HW1 Word Segmentation Report

Group BlackRice

2016/09/27

1 Group Info

Group Member	SFU mail
April Wang	ayw7@sfu.ca
Jiahao Ke	jiahaok@sfu.ca
Yu Tang	yta47@sfu.ca
Ruoxin Zhou	ruoxinz@sfu.ca

2 General Idea

In this project, we present a solution to segment Chinese words and our solution finally hits 93.361 on Leaderboard.

We use some optimizations listed below to improve the naive segmentor based on the pseudo-code provided with HW 1 that uses unigram probabilities:

- Using the bigram model to score word segmentation candidates.
- Adding Good-Turing Algorithm to smooth probability data used by bigram and unigram model
- Using Jelinek-Mercer Smoothing to improve the bigram model
- Using Back-off Smoothing to improve accuracy
- Merging all the digit numbers appeared continuously in a line
- Trying to recognize proper noun that is not collected in the dictionary but appearing more than twice.
- Trying to combine single word that is not collected in the dictionary but each single word has a very late prabability to be single in the sentance

For Jelinek-Mercer Smoothing, we write a bash script to test how different value of lambda would effect accuracy of final result. We get lambda = 0.17 for the best result.

3 Solution

3.1 Bigram model

Our language model combines the bigram language model over Chinese words. The input is a sequence of Chinese characters (without word boundaries): $c_0, ..., c_n$.

Then define a word as a sequence of characters: w_i^j is a word that spans from character i to character j. So one possible word sequence is $w_0^a w_{a+1}^b w_{b+1}^n$.

Score this sequence using Jelinek-Mercer Smoothed bigram probabilities.

$$\arg\max_{w_0^i w_{i+1}^j, \ldots, w_{n-k}^n} P_{JM}(w_0^i \mid start) * P_{JM}(w_{i+1}^j \mid w_0^i) * \ldots * P_{JM}(w_{n-k+1}^n \mid w_{n-g}^{n-k})$$

3.2 Jelinek-Mercer Smoothing

In order to get the best Jelinek-Mercer Smoothing, different λ should be tried to get the score.

$$P_{JM}(w_i \mid w_{i-1}) = \lambda P_b(w_i \mid w_{i-1}) + (1 - \lambda) P_w(w_i)$$

3.3 Good-Turing Smoothing

The bigram probability P_b can be constructed using data in count_2w.txt, while P_w can be constructed using data in count_1w.txt.

In order to deal with events that have been observed zero times, we use Good-Turing Smoothing to smooth data in count_1w.txt and count_2w.txt.

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

3.4 Merging digit numbers

For merging all the digit numbers, we simply go through the lines that has been segmented and whenever digit numbers appearing continuously, we combine them together.

3.5 Recognizing proper noun

For recognizing proper noun, we notice that the testing data is made of by pieces of news. For each news, there would likely be proper noun that appeared more than twice or three times. So for each segmented paragraph, we find those sets of words, while the length of each word is usually one but those sets of words appear continuously for more than twice. We consider those sets of words are likely to be proper noun and we combine them together.