Performance Analysis of N-gram Language Models

Jan Fic
Computer Science
New College of Florida
Sarasota, Florida, United States
jan.fic18@ncf.edu

Rowan Holop
Computer Science
New College of Florida
Sarasota, Florida, United States
rowan.holop13@ncf.edu

ABSTRACT

Modeling language is at the core of Natural Language Processing. We present our analysis of N-gram models (n= 2, 3, 4) ability to predict sentences given a training and test corpus. The data we utilized was from Document Understanding Conferences (DUC): 2005's dataset. This dataset contained a total of 1593 files, 1323 of which were taken as a training corpus, and 270 of which were used for the test corpus. Our results showed decreasing perplexity with smoothing and with higher-order N-gram models, as well as improved log-likelihoods.

CCS CONCEPTS

• Natural language processing—Information extraction

KEYWORDS

Language Modeling, N-gram Language Models, Laplace-k smoothing

ACM Reference format:

Jan Fic, Rowan Holop. 2020. Performance Analysis of N-gram Language Models. (Unpublished)

1 Introduction

Language modeling is a field of study that focuses on predicting likely language patterns through the use of probabilistic models. The ability to predict human language is an integral part of many technologies in use today, such as speech recognition, spelling correction, and machine translation. Probabilistic models allow grammatical error correction systems to understand when "their" should be used instead of "there" by analyzing data and creating probabilities based on the contextual usage of these words.

An N-gram model is a model that analyzes sequences of N number of words; a 2-gram (bigram) model uses sequences of two words, for example, "ice cream." A bigram model relies on the conditional probability of the word preceding it. As N increases in our language models (3-gram, 4-gram, etc), so does the context the model draws from to predict plausible phrasing; in turn, the sentences the higher-order models can produce are more coherent than the sentences produced by lower-order models.

2 Dataset

Our dataset was taken from the DUC: 2005's dataset with permission. In Figure 1, below, we note the counts for unigrams, bigrams, trigrams, and 4-grams, as well as total sentence and file counts. Initially, it may seem unusual that our bigram count is higher than our unigram count, our trigram count is higher than our bigram count, and our 4-gram count is higher than our trigram count. However, when taking into consideration the sentence padding applied to our data set, we can note that it is likely the cause for the increased counts as our N-gram model increases.

Statistical Analysis of the Corpus			
Training Corpus		Test Corpus	
Unigrams:	39173	Unigrams:	16893
Bigrams:	354327	Bigrams:	109102
Trigrams:	675285	Trigrams:	176425
4-grams:	809900	4-grams:	200649
Sentences:	41202	Sentences:	9603
Total Files:	1323	Total Files:	270
		Unknown Words:	5100

Figure 1: Statistical analysis of the corpus including counts of unigrams, bigrams, trigrams, 4-grams, sentences, total files, and unknown words present in the test corpus.

2.1 Data Preprocessing

Preprocessing of the data set comprised many steps. Initially, we noted the majority of the data was formatted using HTML tags, in the style of a newspaper article. Preprocessing began by using an HTML parser designed by BeautifulSoup (Leonard Richardson. 2004.), followed by formatting the case of the text for consistency.

Then, we used NLTK's tokenizer (NLTK Project. 2020.) to create tokens based on the texts' sentences. Tokenization is the process of separating the text into smaller, more digestible parts, called tokens. After the text was split into smaller pieces, we performed lemmatization using NLTK's "WordNetLemmatizer" — this process takes words, removes suffixes, and returns the base of a word. For example, a lemmatizer might take the word “am” and transform the word into its base “be.” Punctuation was also removed from the lemmatized tokens. These tokens were then split into n-gram sets for analysis. N-gram counts can be seen in Figure 1, above.

3 Result Analysis

3.1 Comparison of Test and Training Corpus

Here we can see some simple comparisons of the top 40 unigrams occurring in our training corpus and test corpus:

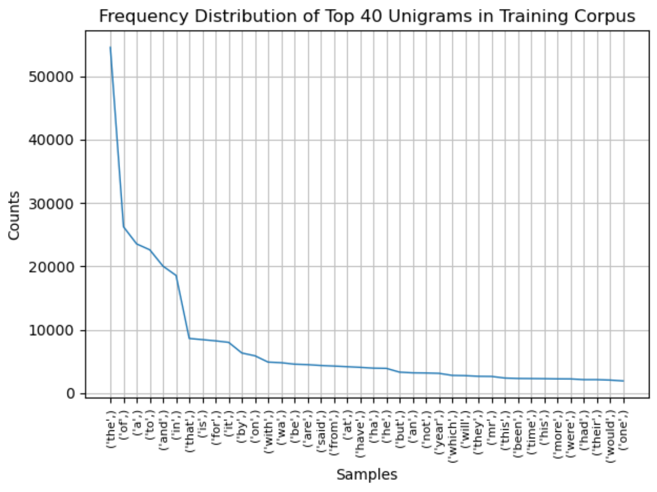

Figure 3: Frequency distribution of top 40 unigrams in training corpus.

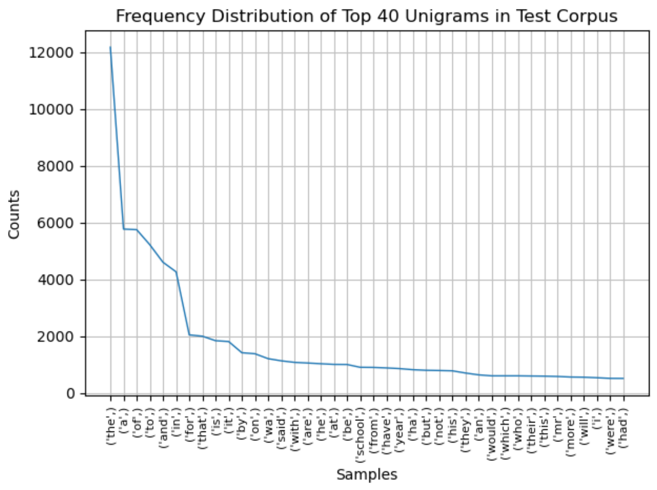

Figure 4: Frequency distribution of top 40 unigrams in test corpus.

Figures 3 and 4 show the frequency distribution of the top 40 unigrams in our training corpus and our test corpus. Interestingly, the shape of the graph is very similar between both graphs. The majority of the most common words are stop words, such as “the”, “a”, “of”, etc, as we expect. Stop words are often filtered out before the examination of data, however, since our goal with the data set is to predict the probability of sentences given a training corpus, removing stop words would make analyzing the likelihood of a given sentence more difficult. We will discuss analyzing sentence probabilities more in the next section.

3.2 Example Tests on Sentences

In the tables below, probability was calculated by taking the log of the following formula:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

For sentence probability, the data was also smoothed to prevent probabilities from being zero.

Probability	Sentence: "It also calls for a 10year plan to restore closeddown rail links and development of harbours”
Bigram	-54.8834995287637
Trigram	-27.9015305704361
4-gram	-5.605802066296

Figure 5

Probability	Sentence: "More than 60,000 such trees will be recycled this year, according to community service charity CSV"
Bigram	-57.5344887619286
Trigram	-24.9928137481933
4-gram	-3.66356164612965

Figure 6

Probability	Sentence: "These trees would once again breathe life into the soil"
Bigram	-38.7119951277833
Trigram	-9.88756140198587
4-gram	-1.20397280432594

Figure 7

The above figures show the performance of our model on three different sentences taken from the test corpus. The most likely sentence would be the one given in Figure 7, as it performed the best on all three models. All of our sentences had the best performance on the 4-gram model, however, some sentences performed better on certain models than others. For instance, Figure 6's sentence performed more poorly on the bigram model than Figure 5's sentence, but better on the trigram model. Without smoothing, the probability of these sentences would all be zero, as they each had at least one N-gram that did not appear in our training corpus. However, this is not overall very surprising given the high number of unknown words in our test corpus.

3.3 Overall Performance

	Bi-gram	
	With smoothing	Without smoothing
Average log-likelihood	-4.44936097469887	-5.08839962682171
Perplexity	85.5722436978157	162.130185518897
	Tri-gram	
	With smoothing	Without smoothing
Average log-likelihood	-1.39962381454228	-2.33377012261512
Perplexity	4.05367474648548	10.3167637691643
	4-gram	
	With smoothing	Without smoothing
Average log-likelihood	-0.362817823108349	-1.15148596792515
Perplexity	1.43737397906546	3.16288937158682

Figure 2: Table of the average log-likelihood for N-grams and their perplexity

Above, Figure 2 shows the results of analyzing our test corpus against our training corpus. Our results were as expected for our simple N-gram model. We utilized Laplace smoothing for our smoothing method, which we know does not perform well enough to be used in modern N-gram models (Jurafsky, Martin. 2020.). However, the simple smoothing method still improved our models overall.

As we had hoped, our 4-gram model performs the best, with smoothing having a very low perplexity and an average log-likelihood fairly close to 0. Since the perplexity is the inverse probability, normalized by the number of words, the lower the perplexity, the more information our model is giving us about our data.

The average log-likelihood was calculated for each N-gram model by taking the log of the probability of each N-gram and dividing their sum by the number of total N-grams in the test corpus. For the data without smoothing, N-grams whose probabilities were zero were taken out of the total N-grams before averaging.

Perplexity was calculated by multiplying the Nth-root of the inverse of the probability of each N-gram together. In this calculation, the Nth-root also accounted for data without smoothing whose probabilities were 0.

4 Discussion

Overall this analysis gave us the expected results. However, because we had such a high number of unknown words in our test dataset, our numbers for perplexity and probability sometimes got very unreasonable. If we were to process this data again, ideally, we would increase the size of the training dataset to get more accurate probabilities. Also, a more advanced method of smoothing could be used instead of additive smoothing, such as Knesser-Ney smoothing. This would allow for more reasonable probabilities and take into account our smaller dataset.

Additionally, the files could be preprocessed to avoid other factors that may add to modeling issues, such as our high count of unknown words. Many of the files appeared to be newspaper articles, and many and identifying information such as "LA051390-0183" at the top of articles to specify document numbers. These identifiers are likely contributing to our very high count of unknown words.

It would be interesting to see how other methods of building a language model on this dataset would compare. Perhaps using the common machine-learning algorithm of Recurrent Neural Network (RNN) would perform better than our more simplistic N-gram model in terms of text likelihood predictions.

REFERENCES

[1] DUC 2005: Task, Documents, and Measures. (2005). Retrieved September 21, 2020 from https://duc.nist.gov/duc2005/tasks.html

[2] Dan Jurafsky and James H. Martin. 2020. Speech and Language Processing (3rd ed. draft). Stanford University, Stanford, CA.

[3] Leonard Richardson. 2004. Beautiful Soup Documentation¶. (2004). Retrieved September 23, 2020 from https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[4] NLTK Project. 2020. Natural Language Toolkit. (2020). Retrieved September 23, 2020 from http://www.nltk.org/index.html