Natural Language Systems

Coursework 1

Alexander Angelov

Task 1

In order to find which POS tag is more likely to be used for the token “race” in the given sentence, we need to calculate the probability for both P(NN)=A*P(race|NN)*P(NN|AT)*P(IN|NN) and P(VB)=A*P(race|VB)*P(VB|AT)*P(IN|VB), where A is the product of the other probabilities (not related to the POS tag of the token/word “race”) and has a value between 0 and 1.

Firstly, the tagged_words method was used to load the Brown corpora as pairs/tuples of token and corresponding POS tag. Secondly, I found the word likelihood probabilities by getting the number of times the word “race” appears in the Brown corpus as the given POS tag (NN and VB) over the number of times the corresponding POS tag appears in the corpus. For example, P(race|NN) = C(NN,race) / C(NN). After calculating, the following results are observed: P(race|NN) = 0.0006; P(race|VB) = 0.00012. Thirdly, a function was created for calculating the transition probabilities. For example, if we want to find the transition probability P(Tag1|Tag2) = C(Tag2,Tag1) / C(Tag2), we first find all occurrences of the phrase **Tag2 Tag1** in the Brown corpus, where Tag1 follows Tag2. Then we divide the result by the number of occurrences of the tag **Tag2** in the corpus. By using this function, I calculated the transmission probabilities: P(NN|AT) = 0.49; P(IN|NN) = 0.277; P(VB|AT) = 0.00016; P(IN|VB) = 0.14. At the end, the total probabilities were calculated: P(NN) = A*0.00008; P(VB) = A*0.0000000027.

We can observe that the probability of “race” being a noun(NN) is around 30912 times higher than the probability of “race” being a verb. This results in POS tag NN to be much more likely to be used in the given sentence.

Task 2

a) Implementation

First, the corpus is divided into documents, because different documents contain different topics. Then the punctuation, digits and special symbols such as dollar sign are removed in each document. This information usually does not include any additional information and it contains lots of outliers, because digits may change and the context window captures less other terms, contributing more meaningful information. Afterwards, the program goes through every word of the document and creates an array with maximum length 2n + 1, containing the occurrence of the target word and the previous n and next n words if such exist (n is the size of the context). By default, the context size is set to 5.

Then a word-by-word matrix full of zeros is created with a size k*m, where k is the number of target words, and m is the number of distinct words in all contexts. We increment a cell A[tw][t] in the word-by-word matrix, if we encounter that the target word “tg” is in the same context as the term “t” (if we have only one occurrence of the term “tg”, we do not increment A[tg][tg]).

A K-Means clustering is used to cluster the 50 target words. For this method, the number of clusters needs to be specifies (50 in our case). The K-Means method return a label of (from 0 to 49) for each target word.

b) Evaluation

After running the program on the Brown corpus we can observe that we need only 48 clusters to classify the data. We receive this result, because 3 of the words does not appear in the brown corpus and can be classified in 1 cluster.

A reverse function is created where the positions of every target word is found. Then a random function is used to choose half of the occurrences of each target word and the corresponding reversed target words are set on their places. Then a word-by-word matrix is created from the new target words (the original 50 and the reversed 50) and the new corpus. The matrix is used to classify each target word to the 50 clusters. In addition, an average accuracy function is created for performing the whole process 5 times and to give the average accuracy of the program. Also, the number of "correct" pairs is counted and visualized.
If we perform the operation on the Brown corpus, we can observe that the average accuracy is around 64% which gives an average of 32 "correct" pairs. We can observe that almost 20 pairs do not pair. I believe there are few reasons for this result. Firstly, few of the target words may be synonyms (ex. product and object) in few contexts. This may result in being in contexts with similar terms. Another reason is that target words may result into the same word after stemming (ex. argument and arguments) which mean that they may be surrounded by the same/similar tokens.

c) Analysis

1. The size of context used.

Window size plays a key role in the program. As a default this parameter is set to 5 and is producing average performance (around 64%). One of the experiments in the program is the effect of the accuracy of the program, when the size of the context is increased. An experiment is made with context size 5, 10, 15, 25 and 40. We can observe that the average accuracy is increasing when increasing the context size. The accuracy spikes from 63% to 85%, when we only increase the context size. This improvement is seen, because by increasing the size, we include more context for each target word. In such a way, we can distinguish better between synonyms. Also few target words may be part of phrases including similar tokens (ex. set up, set down, …) and a larger context may increase the ability to distinguish them and cluster them in different clusters.

2. Two additional features can be used on tokens.

a. Stems

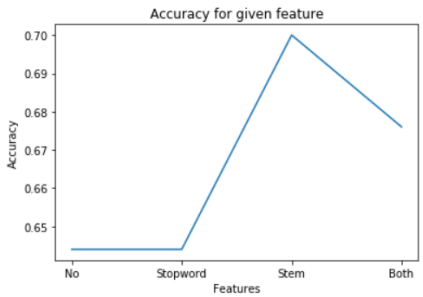
When we use stemming we can observe that there is a minor improvement of the performance with 6%(from 64% to 70%). These 6% corresponds to 3 "correct" pairs which is a good increase. Moreover, stemming reduces the number of distinct words that are used and reduces the number of columns in the matrix. This speeds up the program, but not very significantly.

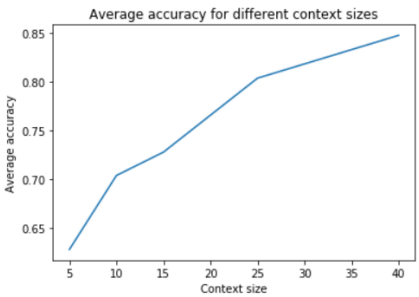
b. Stop-words

When we remove the stop-words in the corpora (without the one in the target list) we can see that the accuracy is similar to the original one. It does reduce the feature dimension length as the stemming feature, but it adds almost no improvement neither to the speed of the program, nor to its accuracy.

c. Both stems and stop-words

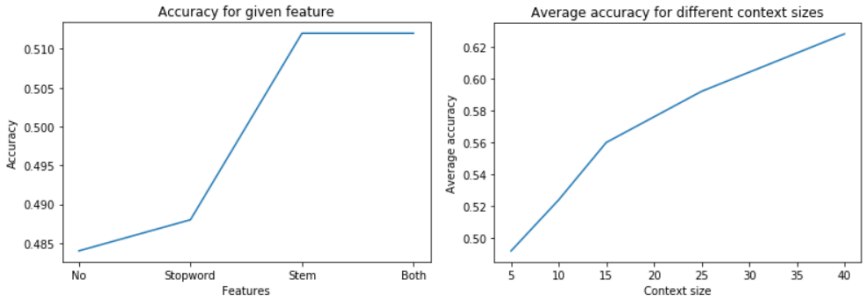
However, when we apply both stemming and stop-word removal we can see between 3 to 4% increase in the accuracy and average improvement in the speed.





3. Product review corpora

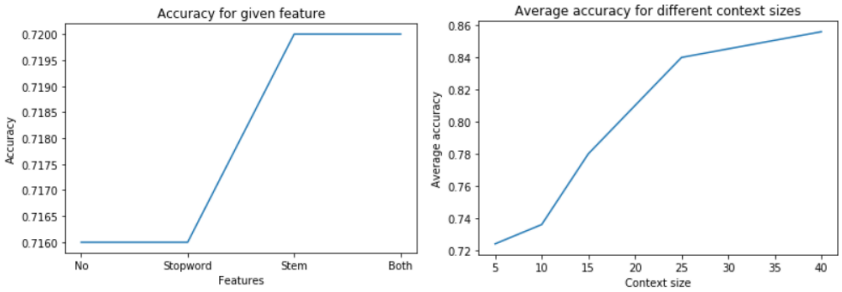
In addition, to the Brown corpora, a Product review corpora was also used. When we conducted the same experiments but with the new corpora, we get much lower results. This is mainly, due to the fact that the new corpus is much smaller (with less training data) and does not contain more of the target words than the Brown corpus. When we look at the accuracy, when different corpus size is used, we can observe that the average accuracy is changing from 49% to 63%, which is with around 20% lower than the results received for the Brown corpus. When we apply stemming and/or stop-word removal, we receive similar results. If we apply only stemming, we get an increase of around 3%. If we apply only stop-word removal, we receive around 1% increase from the original accuracy. If both are applied, then we receive again between 3% and 4% increase.



4. Combined corpora

When both corpuses are combined, the execution gets slower, but the results become more accurate. The reason is that the data is from 2 different sources, which makes the training data larger and more diverse. When we are changing the context size from 5 to 40, we can observe that the accuracy spikes from 72% to 86%, which is only around 7 pairs away from the perfect clustering. The accuracy is both greater than both received from the Brown and Product review corpuses.

When we perform the only stop-word removal or only stemming of the corpus using context size 5, we can see similar results as the original or slight increase with up to 1%. When we apply both stemming and stop-word removal, we can observe also 1% increase in the accuracy. I believe that both stemming and stop-word removal do not contribute a lot to the model and may not be used in this corpus.



5. Different features/resources

Instead of specifying context size, each document can be used for the context of any target word it contains. As a result, we way create word-document matrix for capturing co-occurrence patterns. I believe that this may result in inaccurate results, because the entire document may be the cortex for two different target words, which may lead to incorrect clustering. This is the main reason why it has not been implemented.

Moreover, POS tags may be used in the context in addition to the tokens. This operation may be beneficial to the data. For example, if this method is applied, numbers may be preserved by changing their value to the corresponding POS tag. This may lead to more accurate results. However, using POS tags can increase the memory usage drastically and increase the word-by-word matrix. Also, POS tags may be less accurate for bigger context sizes, if the tag is used on its own (not connected directly with the term).