Report for COMP6714 Project 2

Writer: Wenzheng Li

Date: 16th Nov. 2017

1. Abstract

In this report, there will be a brief introduction on the implementation of the project as well as the aim of the project. Then, the methodology section covers the detail of some core functions and the reasons of the process. Next section will demonstrates the results and a brief discussion on this project also provide. Finally, a conclusion is made to cover the result and the whole project.

2. Introduction

a) This project has referenced some ideas and code from Word2Vec_Demo.

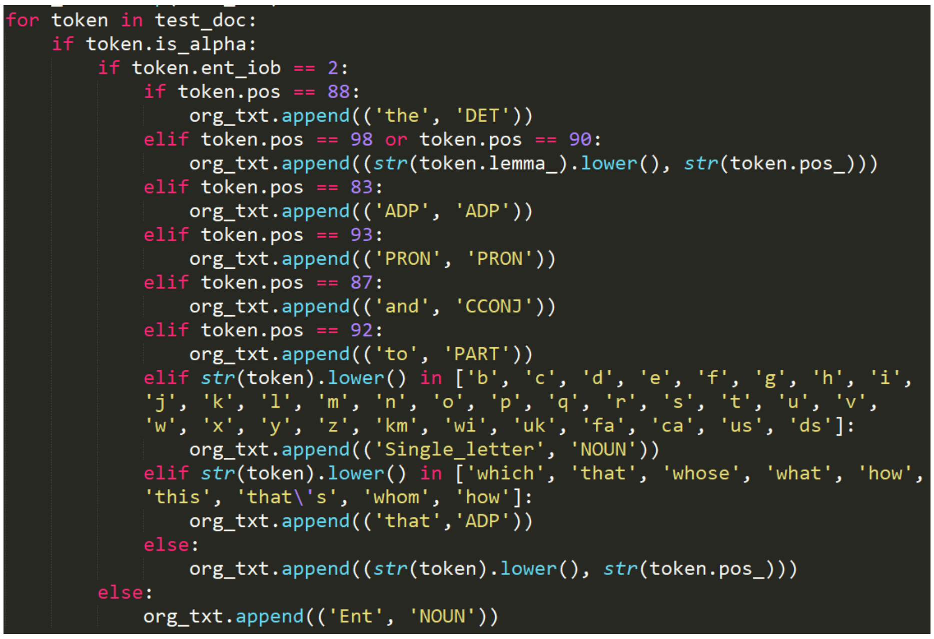
b) The project begin with pre-processing the data and then generate the training data. Meanwhile this process can generate a file store each word's vectors.

c) As for the training model section, we use the function from genism and get the top K adjectives.

3. Methodology

a) Data pre-processing

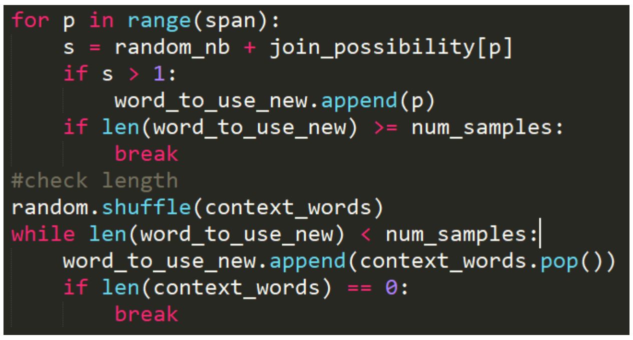
In data pre-processing part, after we unzip the file from the file and store words in a list, we use Nature Language Processing to process the data. For each token (word), we delete the numbers which is useless for training. The words such as conjunction words and some single letters are replaced by the words with same properties. Because this kind of words contribute marginally to training adjectives. Besides, we let all entities replaced by one word, because they have the same function in a sentence for understanding the adjectives. Due to the fact that less frequency provide more information, we let all verbs and nouns back to prototype. Finally we also store each word's part of speech for later processes. Thus in the dictionary as well as reverse dictionary, we not only store the words, but also store its types.



```
for token in test_doc:
    if token.is_alpha:
        if token.ent_iob == 2:
            if token.pos == 88:
                org_txt.append(('the', 'DET'))
            elif token.pos == 98 or token.pos == 90:
                org_txt.append((str(token.lemma_).lower(), str(token.pos_)))
            elif token.pos == 83:
                org_txt.append(('ADP', 'ADP'))
            elif token.pos == 93:
                org_txt.append(('PRON', 'PRON'))
            elif token.pos == 87:
                org_txt.append(('and', 'CCONJ'))
            elif token.pos == 92:
                org_txt.append(('to', 'PART'))
            elif str(token).lower() in ['b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
            'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
            'w', 'x', 'y', 'z', 'km', 'wi', 'uk', 'fa', 'ca', 'us', 'ds']:
                org_txt.append(('Single_letter', 'NOUN'))
            elif str(token).lower() in ['which', 'that', 'whose', 'what', 'how',
            'this', 'that\'s', 'whom', 'how']:
                org_txt.append(('that','ADP'))
            else:
                org_txt.append((str(token).lower(), str(token.pos_)))
        else:
            org_txt.append(('Ent', 'NOUN'))
```

b) Generate batch

We also modified the generate_batch function which is being called in each step when training. When we testing the original code of getting training word from buffer, it reflect a bad performance (about 3 to 4 average hits out of 100). But when I make lower frequency word have a higher possibility to be sampled, the results get better. That is because rare terms are more informative than frequent terms. The code as follow.



```
for p in range(span):
    s = random_nb + join_possibility[p]
    if s > 1:
        word_to_use_new.append(p)
    if len(word_to_use_new) >= num_samples:
        break
#check length
random.shuffle(context_words)
while len(word_to_use_new) < num_samples:
    word_to_use_new.append(context_words.pop())
    if len(context_words) == 0:
        break
```

But in some case, the sampling list may not full, thus we also use random shuffle the make up the vacancy.

c) Compute top K

As for Compute_topk function, genism provide good method so that we can easily generate mode as well as most similar words. One thing need to note that we generate more than top K words so that we can only pick up adjectives to the finally list (as we announced that we put the words' type in the data).

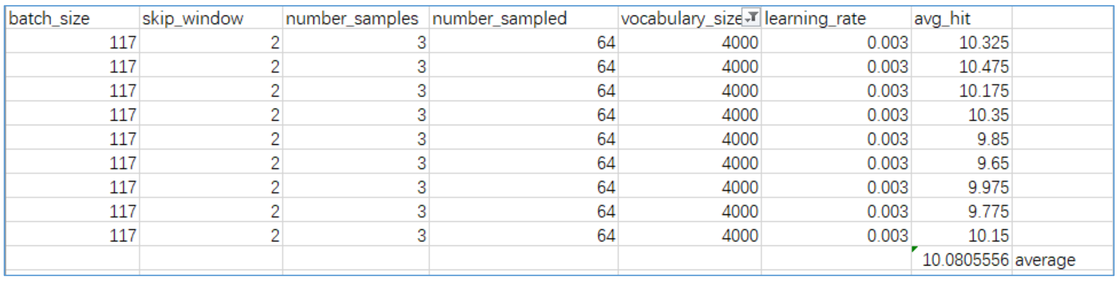
4. Result and Discussion

a) Parameter adjustment

Firstly we notice number sampled have marginally affects to the result, so we keep the value to the original value. As for the batch size, there is a gradually increasing as we reducing the size from 128, after about 117, the uptrend turn to downtrend gradually. After many times adjustment, it is the best for skip window to be 2 and number samples being 3. Which means, apart from centre word, we select 3 out of 4 as samples. When adjusting vocabulary size, a smaller size has a better performance comparing to a large size. After many times attempts, 4000 owns a better performance than other values. Finally, the learning rate, when we increasing the value from 0.001, the performance get better, but turn worse after 0.003, thus 0.003 should be selected.

b) Final result

Thus, as we select batch size to be 117, skip window to be 2, number samples 3, number sampled 64, vocabulary size 4000 and learning rate 0.003, we can make a simply statistic as follow:



batch_size	skip_window	number_samples	number_sampled	vocabulary_size	learning_rate	avg_hit	
117	2	3	64	4000	0.003	10.325	
117	2	3	64	4000	0.003	10.475	
117	2	3	64	4000	0.003	10.175	
117	2	3	64	4000	0.003	10.35	
117	2	3	64	4000	0.003	9.85	
117	2	3	64	4000	0.003	9.65	
117	2	3	64	4000	0.003	9.975	
117	2	3	64	4000	0.003	9.775	
117	2	3	64	4000	0.003	10.15	
						10.0805556	average

Thus in this condition, the average hit can be about 10.08 out of 100.

5. Conclusion

   Finally, we can notice that the average hit seems not very high, but this result is based on the limited learning materials and limited training steps. If there is enough material to be trained, I believe that the result can be much better. As for the project itself, we can see that different frequency of the word may make different contribution to the training, meanwhile, the arguments may also affect the result in different condition.