CSE 256 Assignment 1

Chowdhury, Ranak Roy (A53317421)

October 16, 2019

1 Language Model Implementation

In this section, the details of the language model, smoothing technique and hyperparameter tuning have been discussed.

1.1 Description of the Model

The trigram language model has been implemented for the assignment. A trigram language model assumes that the current word is dependent on the preceding two words of the sentence. The trigram model consists of a finite set of vocabulary, denoted by V, and a parameter $q(w|u,v)$, where w is the current word, v is the word preceding w and u is the word preceding v for each trigram. u, v, and w are such that $w \in V \cup EOS$ and $u, v \in V \cup *$. EOS refers to the end of sentence marker. The value for $q(w|u,v)$ can be interpreted as the probability of seeing the word w immediately after the words u and v. This can be computed as

$$q(w|u,v) = \frac{count(u,v,w)}{count(u,v)} \tag{1}$$

where $count(u,v,w)$ is the number of times the words u, v and w appear consecutively and $count(u,v)$ is the number of times the words u and v appear consecutively in the training corpus. For any sentence $x_1...x_n$ where $x_i \in V$ for $i = 1...(n-1)$, and $x_n = EOS$, the probability of the sentence under the trigram language model is

$$p(x_1...x_n) = \prod_{i=1}^{n} q(x_i|x_{i-2}, x_{i-1}) \tag{2}$$

where we define $x_0 = x_{-1} = *$ The parameters satisfy the constraint for any trigram u, v and w,

$$q(w|u,v) \geq 0 \tag{3}$$

and similarly for any bigram, u and v where the current word depends just on the immediately preceding word,

$$\sum_{w \in V \cup EOS} q(w|u,v) = 1 \tag{4}$$

Each of w, u and v can take $|V|$ possible values. So there are around $|V|^3$ parameters of this model.

1.2 Description of the smoothing method

With text data, there are likely to be pairs of labels and words that never appear in the training set, leaving $q(w|u,v) = 0$. This is undesirable, because it imposes high variance: depending on what data happens to be in the training set, we could get vastly different classification rules. One solution is to smooth the probabilities, by adding a "pseudocount" of α to each count, and then normalizing. $0 < \alpha < 1$.This is known as **Laplace smoothing**, which has been implemented in the assignment for the trigram model. So now the expression for $q(w|u,v)$ becomes

$$q(w|u,v) = \frac{count(u,v,w) + \alpha}{count(u,v) + \alpha|V|} \tag{5}$$

The pseudocount α is a hyperparameter, because it controls the form of the log-likelihood function. This drives the estimation of α. Smoothing reduces variance, but it imposes a bias by moving away from the maximum likelihood estimate. In this case, the bias points towards uniform probabilities. We use our held-out data, i.e. the .dev files, to tune this hyperparameter.

To account for words in the held-out and the test set that have not been encountered during training, the "unknown" symbol UNK has been used. A parameter f, known as the threshold frequency, has been used to filter out words in the training data that appear less than f times. Afterwards, the training corpus and vocabulary have been updated by replacing all those words appearing less than f times by 'UNK'. The $count(u,v,w)$ and $count(u,v)$ have been built based on the updated corpus. Therefore, whenever a word in the held-out or test set appeared that was not seen during training or has been replaced by 'UNK', it was replaced by the word 'UNK'. As a result three possible cases occurred:

- The triplet, u, v and w in the held-out or test set exist in the trained trigram model. This also implies that the corresponding bigram pair u, v also exist in the trained bigram model. Therefore, the expression for $q(w|u,v)$ becomes:

$$q(w|u,v) = \frac{count(u,v,w)}{count(u,v)} \tag{6}$$

- Otherwise, if the triplet, u, v and w was not seen in the trained trigram model, but the corresponding bigram pair u and v exist in the trained bigram model, then the expression for $q(w|u,v)$ becomes:

$$q(w|u,v) = \frac{\alpha}{count(u,v) + \alpha|V|} \tag{7}$$

- If neither of the trigram, u, v and w nor the bigram pair, u and v was seen during training, then the expression for $q(w|u,v)$ becomes:

$$q(w|u,v) = \frac{\alpha}{\alpha|V|} = \frac{1}{|V|} \tag{8}$$

1.3 Hyperparameter tuning and data used for tuning

Table 1: Perplexity values of the trigram model with Laplace smoothing on the held-out data of the three datasets for different combination of the hyperparameters (threshold frequency, f and the Laplace smoothing, α)

Dataset	$f = 1$, $\alpha = 0.1$	$f = 1$, $\alpha = 0.9$	$f = 4$, $\alpha = 0.1$	$f = 4$, $\alpha = 0.9$
Brown	2892.78	2823.59	1319.96	1269.56
Reuters	401.39	394.771	272.625	266.168
Gutenberg	1073.06	1053.16	686.868	667.569

The dev split of the data contains the held-out dataset. A grid search was performed on the two hyperparameters of the model, namely the threshold frequency, f and the Laplace smoothing parameter, α. The results in Table 1 show the perplexity values of the Trigram Language Model with Laplace smoothing on the held-out set of the Brown, Reuters and the Gutenberg dataset. As seen in Table 1, the perplexity values decrease as we increase the values of threshold frequency, f and Laplace smoothing parameter, α. The lower the perplexity value, the better the hyperparameter settings. So $f = 4$, $\alpha = 0.9$ is the best possible combination of the hyperparameters, giving the lowest perplexity values for the held-out data of the three datasets. We are training our model and updating our parameters on the training set. So when a trained model performs well on unseen data (like the held-out data), it means that the model is generalizing well. So from the experimental results we find that as we increase the value of the threshold frequency, f and the Laplace smoothing parameter, α, our model generalizes better, thereby giving lower values of perplexity. It seems evident because when we increase f, we are replacing more words with the 'UNK' token. Therefore, our model is not memorizing these rare and unseen words, and generalizing better to unseen corpus. Similarly, when we increase α, we are doing more smoothing. Therefore, the model generalizes better. However, we should not increase f and α to an extent where the model doesn't learn anything and generates a constant output regardless of the input corpus. Hence, we strive to achieve a well-balanced model by choosing a suitable value for the parameters, f and α. Since $f = 4$ and $\alpha = 0.9$ gives the lowest perplexity (best performance) on the held-out data, we use these values for the remaining set of experiments.

2 Analysis of In-Domain Text

In this section, the performance of a language model build on a training set has been evaluated on its corresponding test set. The perplexity values of the trained models on their respective tests sets have been reported for both the trigram and the unigram language model. The perplexity values of the trigram model on the test set have been shown for various hyperparameter combinations. Examples of generated sample sentences for each of the trained trigram language model have also been shown.

2.1 Perplexity Values of the trigram model for in-domain text on the test set

Table 2 shows the perplexity values of the trigram model with Laplace smoothing ($f = 4$ and $\alpha = 0.9$) on the test set of the three datasets.

2.2 Comparison with Unigram Model

Table 3 shows the perplexity values of the unigram model on the test set of the three datasets.

Table 2: Perplexity values of the trigram model with Laplace smoothing ($f = 4$ and $\alpha = 0.9$) on the test set of the three datasets.

Dataset	Trigram
Brown	1318.63
Reuters	274.987
Gutenberg	672.629

Table 3: Perplexity values of the unigram model on the test set of the three datasets.

Dataset	Unigram
Brown	1604.2
Reuters	1500.69
Gutenberg	1005.79

2.3 Discussion of comparison of observations

The unigram model used was the one that was built in the assignment. The hyperparameter values used were $f = 4$ and $\alpha = 0.9$ because they gave the lowest perplexity (best performance) on the held-out data. As shown in Table 2, the perplexity values of the trigram model on the test set is higher than the corresponding values on the held-out set in the last column of Table 1. The perplexity values of the trigram model for each test dataset is lower than the corresponding unigram model. This seems obvious because a trigram model considers the context by taking the latest two words into consideration while the unigram model does not take any contextual information into consideration. Therefore, the trigram language model better fits the data than the corresponding unigram model, giving lower values of perplexity. Hence, the trigram model outperforms the unigram model.

2.4 Performance w.r.t hyperparameters

Table 4: Perplexity values of the trigram model with Laplace smoothing on the test set of the three datasets for different hyperparameter combinations (threshold frequency,f and Laplace smoothing α)

Dataset	$f = 1$, $\alpha = 0.1$	$f = 1$, $\alpha = 0.9$	$f = 4$, $\alpha = 0.1$	$f = 4$, $\alpha = 0.9$
Brown	2940.63	2870.8	1370.71	1318.63
Reuters	415.071	408.475	281.61	274.987
Gutenberg	1083.12	1063.39	691.929	672.629

The results in Table 4 shows the perplexity values of the Trigram Language Model with Laplace smoothing on the test set of the Brown, Reuters and the Gutenberg dataset. Table 4 shows a similar trend across the table like Table 1, i.e. the perplexity values decrease as we increase the values of threshold frequency, f and Laplace smoothing parameter, α. Hence, the model generalizes better with higher values of f and α. Besides, the perplexity values in Table 4 are all higher than the corresponding values in Table 1, as expected.

2.5 Examples of sampled sentences

The first sentence of every model was generated using a given prefix. The second sentence was generated without any prefix. The prefix used for the three datasets are as follows:

- **Brown Dataset:** the jury and the defense department
- **Reuters Dataset:** the traders in south asia
- **Gutenberg Dataset:** she rejoiced the glory but

The generated sample sentences for both with and without prefix are as follows:

- **Brown Dataset**
  - the jury and the defense department reacted sharply sherlock publication unk 17 reverend obscure interior opinions talk definition polished unk unk ringing what cox issues exchanged court
  - ll keep afloat idol louis meek convention is ethical crude dominate outsiders belonged prime unk designated ultimately super workable usefulness abide returning avoiding read serum polite cop

- **Reuters Dataset**
  - the traders in south asia diplomatic accepts 71 widening 954 aloha useful madagascar weapons true 287 assumes band unk board 638 drastic achieving drilling attend pratt equipment co unit
  - treasury detroit blamed cenergy 66 burdened fsb callebaut vs school 472 unk evaluating unitholders rumored 75p kinds transmission 400 plants tonnages mthly seven attracting unk golden unk
- **Gutenberg Dataset**
  - she rejoiced the glory but spread doted channel relent hireling abhor threats modest humane anybody everybody me distinct unk unk unk pivot readers whiteness unk writers pretty sociable grasp safe crash
  - you are redeem doubting grammar goliath unlucky balls flashes medical causing unk gallery reeling coal tightly indefinite harshly lily musicke trifling camillo unk fearful aforetime pear stun

For the Brown Dataset, when we used the prefix 'the jury and the defense department', the words that immediately followed were reacted sharply which makes sense, whereas when there was no prefix given, the sentence made no sense. Similarly for the Reuters Dataset, when we used the prefix 'the traders in south asia', we found words like accepts, weapons, equipment, co unit which are somewhat compatible with trading. However, when no prefix was given, there was no consistency across the sentence. Similarly for the Gutenberg Dataset, when we used the prefix 'she rejoiced the glory but', we found words with negative connotation like abhor and threats. However, when no prefix was given, there was no consistency across the sentence.

3 Analysis of Out-of-Domain Text

This section discusses the performance of a language model build on the training set of one domain on the test set of a different domain. The comparisons have been shown for both the trigram and the unigram language model. Reasons for the variation of perplexity values across different domains have also been discussed.

3.1 Perplexity values

Table 5 shows the perplexity values of the three trigram models on each of the test dataset ($f = 4$ and $\alpha = 0.9$).

Table 5: Perplexity values of the three trigram models on each of the test dataset ($f = 4$ and $\alpha = 0.9$)

	Brown Test Data	**Reuters Test Data**	**Gutenberg Test Data**
Brown Model	1318.63	2354.08	1691.72
Reuters Model	1468.57	274.987	1765.62
Gutenberg Model	1419.23	1917.58	672.629

3.2 Comparison with unigram model

Table 6 shows the perplexity values of the three unigram models on each of the test dataset.

Table 6: Perplexity values of the three unigram models on each of the test dataset

	Brown Test Data	**Reuters Test Data**	**Gutenberg Test Data**
Brown Model	1604.2	6736.6	1762.01
Reuters Model	3865.16	1500.69	4887.47
Gutenberg Model	2626.05	12392.5	1005.79

3.3 Discussion of comparison of observations

Table 5 shows that the test results for the in-domain data is much better (lower perplexity) than the corresponding results for the out-of-domain data. Model trained on data of a particular domain learns features of that domain and thereby performs better on the test data belonging to that domain. Since the features vary across domains, it is highly unlikely that a model built on a set of features will perform well on data that follows distribution of another set of features.

The results in Table 6 shows similar trend across the table compared to Table 5. The in-domain perplexity values on the test set is still lower than the the corresponding values for out-of-domain texts. However, the perplexity values of the trigram model is generally lower, and therefore better, than the corresponding values of the unigram model for both the

in-domain and out-of-domain texts. This is because english language follows certain grammatical structure and semantics. As a result of which, certain words will always appear close to one another. The trigram model is more complex and can consider this contextual information unlike the unigram model, which ignores all previously seen words. And therefore, the trigram model generalizes better to test data of both in-domain and out-of-domain texts, giving lower values of perplexity than the unigram model.

3.4 Comparison of performance on different corpora

Another interesting point to note across Table 1, 2 and 4 is that the Reuters Dataset trained on the trigram model with Laplace smoothing gives the lowest values of perplexity on both the held-out set and the test set, followed by the Gutenberg and the Brown Dataset. For the Reuters Dataset, the difference between the perplexity values of the trigram model with Laplace smoothing and the unigram model is much higher than the corresponding differences for the two other datasets in Table 2 and 3. One reason could be that the Reuters Dataset has a much smoother progression and flow of ideas across the text. Therefore, information which appear quite close also carry similar meaning and talk about the same topic. Hence, it prioritizes contextual information more than the other two datasets. Since, the trigram model looks to encode this context into its architecture, therefore, the trigram model gives much lower perplexity values for the Reuters Dataset than for the other two datasets. However, the unigram model does not consider any context and therefore, the perplexity value of the unigram model for the Reuters dataset in Table 3 is quite high compared to the other two datasets.

4 Adaptation

In this section, the trigram language model initially built on a particular domain has been adapted for a different domain. Then the trained model has been evaluated on its performance on the test set of the new domain. This section discusses the approach used and the comparison of the performance of the trained trigram model.

4.1 Description of Approach

For the adaptation task, the whole of Brown's train data was used. Then a fraction of Reuters' train data was infused into Brown's train data and the resulting dataset was shuffled. This merged data was then used to train the trigram language model. The performance of this language model was then measured by the perplexity of the model on the Reuters' test data.

4.2 Relevant Comparisons

Table 7: Perplexity values of the trigram model with Laplace smoothing on the Reuters' Test Data for different fractions of Reuters' Train Data infused into Brown's full Train Data ($f = 4$ and $\alpha = 0.9$)

Fraction of Reuters' Train Data infused into Brown's full Train Data	**Perplexity values on Reuters' Test Data**
0	2354.08
25%	827.12
50%	589.15
75%	455.78
100%	367.66

As the results in Table 7 display, when more of Reuters' train data was infused into Brown's train data, the performance on Reuters' test data improved as perplexity values decreased. From Table 2, we find that the trigram language model trained on the Reuters' train data gives a perplexity of 274.987 on Reuters' test data. And when we adapt Brown's trigram language model with Reuters' train data, the lowest perplexity we achieve is 367.66, which occurs when we infuse all of Reuters' train data into Brown's train data. So as the fraction of Reuters' data in the total training data increases, the model emphasizes more on the features of Reuters' data and less on that of Brown's data. Therefore, the performance on Reuters' test data improves with increasing fraction of Reuters' data in the training set.