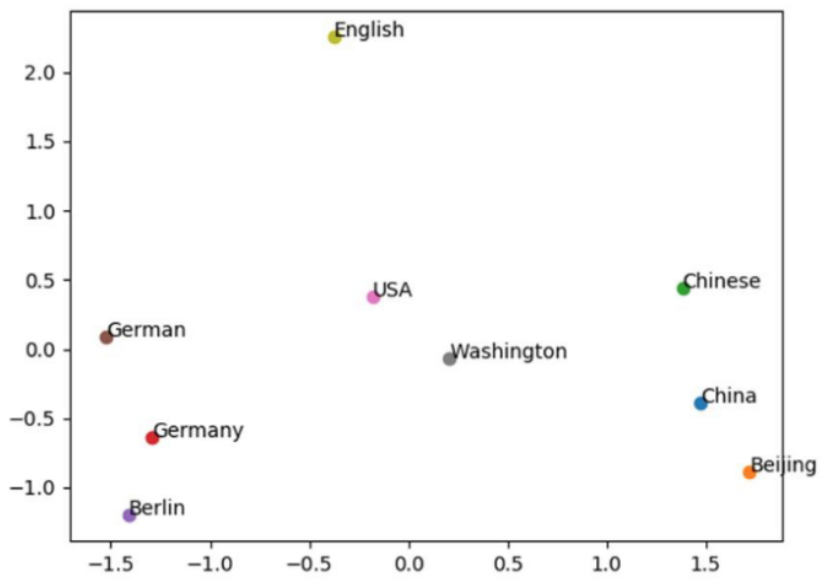

Figure 2: Embedding

2.4 Evaluation

The evaluation lasted very long time.So I did limit for Google vector. My accuracy is 0.2189930413426 it is because i did not evaluate all vectors in the Google vector. When I get the test set and and only using these vectors my accuracy was %86.

```
def cos_similarity(x, y):
    dot_product = np.dot(x,y)
    norm_x = norm(x)
    norm_y = norm(y)
    cos_sim = dot_product / (norm_x*norm_y)
    return cos_sim



def find_analogy(first_a, first_b, second_a, word_to_vec, all_words):

    try:
        f_a, f_b, s_c = word_to_vec[first_a], word_to_vec[first_b], word_to_vec[
                                                  second_a]
    except(KeyError):
        return


    max_cosine_sim = 0
    best_word = None


    for w in model.vocab:

        if w in [first_a, first_b, second_a]:
            continue

        u = f_b - f_a + s_c
        v = word_to_vec[w]

        cosine_sim = cos_similarity(u,v)

        if cosine_sim > max_cosine_sim:
            max_cosine_sim = cosine_sim
            best_word = w


    return best_word
```

3 Document Classification

3.1 DataSet

I learned classifying the movie plots by genre using Doc2vec for feature representation and using logistic regression as a classification algorithm. The movie dataset contains short movie plot descriptions and the labels for them represents the genre. There are six genres present in the dataset:

- Sci-fi
- Action
- Comedy
- Fantasy
- Animation
- Romance

Total train data size = 2000 , total test data size = 428. For text processing i removed punctions, stopwords and i do stemming for my dataset.

3.2 Doc2Vec

Doc2vec modifies the word2vec algorithm to unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents.

Since the Doc2Vec class extends gensim's original Word2Vec class, many of the usage patterns are similar. We can easily adjust the dimension of the representation, the size of the sliding window, the number of workers, or almost any other parameter that you can change with the Word2Vec model.

We found that by concatenating 3 models the accuracy among the test cases increased , we think this is because the number of features has increased and each model is trained on a different set of data.

I use this hypermaters for Doc2Vec

```
model = Doc2Vec(min_count=5,window=15,vector_size=400,
                workers=multiprocessing.cpu_count(),alpha=0.025, min_alpha=0.00025
                                                        , dm=0,dm_mean=1,
                                                        dm_tag_count=1)
```

```
min_count# Ignores all words with total frequency lower than this
window # Maximum distance between the current, predicted word in sentence
vector_size # Dimensionality of the generated feature vectors
workers # Number of worker threads to train the model
alpha  # The initial learning rate
min_alpha # Learning rate will linearly drop to min_alpha as training progresses
dm #defines the training algorithm. If dm=1 means 'distributed memory' (PV-DM) and
                                          dm =0 #means 'distributed bag of words'
                                         (PV-DBOW)
```

We have also introduced the unsupervised reviews to one of the models to increase the vocab of it and labels in **doc2vev** are all unique for every input so we assign different and unique labels to these unsupervised reviews. This technique can be seen as ensemble of more than one model to increase accuracy

3.3 Evaluation

My training accuracy 0.836 My Testing accuracy 0.5011709601873536. My testing accuracy is changing every time i ran the my code. I increase the epoch because when i decreased it accuracy is decreased. I change the window size when i decreased window size training accuracy is increased but testing accuracy does not change too much.