Haorui Chen-10407315-COMP34412-cwk1

## 1 Part-of-speech (POS) tagging

1. The POS tagger I used is NLTK 3.4.5.
2. Word likelihood probability P(wi | ti) can be calculated as C(wi, ti)/C(ti).

First I obtain a list of POS tags appeared in tagged corpus A by going through the corpus, then, for each tag, I build a corresponding list in which I collect all words in the corpus with the specific tag attached. Finally, for each list I get, I count occurrence times of each words, getting C(wi, ti) for each word, and divide it by total number of occurrences, C(ti), of that tag in the corpus to get P(wi | ti), likelihood probability for each word.

1. Tag transition probability P(ti | ti-1) can be calculated as C(ti, ti-1)/C(ti-1)

a matrix is first set up to count C(ti, ti-1), occurrences of different tag transitions. Each element in the matrix means number of times ti, the tag given by its column, follows ti-1, the tag given by its row in the tagged corpus. I go through the whole corpus to get the final result of the matrix. Finally each element C(ti, ti-1) is divided by corresponding occurrences of ti-1 to get final tag transition probability P(ti | ti-1) for each pair of tags.

1. The tag is correct by English grammar. From the corpus the system learns that word ‘race’ shall never be used as a verb ( although this is not the case ), making possibility of ‘race’ being assigned a VB tag be 0, smaller than that of the word assigned a NN tag.

## 2 Distributional semantics

1. The program takes into consideration of stemming and window size.

a. If the whole document is used as context of target words (a.k.a ‘targets’ below), all words except targets are turned to their stems if we choose to use stem as context. Then the whole corpus is first gone through to collect all vocabulary words/stems.

For each document, we see it as multiple windows for those targets occur in it. The exact number for each target depends on term frequency of it in current document. For each vocabulary we add this number to corresponding cell in target-context matrix. At last, the same number is subtracted from the cell which row and column all represents the target, as itself cannot be context word/term.

b. If a window is chosen as context, windows for all targets in corpus are first collected with knowledge of location of every target in corresponding window. Then for each window, all words except target are turned to their stems on using stem as context. At last, for all the windows we count the frequency of co-occurrence of targets and contexts.

c. After target-context matrix is get, we use Kmeans to cluster word vectors, build required number of clusters and assign each word to the cluster indicated by result of Kmeans for convenience of presenting result.

1. For the baseline, I use the whole document as context window, word instead of stem as context word and corpus B as corpus.

My standard of a ‘true’ cluster is that it shall contain exactly a pair of origin and reverse, being exactly at size of 2.

Accuracy I got is around 32% across multiple runs.

Using the whole document as context window, we are very likely to have many targets using the same context(document), thus have the same vector at the end of each document. Similar circumstance may arise for multiple documents, as a consequence, targets and their reverses may have similar results that make them hard to distinguish. Even if a pair of origin and reverse are put into the same cluster, it is likely to be contaminated by other words. All these make a true pair hard to appear.

1. I have done various experiments over context window size, choice of window, usage of stemming and corpus.

a. For the choice of using whole document versus window as context, using window always give obviously better performance than using whole document, no matter what size the window is (with in those I have tested). Heuristically, especially when the document is too long, a word too far away from target has little relation with it and usually doesn't be considered as part of 'context'; also, using too many words as context will actually cause each context word be less helpful on distinguishing targets, as other windows may also contain these words. Just as I mentioned above, doing this means eventually lead to obvious similarity among target vectors.

b. I found using word and stem as context have similar performance. With other conditions kept the same, the curve of accuracy versus window size looks similar when for word and term. The reason is that for the range of window size I used, different words that share the same stem do not yet started to appear massively, so a vector built on stems shall not be obviously different from that built on words, except for its length due to shrink of vocabulary size; since the vector of words for a target is similar with that of stems, the result of clustering will consequently not differ obviously. I also notice that when using whole document as context, using stem has a way worse performance than using word, the reason is that in this case the side effect of different words sharing the same stem start to have larger impact, which makes target vectors look more similar and thus more difficult to cluster.

c. As for context window size, I have experimented among range of 2 to 20 words on each side of target. It shows that as window size increase, accuracy will go down very slightly and then starts to fluctuate. When size of context being fairly small, context words of one target are likely be quite different. This is guaranteed by the fact that targets have different POS tags which lead to different domain of context word selection. Also, targets have their own meanings, so for one target some words may be more likely to appear around it and for another target a different cluster of words may appear, this further narrow down size of domain of context of a target and makes context distinguishable. As window size grows, more commonly-used words and eventually stopwords will be added in context. While these words still contribute to context pattern of target, they will also be more likely to appear in context window of other targets, adding slight similarities to target vectors. In the meanwhile, amount of such kind of words is still limited so a performance still high enough can still be achieved.

d. Corpus C and combined usage of corpus B and C both give far better performance than Corpus B as training data, because type of tokens in corpus C is more limited-mostly English words and digits; however, in corpus B tokens are more varied as mathematical expressions bring more kinds of characters and complex digital strings, which means a significantly larger corpus is needed to collect those increased context patterns added to targets. Corpus B and C are roughly at the same size, yet context patterns of target in corpus C is more limited and more regular, so from the result we can see that vectors for targets can already be distinguished from each other and more origin-reverse pairs can be got as it can be concluded from the corpus that they share more similar context pattern. However, for corpus B, such a size may not be enough to reach the same conclusion, especially for contexts related to those characters and complex digit strings. This is also the reason why performance of combined usage of B and C gives a slightly worse accuracy.

e. Rather than using counts to indicate frequency of co-occurrence, we can also use binary indicators for each target-context pair, if a word appears in context, no matter how many times, we set corresponding cell to 1; also, we can manually design weights for different kinds of words to indicate their importance in context.