Next Word Prediction using N-grams

Nikitha Nagaraj

Nikithanagaraj23@gmail.com

Table of Contents

ABSTRACT

Word prediction is an "intelligent" word processing feature that suggests a set of words which can appear next in the sentence. We see this feature being used in everyday life in predictive text entry systems, word completion utilities like messaging applications, and writing aids. The main essence of this project is to develop a word prediction engine which provides a list of the possible words which are most likely to appear next in a sentence based on the phrase which has been entered by the user. The project is designed to use a corpus of text data which is a unique collection of words, phrases, and sentences collected from news articles which is used to train the data. While generating the predictive model a portion of the data is used for training the model and a portion of the data corpus is used to test the model. In addition to this the project aims to portray the variation in accuracy based on the size of the training data corpus.

INTRODUCTION

The prediction of user behavior is a basis for construction of various assistance systems. To name a few these predictive models come in hand while developing augmentative communication systems to predict the next words a disabled user wants to convey, computer aided education where the system helps kids learn to read, speech recognition and context sensitive spell check. Predictive models are used in day to day life in our mobile devices.The predictive model which was developed in this project is an n-gram based language model which uses probabilistic data. The model is fed with different sizes of data and tested against a test dataset to check how accurately the model is able to predict the next word possible.

BACKGROUND

Over the past few years a number of systems have been proposed to predict words based on the previously entered data. These methods either fall under the classic statistical and probabilistic language models as the n-gram models or the syntactic methods in which syntactic information is used to predict the next word. The probabilistic approach uses unigrams and bigrams in order to do the prediction task. In the paper, “Prediction and Entropy of Printed English” written by Shannon, the entropy and redundancy of a language is determined using the knowledge of predicting the next letters when the preceding text is known. In the syntactic method of word prediction part of speech is used along with unigrams and bigrams. For instance in the paper “The Use of Syntax in Word Completion Utilities”, the author presents a comprehensive review of prior related work in word prediction. Fazly also presents a collection of experiments on word prediction applied to word completion utilities. The implemented and evaluated algorithms were based on word unigrams and bigrams, and based on syntactic features like Parts of speech tags in the syntactic predictors, and combination. The training and testing are done on texts taken from British National Corpus. Other methods involving the use of recurrent neural networks have also been used in the recent past to solve the same issue. Using neural networks the model not only looks at the previous few words, but also remembers the context of the sentence and predicts the next words. With an extensive amount of training, word prediction using neural networks can reach a very high

accuracy level as the system learns the context of the language and predicts words unlike n-grams which uses only the previous three to four words to do the same task.

UNDERSTANDING THE PROBLEM

Natural language processing (NLP) is the ability of a computer program to understand human speech as it is spoken. It is the field of computational linguistics which provides approaches, tools and methods that may be used in a range of processing tasks. Computers require humans to speak to them in a language that is unambiguous and highly structured for NLP applications. This could be challenging as generally human speech is not always precise. The structure of the language could depend on many complex variables including slang, regional dialects and social context.

The challenge in analysing a large data corpus is to understand the structure and arrangement of the words. The essence of the project is to collect a large chunk of data in the form of text, clean and analyse the data and then feed it to the predictive language model. The model ideally takes the input from the user and based on the last three words entered, a list of words are generated which are most likely to appear next. The input data could be in a formal or informal format. There are different approaches to predict the next possible word. We could use either language modelling approach or neural networks. It is expected that the accuracy of a predictive model depends on the number of unique words that are available in the data corpus.

DATA PROCESSING

Data

The basic approach to train a language model would be to first collect a large corpora of data which is a collection of words, phrases, conversations and sentences. In this case the data corpus is a collection of text from a bunch of news articles. This collection reads 1%,3%,5%,10% and 90% of the entire data corpus which originally has 1,010,242 lines of data which add up to almost 200Mb of data . Around 0.5% of this data is set aside as a testing data set which will be used to test the predictive language model.

Data from different sources can yield different results as the structure of the language differs. For instance the data from twitter consists of very short informal sentences with short hand writings for certain words. The data in the news articles however are more precise as the language used is very neat with fewer number of abbreviations and shorthand words used. The model will refrain from proposing profanity words. The source data for this project has been taken from the Coursera dataset downloaded from https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip.
Each of these files consists of multiple sentences and phrases in the form of first person speech as well as narration. The data might also include extra symbols and Unicode characters which need to be taken care of. These files may include formal and informal language, punctuations like single and double quotes surrounding phrases or words and contradictions. Hence this data needs to be cleaned in order to use it to train our model.

The data being used to train the model consists of text like the following:

> *"It's just another in a long line of failed attempts to subsidize Atlantic City," said Americans for Prosperity New Jersey Director Steve Lonegan, a conservative who lost to Christie in the 2009 GOP primary. "The Revel Casino hit the jackpot here at government expense."*

Number of Words	% of data used
32831	1
59573	3
77573	5
12649	10
342275	90

Fig. Number of words present against percentage of the data corpus

Data Cleaning:

Prior to using the data to train the model it is important to clean the data corpus as there might be a considerable amount of unwanted texts and symbols. For our model we will be using data which are only English words. In this model predictions are done for lower case letters. So first the entire data corpus is converted to lower case format. The data corpus being used cannot be read directly, they need to be read using utf-8 encoding. Since punctuations do not play a very important role in the word prediction, for our model we will be removing the punctuations in order to make parsing and tokenizing simpler. Considering the fact that the data has been collected from various sources we will need to remove the extra spaces, tabs and blanks in order to compress the data. We do not remove numerical data and stop words from our dataset as they are very commonly used which imply that they do play a major role in predicting the next words in the language model.

Since this model does not predict offensive words we will need to clean the data corpus to get rid of these words. In order to do so we use data from the links http://www.bannedwordlist.com/lists/swearWords.txt and http://www.cs.cmu.edu/~biglou/resources/bad-words.txt.

MODELING AND PREDICTION

Building the N-gram tables:

In the field of computational linguistics and probability, a contiguous sequence of words in a given sequence of text or data is known as an n-gram. The items can be phonemes, syllables, letters, or words. In our case we will be using a list of words as our n-grams,

these are called shingles. Based on the number of items in our n-grams they will be considered as unigrams, bigrams, trigrams, four-grams and so on. The n-gram model is a type of probabilistic model for predicting the next item based on the n-1 sequence of words prior to it.

Now n-gram tables need to be generated so that they can be used later. For this the data corpus is parsed and split by space to get all the words. Now we generate unigrams (n=1), bigrams (n=2), trigrams (n=3) and four-grams (n=4) from this data. Once the n-grams have been generated the frequencies of each of the entries need to be calculated and sorted in the reverse order of their frequencies. In Natural Language Processing, the ratio of unique words to the total number of occurrences of these words is known as language coverage. If more words are included we expect that the language coverage will increase. The larger the corpus the more time and more memory space is required to generate the n-grams.

Once we have the frequencies against the n-grams we split the n-gram tables to give the n-1 words and the final word corresponding to the predictor and the predicted word. For example, view the table below after splitting the four-grams as n-1 and n words respectively:

for the first	time
the end of	the
at the end	of
the rest of	the
said in a	statement
one of the	most
in the united	states
at the same	time
is one of	the
when it comes	to

While using an n-gram model, the length of the input phrase does not matter. The model can predict a word on a four-gram, trigram, bigram or even unigram. Since the frequencies of the n-grams have already been computed and stored, accessing this data when required becomes very easy.

Probabilistic models use conditional probability chain rules and assume that the probability of a word depends only on the limited number of previous words that we have come across. This solves the issue that the longer the sequence the more difficult it is to find it in the data corpus.

Conditional probability for the phrase is calculated using the formula:

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

Where $w_i$is the last word and w_{i-1} is the preceding few words in the input text. The probability of the word w_i following the words w_{i-1} is given by the ratio of the number of times the word phrase (w_{i-1}, w_i) appears in the corpus to the number of times the number of times the phrase w_{i-1} appears in the data corpus. In order to generate more

accurate results we add a smoothing value of 0.1 to the above formula. The formula now becomes:

$$Pr(w_i|w_{i-1}) = \frac{count(w_{i-1}, w_i) + 0.1}{count(w_{i-1}) + 0.1 * V}$$

Where V is the vocabulary set which includes all the alphabets (a-z) and numbers (0-9) which adds up to a total of 37.

Prediction of the word:

The prediction algorithm takes the text entered by the user as the input. The model then displays a list of words which are most likely to appear by looking at this input data. The last three words entered are considered as the n-1 items. These n-1 items are used to find the next words which can appear by calculating the conditional probability for each of these probable words.

To do so the following steps are followed:

- The user inputs the data in the form of text. The length of the data entered does not affect the model. The only constraint is that the data should contain more than one word.
- The data entered is now cleaned and evaluated to predict the next word.
- Since we have generated the n-gram table to show the nth word based on the (n-1)th word accessing the possible next words becomes simple.
- Now based on the size of the text the four-gram, trigram or bi-gram tables are looked up and the probable words are found.
- If no words are predicted from the four-gram look up, we move to the trigram look-up and check for the possible next words. If there are no words predicted here either, we move to the bi-gram look up table.
- Once we have the list of possible next words we calculate the conditional probability using the n-gram frequency tables.
- This list of possible words are sorted by the decreasing probability and displayed as the list of predicted words.

Below is an example of how words are predicted by the model:

Enter the data to predict the next word:

will you

Predicted Words:

['get', 'have', 'need', 'were', 'feel', 'better']

Enter the data to predict the next word

what is

Predicted Words:

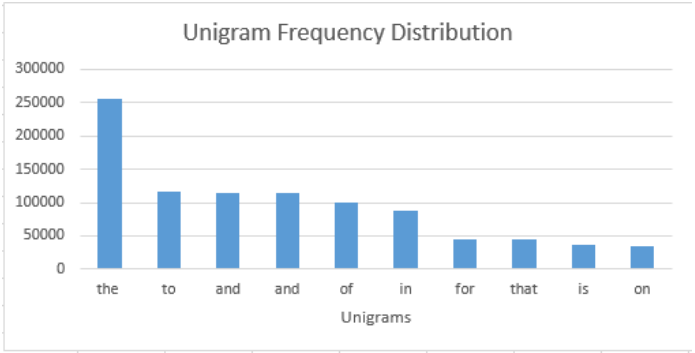
['a', 'the', 'going', 'expected', 'worse', 'another', 'accused', 'located', 'moving', 'little', 'left', 'promoting', 'unprintable']

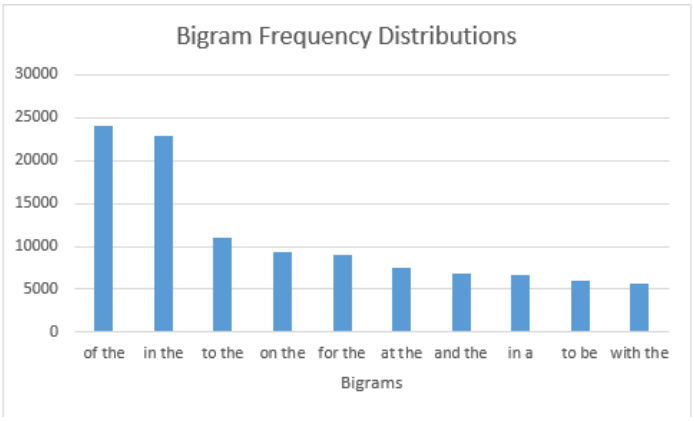
Since the main aim of this project is to analyze how the accuracy of the predicted data varies with the amount of data used to train the model, a portion of 0.5% of the entire corpus has been set aside to test the model.

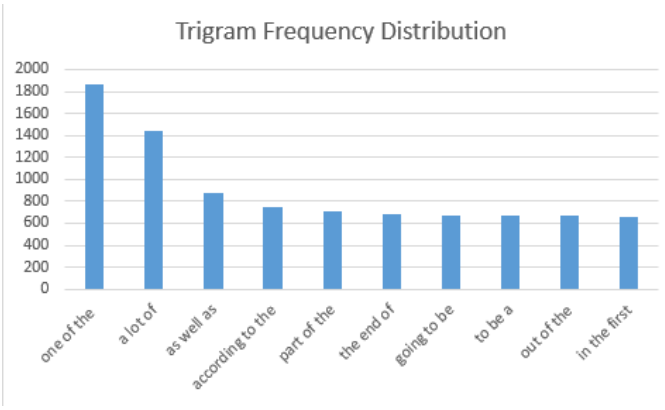
For this scenario, the test data is passed through the model and cleaned to remove unwanted data as we did for the training corpus. Now as we parse through this data, we check what the $(n+1)^{th}$ word would be based on the prior n words, assuming we are at the nth position in the text. So now based on the size of the text, four-gram, trigram or bi-gram tables are checked to find the next probable word as done before for a user input. Once we get the list of words which are most likely to appear next based on our model we check if the $(n+1)^{th}$ item in our data is present in the list of predicted words. If the word exists it is registered as a success, else as a loss. Based on the ratio of successful predictions to the total number of words present in the corpus the accuracy is provided.

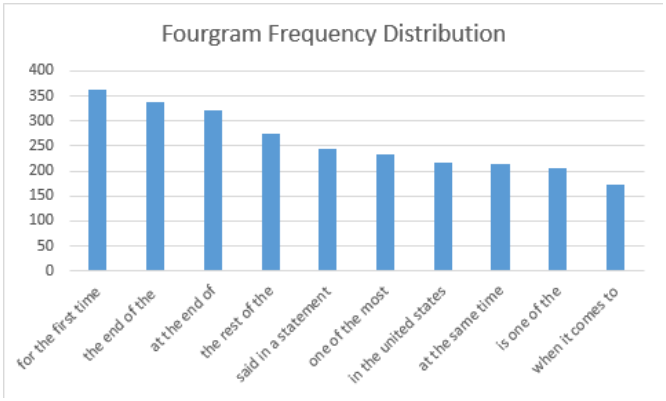
RESULTS

The charts below show the top 10 most frequently occurring unigrams, bigrams, trigrams and four-grams in 10% of the total data corpus which would add up to 101,024 lines of text.







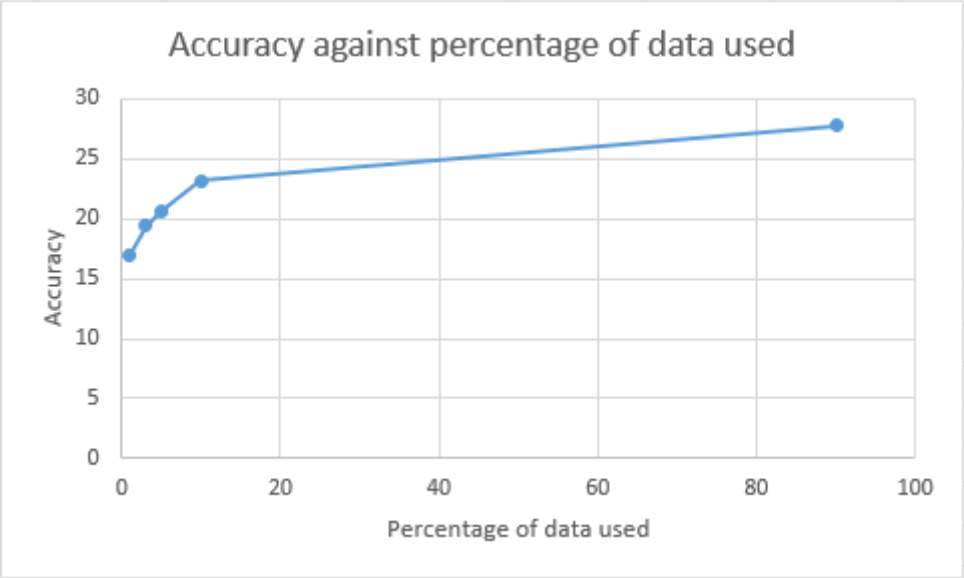


When the test data was run against different quantities of data sets (1%, 3%, 5%, 10% and 90%) of the data corpus it was observed that the accuracy of the predicted text kept increasing. The accuracy levels increased from a value of 17.0% to 27.76% as we increased the data set.

The accuracy was calculated using the formula:

$$Accuracy = \frac{number\ of\ words\ predicted\ correctly}{total\ number\ of\ words\ in\ the\ test\ data} * 100$$

The below shown graph depicts the change in accuracy as the training data set was increased.



CONCLUSION AND FUTURE WORK:

In this prediction modeling approach many observations were drawn based on the type of data, size of data, type of speech, word types and vocabulary. In order to make it easier to predict the next possible words the algorithm was made such that look up tables were generated to pull out the words which are most likely to follow by keeping the key as the n-1 words and the nth word as the value. Different quantities of data yielded a variation in the accuracy received. Whenever a three word phrase was not found in the look up, we moved to a two word phrase look up or even a one word look up. This is called Katz Back off approach. The accuracy we received from the model ranged from 17% to 27.76% as we increased the data set from 1% to 90%.

Since it is evident that the accuracy seems to be increasing as we provide more data as a training corpus, the accuracy of this model could be improved by training the model with much more data. This would however, required enormous amounts of processing power, time and memory in order to do the entire computation. N-grams deals with predicting words based on only the previous few words. This might not always be what is required as in some cases we will need to come up with words related to a noun in the same context. For instance “The laptop I left on the top floor crashed”. Here the n-gram model would not be able to guess the word crashed as it is in reference to the laptop. In such cases we will need a model which learns the context of the data. This is where neural networks could provide a better accuracy rate than n-grams.

REFERENCES:

1. Vlasblom, A. (2017). Text Prediction Using N-Grams. [online] Rstudio-pubs-static.s3.amazonaws.com. Available at: http://bit.ly/2oEK6M0 [Accessed 24 Apr. 2017].
2. En.wikipedia.org. (2017). N-gram. [online] Available at: https://en.wikipedia.org/wiki/N-gram [Accessed 24 Apr. 2017].
3. Rpubs.com. (2014). RPubs - Report1. [online] Available at: http://bit.ly/2oEOxGx [Accessed 24 Apr. 2017].
4. Rstudio-pubs-static.s3.amazonaws.com. (2017). Next Word Prediction:Exploratory Report. [online] Available at: http://bit.ly/2oEU9k0 [Accessed 24 Apr. 2017].
5. Lagunita.stanford.edu. (2017). [online] Available at: http://stanford.io/2oER4QX [Accessed 24 Apr. 2017].
6. Modsimworld.org. (2017). [online] Available at: http://bit.ly/2oERc2T [Accessed 24 Apr. 2017].
7. Pdfs.semanticscholar.org. (2017). [online] Available at: https://pdfs.semanticscholar.org/245c/304bf1a24bacb16dc8d3a472eb045239ed6f.pdf [Accessed 24 Apr. 2017].
8. Cs.cornell.edu. (2017). [online] Available at: http://www.cs.cornell.edu/courses/cs4740/2014sp/lectures/n-gram-models-1.pdf [Accessed 24 Apr. 2017].
9. En.wikipedia.org. (2017). Katz's back-off model. [online] Available at: https://en.wikipedia.org/wiki/Katz%27s_back-off_model [Accessed 25 Apr. 2017].
10. Languagelog.ldc.upenn.edu. (2017). [online] Available at: http://languagelog.ldc.upenn.edu/myl/Shannon1950.pdf [Accessed 24 Apr. 2017].
11. SearchContentManagement. (2017). What is natural language processing (NLP)? - Definition from WhatIs.com. [online] Available at: http://searchcontentmanagement.techtarget.com/definition/natural-language-processing-NLP [Accessed 24 Apr. 2017].
12. Isl.anthropomatik.kit.edu. (2017). [online] Available at: http://isl.anthropomatik.kit.edu/cmu-kit/downloads/00266531(1).pdf [Accessed 24 Apr. 2017].
13. Www2.edc.org. (2017). What is Word Prediction?. [online] Available at: http://www2.edc.org/ncip/library/wp/what_is.htm [Accessed 24 Apr. 2017].

14. Pdfs.semanticscholar.org. (2017). [online] Available at: http://bit.ly/2oEE6To [Accessed 24 Apr. 2017].

15. Ftp.cs.toronto.edu. (2017). [online] Available at: http://ftp.cs.toronto.edu/pub/gh/Fazly-thesis.pdf [Accessed 24 Apr. 2017].

16. Pdfs.semanticscholar.org. (2017). [online] Available at: https://pdfs.semanticscholar.org/05ab/08c1f3ddfcb645a53ea0c9c9c936e194c10d.pdf [Accessed 24 Apr. 2017].

17. Coursera. (2014). SwifKey Text Dataset [Data file]. Retrieved from https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip.

18. Bannedwordlist.com. (2017). [online] Available at: http://www.bannedwordlist.com/lists/swearWords.txt [Accessed 24 Apr. 2017].

19. Cs.cmu.edu. (2017). [online] Available at: http://www.cs.cmu.edu/~biglou/resources/bad-words.txt [Accessed 24 Apr. 2017].