Homework 1 - CMPT413/713

September 28, 2021

Team Arceus Members: Shan Ying Yang, Shashank Thanalapati, Rylen Sampson

Task Summary

Implementing the provided baseline algorithm in Python to restore missing word boundaries, and word segmentation in the Chinese Language. In extension, implementing a second step to improve the accuracy and the F-score.

Methods - Unigram

1. **Baseline algorithm:** Implementing the iterative segmenter algorithm given in the homework page was the first added on top of the default solution. This is a simple Unigram model, that traverses through the entire corpus to find sequences that pass. On top of which we added the long word character limit, just to ensure we get the best base score of **0.88**.

2. **Additive/Laplace Smoothing:** It's a simple smoothing technique that works for basic models (such as this one), which adds one to the unigram probabilities thereby eliminating 0-denominators, giving an advantage to lower chances and while maintaining similar (or higher) accuracy with the outliers.

Overall F-score achieved is **0.93**.

Methods - Bigram

1. **Implementing Bigram:** A bigram model is when we take the 2 tokens out of the corpus instead of just one individually. We then take the conditional probability of a token given the preceding one. It is reccomended to use bigrams (or n-grams where n¿1) when we have a large enough corpus that the probability of 2 adjacent words occurring next to each other is equivalent or very close to that of the word individually. Using just bigrams, we got a score of **0.81**.

2. **Backoff:** The aforementioned bigram score is lower than that of the unigram score, even though it's supposed to increase it. The main reason being, even though our corpus is large, sometimes predictions of individual unigrams are higher than it's combined bigram conditional probabilities.In those cases, we use backoff smoothing where the unigram probability is prioritized rather than the grouped tokens. Implementing this boosted our score to **0.90**.

3. **Linear interpolation:** In an attempt to weight the predictive power of the unigram or bigram sequences being chosen, we implemented Jelinek-Mercer smoothing through the linear interpolation form. By varying the lambda parameter (the final one chosen was $\lambda = 0.9$) we obtained a score of **0.92** on the *dev* dataset.

Results

Method results

Approach	F1-score
Unigram (baseline)	0.88
Unigram (Laplace smoothing)	0.93
Bigram (baseline)	0.81
Bigram (Backoff)	0.90
Bigram (Linear interpolation)	0.92

Table 1: Shows the best obtained F1-score for each implemented method.

For unigram models, the one including Laplace smoothing achieves the overall highest score. Amongst the bigram models, the one incorporating Linear interpolation smoothing achieves the highest score and a close second score to our overall highest.

Linear interpolation results

λ	F1-score
0.8	0.81
0.5	0.84
0.3	0.85
0.1	0.92

Table 2: The effect of varying λ in JM linear interpolation smoothing.

This results in an F1-score of **0.92**.

Contributions

Everyone: We all attempted and implemented the unigram and baseline bigram model ourselves.

Shan Ying Yang: Tested additive smoothing for unigram and bigram to enhance F-Score.

Shashank Thanalapati: Implemented linear interpolation and backoff smoothing for unigram and bigrams attempting to increase F-score.

Rylen Sampson: Implemented JM smoothing and tested various λ values to obtain the best F1-score on the development data.