Assignment 1 Report

September 18, 2019

1 Text Preprocessing

For preprocessing, there is not anything fancy. By deleting space and punctuations, I tokenized all text files and turned them into list of tokens. The tokenization is followed by a function called *lower()*, which transforms every characters into their lowercase form. In order to preserve the context, I did not use lemmatizer or stemmer. Because all models in this assignment are built based on letters, it is necessary to keep the original context as much as possible and collect all possibilities.

2 Model Implementation

Before introducing models, I would like to mention that the start tag <s> and the end tag </s> are added in each word. For example, if the original word W = cat, with w_1 = c, w_2 = a, and w_3 = t, then the new notation will be w_0 = <s>, w_1 = c, w_2 = a, w_3 = t, and w_4 = </s>.

2.1 Unigram

The implementation of unigram only follows the formula from textbook:

$$P(w_i) = \frac{C(w_i)}{Total\ count\ of\ words}$$

2.2 Bigram

The bigram also uses the formula from textbook:

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{\sum_w C(w_{i-1}w)}$$

The bigram probability calculation usually starts with w_2, if there is no "<s>" before w_1. However, since I added <s> and </s> in my bigram model, the probability of w_1 will be $\frac{C(w_0w_1)}{\sum_w C(w_0w)}$. Take the word W = cat from the beginning of this part as an example, the full list of bigram probabilities are:

$$P(w_1) = \frac{C(w_0w_1)}{\sum_w C(w_0w)} \quad P(w_2) = \frac{C(w_1w_2)}{\sum_w C(w_1w)}$$
$$P(w_3) = \frac{C(w_2w_3)}{\sum_w C(w_2w)} \quad P(w_4) = \frac{C(w_3w_4)}{\sum_w C(w_3w)}$$

where w_0 and w_4 are special tags. In my code, it will loop through the tokenized text twice: the first loop is to record counts for patterns with three words and two words, and the second loop is to calculate probabilities of each pattern using counts recorded in the previous loop.

2.3 Trigram (No smoothing)

Usually the trigram probability calculation starts with w_3 due to the same reason that is mentioned in the bigram subsection, but since I added "<s>" and "</s>", the calculation will start from w_2 to w_{n+1}, in which $w_{n+1} =$ </s>. It also takes two loops, one for storing pattern counts and one for calculating probabilities.

2.4 Trigram (Laplace smoothing)

The laplace smoothing takes following fomula:

$$P_{Laplace}(w_i) = \frac{c_i + 1}{N + V}$$

In this assignment, N is taken from the pattern countsthe, and V is taken from the unigram, that $V = len(Unigram)$. Notice that this will also include two tags: "<s>" and "</s>".

2.5 Trigram (Backoff)

In the traditional Katz backoff algorithm, if the N-gram we need has zero counts, we approximate it by backing off to the $(N-1)$-gram with discounting and a function α. With discounting and α, it will rearrange the possibilities so that $\sum_i P(w_i|w_j w_k) = 1$.

In my implementation, I simplified the discounting part and α by ripping off the backoff weight function α and replacing the discounting with a variable λ that is between 0 and 1. If the trigram has zero count, it will backoff to the bigram, and the probability retrieved from bigram will be multiplied by λ. If the bigram still has zero count, it will backoff to the unigram, and the probability retrieved from unigram will be multiplied by λ^{10}. Since "<s>" will never appear after some letters (it only appears in the beginning of a word), the model will ignore the case of $(w_{i-2}, w_{i-1}, < s >)$ The simplification brings out a shortcoming: without the weight function, and with the exclusion of "<s>", there is no guarantee that the total probability of a given context could be greater than 1.

2.6 Trigram (Linear interpolation)

According to the assignment description, all λs in this model have same weight. Since $\sum_i \lambda_i = 1$, the value of all λs are $\frac{1}{3}$. In the trigram model, we have

$$\begin{aligned} P(w_i|w_{i-2}w_{i-1}) &= \lambda P(w_i|w_{i-2}w_{i-1}) + \lambda P(w_i|w_{i-1}) + \lambda P(w_i)) \\ &= \frac{P(w_i|w_{i-2}w_{i-1})}{3} + \frac{P(w_i|w_{i-1})}{3} + \frac{P(w_i)}{3} \\ &= \frac{P(w_i|w_{i-2}w_{i-1}) + P(w_i|w_{i-1}) + P(w_i)}{3} \end{aligned}$$

In this homework assignment, there might be some float rounding errors, so the $\sum_i \lambda_i$ might not be exactly 1.

3 Trigram Models Check

Here are some excerpts of my trigram models. The first single letter is the w_i, and those tuples are the context (w_{i-2}, w_{i-1}). The probability is $P(w_i|w_{i-2}w_{i-1})$.

3.1 Trigram (No smoothing)

```
e:
('<s>', 'r') 0.7773399014778325
('t', 'h') 0.6603921568627451
('<s>', 's') 0.1330671989354624
('<s>', 'd') 0.4328571428571429
('a', 'r') 0.272108843537415
('u', 'm') 0.24
```

3.2 Trigram (Laplace)

```
<s>:
('<s>', 'r') 0.0009541984732824427
('r', 'e') 0.000434593654932638
('e', 's') 0.0006688963210702341
('s', 'u') 0.0029850746268656717
('u', 'm') 0.007518796992481203
```

3.3 Trigram (Backoff)

```
</s>:
('<s>', 'r') 0.15012256403971075
('r', 'e') 0.2760141093474427
('e', 's') 0.48632010943912446
('s', 'u') 0.03155826558265583
('u', 'm') 0.33
```

3.4 Trigram (Linear Interpolation)

```
<s>:
('<s>', 'r') 0.06844977678278166
('t', 'h') 0.06844977678278166
('<s>', 's') 0.06844977678278166
('<s>', 'd') 0.06844977678278166
('a', 'r') 0.06844977678278166
('u', 'm') 0.06844977678278166
```

3.5 Validation Check

To check that the sum of probability for a fixed context is 1, I created a Google Spreadsheet. We may notice that the sum of probability for a fixed context in the backoff and lineaer interpolation models is not 1. This could because of:

1) Improper method while processing the tag "<s>"

2) Float rounding errors

For more details, please check the Google Spreadsheet above.

4 Perplexity

There are 100 lines in the test file, and if we could check the test file, then we will find that it is obviously written in English. However, in many cases we are not allowed to check the test file, or our insufficiency of knowledge may not lead us to a clear conclusion. As a result, we need to test models in all languages and calculate their perplexity scores. To calculate the overall perplexity for each model, I calculate the perplexity for each sentence, them added them up. For the perplexity of each sentence, I used the formula from the textbook:

$$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}} = (\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-2}w_{i-1})})^{-\frac{1}{N}}$$

The url for the full perplexity test result is here:
https://docs.google.com/spreadsheets/d/1HNmbXtnLlbsuOJHS2gx_p33wGs8LjTTfP6Cu_dXv1fc/edit?usp=sharing
According to the result, the perplexity results of unigrams and bigrams are not very good. Among all trigram models, English models have the lowest perplexity (of course), and among all models, the English trigram model with backoff has the lowest perplexity.

The reason that why the English models have the lowest perplexity could be:

1) The test file does not contain many named entities, so these English language models will probably not get influenced by the loan words. Although there are few loan words like "Al-Qaeda" and "Musharaf", since they are Arabic-originated, none of the three languages share the similarity on alphabetic arranging. As a result, all models, regardless of languages, have high perplexity on these loan words.

2) German has vowels with umlauts, and Spanish also has vowels with accent marks. These letters could lower the probability of other letters that only appear in English. Take the spreadsheet in section 3.5 as an example. Without smoothing, the probability of "thá" and "thä" is 0. This is because:

   – There is no "á" or "ä" in English, and

   – Those letters didn't appear in the English training text

   Since $\sum P(w|w_{i-2}w_{i-1})$ is always 1, adding those "noise" letters (those non-English vowels) could reduce the probability of generating other English letters. The higher the probability a non-English vowels has, the lower the probability other letters will have, which will make the language model less like a English language model.

5 Research Task

5.1 From Theory

Since the English trigram model with backoff has the lowest perplexity score, I decided to improve the backoff model. According to the 2nd edition of the textbook, there is a language model called Katz backoff. In Katz backoff model, if the N-gram we need has zero counts, we could backoff to the $(N-1)$-gram, $(N-2)$-gram, ..., until we reach a model that has some counts. In trigram, we could calculate in this way:

$$P_{katz}(z|(x,y)) = \begin{cases} P^{'}(z|(x,y)), & \text{if } C(x,y,z) > 0 \\ \alpha(x,y)P_{katz}(z|y), & \text{elif } C(x,y) > 0 \\ P^{'}(z), & \text{otherwise.} \end{cases}$$

For $P_{katz}(z|y)$, we have:

$$P_{katz}(z|y) = \begin{cases} P^{'}(z|y), & \text{if } C(y,z) > 0 \\ \alpha(y)P^{'}(z), & \text{otherwise.} \end{cases}$$

In these formulas, the $\alpha(w)$ is the backoff weights, and the probability $P^{'}$ is the discounted probability. The α function will pass the leftover probability mass to the lower-order N-grams. In order to calculate α, we need to define $\beta(w)$ first.

$\beta(w)$ is a function of the $(N-1)$-gram context that represent the total left over probability mass. In the trigram model that backoffs to the bigram model, the β should look like this:

$$\beta(x,y) = 1 - \sum_{z:C(x,y,z)>0} P^{'}(z|(x,y))$$

Since β represents the total probability mass, we need to divide this total probability mass. Each individual bigram will only get a fraction of this mass, so we need to normalize β by the total probability of all the bigrams that begin some trigram that has zero count:

$$\alpha(x,y) = \frac{\beta(x,y)}{\sum_{z:C(x,y,z)=0} P_{katz}(z|y)}$$

5.2 Implementation

In my implementation, I used a variable λ to represent the discounted multiplier such that $P^{'}(w) = \lambda P(w)$. In order to calculate with more convenience, I also built two inverted indices, one is for the bigram, and the other is for the trigram. First, the program will calculate all $P_{katz}(z|y)$, and store them in a dictionary. For each context y in the inverted index dictionary of bigram, there will be an $\alpha(y)$. Then, the program will check if $C(y,z) > 0$ and it will follow the conditional equation above. After calculating all $P_{katz}(z|y)$, we could calculate $P_{katz}(z|(x,y))$ similarly.