CS 325 HW1

Qingyuan Zhang

In my ZhangSegment program, I improve mainly 3 aspects to generate word segments better and efficiently. Firstly, I define a certain threshold of unigram. If a word in our hashtag that exceed likelihood of 0.0000004 in unigram, we increase the likelihood of this certain segment when we put it into the sequence. As a result, the sequence contain the certain words exceeds the threshold will more likely to be chosen. By doing this, we can obtain a better segmentation with familiar words. Secondly, I count how many bigram likelihood appears in each sequence and find the sequence with highest ratio between the words appears in bigram and the size of the sequence. A correct segmentation is often the one with more words in the bigram and less size. Finally, I try to solve the problem that the original segment program takes long time to test the long hashtag. I simply stop the recursion when the likelihood of certain sequence becomes so low.

During testing the following tags:

#2thingsthatdontmix
#10turnons
#90s
#100thingsaboutme
#alliwant
#annoyingbios
#aprilfoolsjokes
#areallygoodejob
#arelationshipisoverwhen
#artistseveryonelikesbutidont

My first improvement of word segment can separate the words in unigram fairly good. However, when it deals with words such as about and April. It cannot generate with a correct segment because the single word “a” has a comparatively large likelihood to appear. Since my program increase the likelihood when word in sequence appears exceed certain threshold. As a result, the sequence including “a” get a very large likelihood.

My second improvement of word segment mainly deals with the unknown words. During test with hashtag #imetjinho. The correct segment is “I met jinho”, which includes one words pair that appears in the bigram and sequence length with three. So I multiply the original d_total with 1/3. For other sequence, which has words pair that appears in the bigram, for the same reason, we multiply its original d_total with number of time the bigram count divided by the length of the words. By doing this process, we generate more correct segment with unseen words

Finally, I deal with the problem that the original segment program takes so much time testing the long hashtag. The reason why it takes so long is that the recursion runs every possible segment of that hashtag. In order to solve this problem, I reduce the number of recursion by stop the certain recursion that lower than a threshold of likelihood. In the case of this unigram and bigram data, I take d_total < -15. So when our recursion will not keep running recursion with low likelihood segment and thus save the running time.