Homework 2: Language Modeling

Seokchan Ahn
Department of Computer Science
University of California, Irvine
seokchaa@uci.edu

Abstract

A Language model is probabilistic modeling of language. It enables differentiating between a grammatically correct English sentence and a random sequence of words. In this work, I implement an N-gram language model, which expects the probability of the next token given the previous N tokens. I also find the best hyper-parameters for overall domains, evaluate the performance of the models trained with data from the three different domains and compare the results between the models.

1 Implement an N-gram Language Model

N-gram language models model the probability of the next token based on the previous $n-1$ tokens. Given the context, a sequence of $n-1$ tokens, the next token's probability is calculated by dividing the frequency of the n-gram by the frequency of the previous $n-1$ tokens.

1.1 Special Tokens

At the beginning of a sentence, there can be tokens that have less than $n-1$ tokens ahead. For that cases, I added $n-1$ padding tokens <p> to every sentence during training so that every token on the front could be a part of the n-gram.

I also added an 'end of sentence' token <eos> to the end of every sentence so that the language model can stop generating the next tokens by generating the <eos> token.

1.2 Smoothing

I implemented smoothing methods to ensure the language models have a non-zero probability for every token. For OOV words, I assigned the probability that is inversely proportional to the number of OOV words to prevent unknown tokens have a high probability. For n-grams with zero probabilities, I assigned a very small probability of 1e-6 to prevent its logarithm diverge.

1.2.1 Laplace Smoothing

Laplace Smoothing is a method that assigns a default count value for every n-gram. It is called add-1 smoothing when the default count value is one and add-λ smoothing when the value is λ.

1.2.2 Backoff

Backoff is a method that delegates to the lower-gram model, when given n-gram never occurs in the training data. For example, if the probability of an n-gram is zero, it uses (n-1)-gram model instead, and so on.

1.2.3 Interpolation

Interpolation is a method that combines all the 1 to n-gram models by using a weighted sum of the probabilities. The sum of the weights should be 1.

2 Hyper-parameter Search

I use grid-search to find the best hyper-parameters for each smoothing method. Grid-search is a commonly used method of performing a hyper-parameter search, which is simply trying all the hyper-parameter combinations in the specified range with equal intervals. I choose the best hyper-parameter based on the average of the dev set perplexities across the corpora. I use the Brown(Francis and Kucera, 1979), Gutenberg(Gerlach and Font-Clos, 2018), and Reuters(Rose et al., 2002) corpora for the training process.

2.1 Laplace Smoothing

I tried 1 to 4-gram language models with the lambda values in [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]. The best parameter found was 2-gram without smoothing, and the perplexity was 1397.18. Laplace smoothing did not help in terms of perplexity for $n = 1, 2, 3$.

Model	Unigram		Laplace		Backoff		Interpolation	
Dataset	Train	Test	Train	Test	Train	Test	Train	Test
brown	1513.8	1758.2	74.6	2977.9	1.8	386.0	336.1	2288.2
reuters	1466.9	1576.9	51.2	471.1	1.5	74.0	274.1	861.9
gutenberg	981.4	1035.8	91.9	785.4	1.6	144.8	300.3	901.9

Table 1: In-Domain Perplexity

Model	Unigram	Laplace	Backoff	Interpolation
brown	8826.4	34413.7	3728.1	17428.4
reuters	8589.1	63076.5	4190.7	23021.2
gutenberg	20860.1	116988.8	12300.0	49217.7

Table 2: Out-Domain Perplexity on Test Set

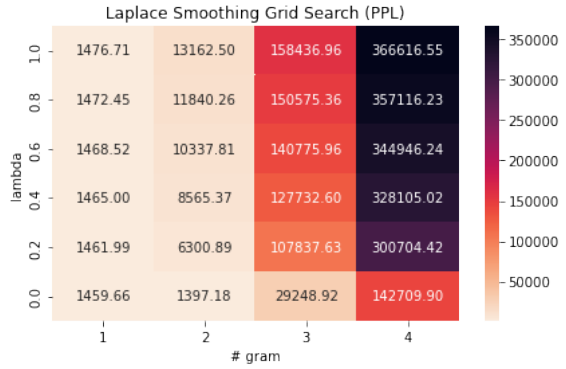

Figure 1: Laplace Smoothing Grid Search

2.2 Backoff

I tried 1 to 9-gram language models with the backoff strategy. The perplexity keeps decreasing as the n increases, but it was almost converged after $n = 4$. The best perplexity found was 201.0 at $n = 9$.

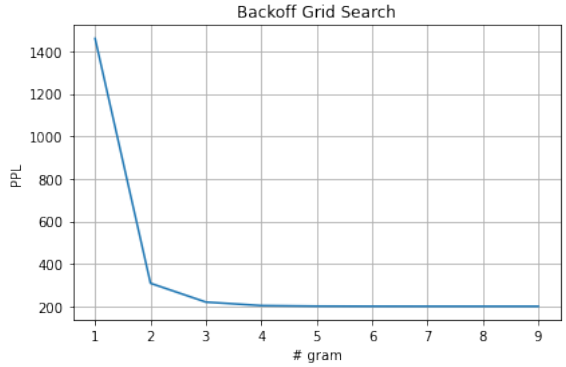

Figure 2: Backoff Grid Search

2.3 Interpolation

I conducted a grid search for the bigram language model only because there are too many cases with the larger n. The perplexity was found at the weight values are 0.5 for both bigram and unigram, and the value was 1349.0.

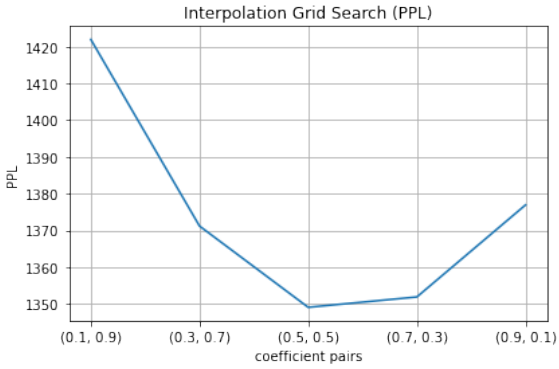

Figure 3: Interpolation Grid Search

3 Empirical Evaluation of Language Models

I compared the in-domain and out-domain perplexity of the unigram model and the models from the previous section with the best hyper-parameters.

3.1 In-Domain Perplexity

Table 1 shows the summary of in-domain perplexity values for each model and each dataset. All smoothing methods showed lower perplexity than the unigram model. The model using backoff was the best, and the model using interpolation was the worst in terms of perplexity. The perplexity difference between the train and test set was not significant for the unigram model, but it was much more extensive for the models with smoothing.

Model	Generated Sentence w/ Brown Model and Prefix "The event"
Unigram (n=1)	The event was book this negate then the Committee alleviation her
Laplace (n=2)	The event it seemed to general acceptance of existing library system
Backoff (n=9)	The event staged yearly by the who are twofold Commission Pullman
Interpolation (n=2)	The event for 76 reaches issue of life is continuing

Table 3: Generated Examples (Model-wise)

Domain	Generated Sentence w/ Backoff Model and Prefix "The event"
brown	The event staged yearly by the who are twofold Commission Pullman
reuters	The event in the Great MTBE in Breslube Enterprises of Toronto
gutenberg	The event has descended reproach two sisters seeing could wish

Table 4: Generated Examples (Domain-wise)

3.2 Out-of-Domain Perplexity

Table 2 shows the out-domain perplexity values for each model. Among those four models, the model using Laplace was the worst, and the model using backoff was the best. In particular, models using Laplace and interpolation were even worse than the unigram model.

4 Language Generation and Scoring

In this section, I generate some example sentences from each model and compare the characteristics of the output sentences.

4.1 Comparison Between Generated Examples with the Same Prefix

To compare the characteristics of each model, I fed the same prefix "The event" for every model and generated the following tokens.

4.1.1 Unigram vs. N-gram Models

First, I compared the outputs from the unigram model and n-gram models. As shown in Table 3, The sentence from the unigram model is grammatically wrong since it does not take account of the context at all. In contrast, the sentences generated by n-gram models are relatively more grammatical, not wholly correct though.

4.1.2 Difference Between Domains

I also compared the outputs from the models trained on different corpora. In Table 4, we can see the named entities such as 'Pullman' or 'MTBE' in the sentences generated by brown and reuters models, which contain news data. On the other hand, the sentence generated by gutenberg model seems less informative but more likely to be a sentence in a novel.

4.2 Sentence Scoring

4.2.1 Unigram vs. N-gram Models

Model	Perplexity
Unigram (n=1)	1123.56
Laplace (n=2)	894.75
Backoff (n=9)	135.42
Interpolation (n=2)	1002.65

Table 5: Perplexity Comparison for "we feel that in the future Fulton County should receive some portion of these available funds"

Model	Perplexity
Unigram (n=1)	1488.47
Laplace (n=2)	30630.00
Backoff (n=9)	1598.16
Interpolation (n=2)	6752.18

Table 6: Perplexity Comparison for "was book this negate then the Committee alleviation her"

I compared the models with a natural sentence and a random sequence of words, "we feel that in the future Fulton County should receive some portion of these available funds" and "was book this negate then the Committee alleviation her" for each.

Table 5 shows the perplexity values of a 'natural' sentence per model. The unigram model's perplexity is the highest because the sentence is treated as a bag-of-word in a unigram model. However, n-gram models give a higher probability for the common n-grams.

However, for the random sequence of words, the unigram model tends to get lower perplexity since n-grams in such sentences would be very rare.

4.2.2 Difference Between Domains

Model	Perplexity
brown	135.42
reuters	337.47
gutenberg	903.45

Table 7: Perplexity Comparison for "we feel that in the future Fulton County should receive some portion of these available funds"

Model	Perplexity
brown	158.97
reuters	399.01
gutenberg	93.09

Table 8: Perplexity Comparison for "How often we shall be going to see them , and they coming to see us !"

Table 7 compares perplexity values for each model for a sentence from brown test set. The brown model's perplexity was the lowest since it is from the same corpus, and reuters model was next because reuters corpus is from the news domain, which overlaps the brown corpus' domain. Since gutenberg corpus is from the literature domain, not from the news domain, the perplexity of gutenberg model was the highest. I also compared the perplexity values on a sentence from gutenberg test set. Similarly, the in-domain model's perplexity was the lowest, and the out-domain models' perplexity values were relatively higher.

5 Discussion

An observation we can find is that n-gram models could extract context from the corpus while unigram models cannot. Although the unigram model outperformed n-gram models on a sentence of a random sequence of words, we cannot say the unigram model is actually better because that means the unigram model falsely gives a higher score to a non-sensical sentence. This happens because the unigram model doesn't take the context into account at all and scores based on the individual words' probabilities, but n-grams models calculate each word's probability based on the n-gram's frequency, which makes more sense, and that is why n-gram models outperform the unigram model on a grammatically correct sentence.

Another observation is that a language model cannot perform well with the out-domain data. The model trained on brown corpus showed the lowest out-domain perplexity in general and for the examples, and that means it generalizes better than other models. This is because the brown corpus is the 'standard' corpus that covers general domains, unlike reuters and gutenberg corpora. The language used in a domain can be highly different from the language used in another domain. For example, in the news domain corpus like reuters, there are a lot of named entities like names of places or public figures, which are not frequently used in a literature domain like gutenberg corpus. On the other hand, spoken English is frequently used in the literature domain, whereas it is not in the news domain. Therefore, for the better model performance, we need to check whether our corpus covers the domains of target tasks.

6 Statement of Collaboration

I worked on this homework by myself without discussion.

References

W. N. Francis and H. Kucera. 1979. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US.

Martin Gerlach and Francesc Font-Clos. 2018. A standardized project gutenberg corpus for statistical analysis of natural language and quantitative linguistics. Cite arxiv:1812.08092.

Tony Rose, Mark Stevenson, and Miles Whitehead. 2002. The reuters corpus volume 1 - from yesterday's news to tomorrow's language resources. In *Proceedings of the Third Internaional Conference on Language Resources and Evaluation*, Las Palmas de Gran Canaria.