CS571 Nature Language Processing
Homework 1 Report: Word Segmentation

Zhexiong Liu

1 Problem Description

This project aims to write a program that can seperate the Hashtags, strings that do not include whitespaces, into a list of tokens representing their the most likely sequences. For example, input Hashtag "#helloworld" and return ["hello", "world"].

2 Methods

2.1 Unigram Approach

Given a unigram dictionary, the probability $P(x_i)$ of world w_i is computed by $P(x_i) = \frac{C(x_i)}{N}$, where N is the number of the whole worlds in the dictionary. In addition, Laplace smoothing is implemented in the algorithm to alternatively represent the probability of an unseen world. In the world segmentation algorithm, depth first algorithm is implemented to break down given strings. For example, string "hello" can be searched by the order "h","he","hel","hell","hello"... This algorithm, somehow, achieves satisfactory results while segmenting strings with length less than 17 (experiment result); however, it cannot separate longer strings within a minutes.

2.2 Bigram Approach

As for bigram dictionary, the probability $P(x_{i+1}|x+i)$ of word x_{i+1} while word x_i existing is computed by $P(x_{i+1}|x+i) = \frac{C(x_i,x_{i+1})}{C(x_i)}$. Note that "add one" smooth incorporated with back-off technique is implemented in the algorithm. In other words, the unknown probability of a bigram model can be found in the unigram probability, in which the parameter β in the formula is replaced by 1. This "add one" revision slightly reduces the running time but with satisfactory performance. Besides, in word segmentation algorithm, a dynamic programming idea is implemented while computing the probability of a divided string, in which the algorithm remembers the updated probability in order to eliminate repeated computation of probability. This approach greatly reduces running time; thus it is capable of breaking down long strings.

3 Experiments

In the experiment part, two approaches with corresponding tracks are implemented and compared with each other. For unigram approach, it can separate short strings with satisfactory accuracy, but cannot easily separate long strings since this hard code has relatively high time complexity. Note that "#helloworld" cannot be separated, but "#helloworlds" can in approach 1. Regarding bigram approach, the dynamic programming is assembled into the segmentation algorithm, which improves the efficiency of breaking down long strings. Actually, a string with length of 76 can be separated within one second in bigram algorithm instead of several minutes in unigram algorithm. However, bigram algorithm is also unable to separate "#helloword" into "hello" and "world". This could be caused by the high frequency of string "helloworld" in unigram dataset. Another attempt to improve the performance of bigram algorithm is to utilize discount smooth technology. This test failed when computing the term $P_d(x_?|x_i)$ in the formula as it costs much time in computation. Instead, a simple "add one" smooth can also archive satisfactory result.

4 Problems

These algorithms cannot separate short strings with high frequency (such as "#helloword") since unigram dataset includes a exactly "helloword" string but bigram does not. Therefore, these two algorithms both likely contain a string as a whole if it shows in unigram. Meanwhile, unigram algorithm cannot efficiently separate long strings because only dfs algorithm cannot greatly reduce the computation complexity of the segmentation algorithm. In addition, Some capitalized words in bigram dataset could affect the overall performance, which should be changed to lowercases.