Report

Last update: April 14, 2019

1 The Language Model

The language model we build is the Trigram language model with Stupid Backoff smoothing. The Trigram language model is a type of n-gram language model which is a probabilistic distribution over sequences of words. It is under the assumption that the probability of obseving the i^{th} word can be approximated by the probality of observing it considering the context history of the preceding 2 words. Given a sentence $[w_1, w_2, w_3, ..., w_m]$, the probability of observing this sentence based on the trigram model is approximated as

$$P(w_1, ..., w_m) = \prod_{i=1}^{m} P(w_i|w_1, ..., w_{i-1}) \approx \prod_{i=1}^{m} P(w_i|w_{i-2}, w_{i-1}) \tag{1}$$

where the conditional probability can be calculated from frequency counts:

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})} \tag{2}$$

However, we can find that we would often encounter with many trigrams that have not been seen before. From which, we can infer that we shouldn't set n of n-gram language model too large, for that the longer sub-sequences are, the smaller opportunities for them to be observed. If a trigram is not seen before, then the conditional probalitiy would be zero based on the equation above, leading to the zero probability of observing a sentence. Therefore, some form of smoothing is necessary. In this work, the Stupid Backoff smoothing is applied to calculating conditional probabilities. Stupid Backoff smoothing is conducted by going back to n-1 gram level to calculate the probabilities when encountering a word with zero conditional probability. Thus, the conditional probability using Stupid Backoff smoothing is as below:

$$P(w_i|w_{i-2}, w_{i-1}) = \begin{cases} \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})} & if count(w_{i-2}, w_{i-1}, w_i) > 0 \\ 0.4 * P(w_i|w_{i-1}) & otherwise \end{cases}$$

$$P(w_i|w_{i-1}) = \begin{cases} \frac{count(w_{i-1}, w_i)}{count(w_{i-1})} & if count(w_{i-1}, w_i) > 0 \\ 0.4 * P(w_i) & otherwise \end{cases}$$

$$P(w_i) = \begin{cases} \frac{count(w_i)}{N} & if count(w_i) > 0 \\ 0.000001 & otherwise \end{cases}$$

where N is the count of words in the whole corpus. The last two equations can be regarded as the Bigram and Unigram model respectively. Take the last equation for an example, if we encounter with the case that the word has not been seen before, the probability of this word is then set to be a very small value (as provided in lm.py), i.e backoff = 0.000001.

Furthermore, the sampler used for sentence generation is the similar with the sampler provided in lm.py. The difference is that when using Trigram, the previous words that are took into account is the preivous two words, instead of no previous words when using Unigram.

2 Analysis on In-Domain Text

We evaluate the smoothed Trigram model by comparing with the smoothed Unigram model and the smoothed Bigram model which are using the same smoothing method. All three models are trained and tested on the on the three datasets, namely Brown, Reuters and Gutenberg.

2.1 Empirical Evaluation

We use the training sets to train the models and calculate the perplexities on the test sets. The result is presented in the following table. From the table, we can find that both the smoothed Bigram and Trigram models have better performance than the Unigram model. The Trigram model is better than the Bigram model on Reuter dataset, however the opposite on the other two datasets.

Table 1: Empirical Evaluation of In-Domain Performance

	Brown	Reuters	Gutenberg
Unigram	1604.198	1500.695	1005.79
Bigram	642.768	198.092	313.051
Trigram	1018.417	178.707	355.787

2.2 Qualitative Analysis

Here we present one samples (the maxlength is set as 20) for each trained model on the three datasets, using the the same prefix ‘It is‘. We also calcualte the log-probablities of the generated sample sentences using the corresponding logprob_sentence() functions.

Databse: Brown

1. Unigram Sample|: -246.728 |It is most Laos of for meaning found then will talk the marched adjusted You better had of not so sensitivity hard exert
2. Bigram Sample|: -131.371 |It is By tenements Only violation teachers qualify Falls sit Dirty agers message corporeal multiple Franklin symmetry strategic Gazing exhibits antelope

3. Trigram Sample|: -56.791 |It is Genevieve giants piezoelectric Twentieth Feds impatient Boys hey Pall Packing How drinker Augusta planned Sturley domains imports steroids saturation

Databse: Reuters

1. Unigram Sample|: -246.728 |It is most Laos of for meaning found then will talk the marched adjusted You better had of not so sensitivity hard exert
2. Bigram Sample|: -131.371 |It is By tenements Only violation teachers qualify Falls sit Dirty agers message corporeal multiple Franklin symmetry strategic Gazing exhibits antelope
3. Trigram Sample|: -56.791 |It is Genevieve giants piezoelectric Twentieth Feds impatient Boys hey Pall Packing How drinker Augusta planned Sturley domains imports steroids saturation

Databse: Gutenberg

1. Unigram Sample|: -246.728 |It is most Laos of for meaning found then will talk the marched adjusted You better had of not so sensitivity hard exert
2. Bigram Sample|: -131.371 |It is By tenements Only violation teachers qualify Falls sit Dirty agers message corporeal multiple Franklin symmetry strategic Gazing exhibits antelope
3. Trigram Sample|: -56.791 |It is Genevieve giants piezoelectric Twentieth Feds impatient Boys hey Pall Packing How drinker Augusta planned Sturley domains imports steroids saturation

3 Analysis on Out-of-Domain Text

Here we compare the out-of-domain performance for the three smoothed models.

3.1 Empirical Evaluation

Here we present perplexity of all three models on all three domains(as computed in data.py). All values are calculated on the test dataset in each dataset. From the three tables (Table 2, 3, 4), we can find the following conclusions. 1) From the second columns of the three tables, namely testing on the Reuters corpus, the out-of-domain performance of the Unigram model is the best, then comes the Bigram model. 2) Testing on both the test sets of the Brown corpus and the Gutenberg corpus, the out-of-domain performance of the Bigram model is the best, then comes the Unigram model. 3) The out-of-domain performance of the Trigram model is the worst on all three datasets. We think the cause of these conclusions is that the trigrams of one domain can be much more rarely seen on another different domain than the bigrams or unigrams.

Table 2: Out-of-Domain Performance of the Smoothed Unigram Model

	Brown	Reuters	Gutenberg
Brown	1604.2	6736.6	1762.006
Reuters	3865.16	1500.69	4887.47
Gutenberg	2626.05	12392.5	1005.79

Table 3: Out-of-Domain Performance of the Smoothed Bigram Model

	Brown	Reuters	Gutenberg
Brown	642.768	5528.27	1181.21
Reuters	2811.82	198.092	5101.5
Gutenberg	1754.95	15224.7	313.051

Table 4: Out-of-Domain Performance of the Smoothed Trigram Model

	Brown	Reuters	Gutenberg
Brown	1018.42	11419.7	2207.69
Reuters	5624.35	178.707	11130.7
Gutenberg	3207.42	33837.1	335.787

3.2 Qualitative Analysis

As we analyzed before, the reason why the Unigram and Bigram models generalize better than the Trigram model is that the trigrams from one domain might be rarely seen in another domain. We randomly selected a sample sentence from the Gutenberg test set. 'But the Martins occupied her thoughts good deal she had spent two very happy months with them and now loved to talk of the pleasures of her visit and describe the many comforts and wonders of the place'. We trained the models are on the Brown training set. Using these models, the log-probablity of observing the sentence are -386.654, -344.619 and -378.385, respectively. The occurrence of the unigrams, bigrams and trigrams are 37, 24 and 0, respectively, which proves our claim.

4 Adaptation

Firstly, we randomly select a small set (1%) from the Gutenberg training set. Then we duplicate the small set and insert into the Brown training set. Finally, we use the new training set to train the Trigram model and test in on the Gutenberg test set. We set the number of time (k) that we duplicate the small set as ranging from 1 to 19 with step length 3. The result is presented in the table below. From Table 1, the in-domain performance is perplexity = 355.787. From the table, we find that our adaptation improves the out-of-domain performance (perplexity = 2207.69 from Table 4), which reaches the best when k=10, but still worse than the in-domain performance.

Table 5: Out-of-Domain Performance after Adaptation

k	1	4	7	10	13	16	19
Brown	1605.311	1438.454	1383.449	1381.993	1411.805	1460.972	1522.409