Natural Language Processing: A Model to Predict a Sequence of Words

Gerald R. Gendron, Jr.
Confido Consulting
Chesapeake, VA
Gerald.gendron@gmail.com; LinkedIn: jaygendron

ABSTRACT

With the growth of social media, the value of text-based information continues to increase. It is difficult to analyze a large corpus of text to discover the structure within the data using computational methods. Alan Turing (1950) opens his influential article "Computing Machinery and Intelligence" with the statement, "I propose to consider the question, 'Can machines think?'" (p. 433). Overall, this Turing Test has become a basis of natural language processing. The essence of this project is to take a corpus of text and build a predictive model to present a user with a prediction of the next likely word based on their input. A key aspect of the paper is discussion of techniques balancing accuracy and scalability for large data sets. This paper provides the analysis decisions used to develop that predictive text model for a corpus of over 500,000 blog articles. The resultant model exists as a web-based data product that allows for customizable user reporting. Additionally, the work presented in this project follows the tenets of reproducible research and all code is available in an open-source repository to enable readers to review the approach, reproduce the results, and collaborate to enhance the model.

Keywords: natural language processing, predictive model, text mining, predictive text analytics, N-Gram, data product, Good-Turing Smoothing, Katz back off

ABOUT THE AUTHOR

Gerald "Jay" Gendron is a data scientist who consults for various organizations through his group Confido Consulting. He is dedicated to the idea that decision makers have greater access to analysis than some may lead them to believe. As open data becomes more and more commonplace, decision leaders can have faith in their decision-making processes by asking questions and requesting data products rather than dashboards and reports. Jay is a business leader, artist, and author who writes on various perspectives of how good questions and compelling visualization make analysis accessible to decision makers. His analytic pursuits include finding trends in the startup and entrepreneurial communities, assessing learning and sociological impacts of technology-centric training systems, and making results tell the story of business case analyses for procurement and supply chain strategies. He is especially enjoys writing about the impact generational differences have on workforce training and managerial approaches. Jay is an award-winning speaker who has presented at international conferences and symposia. He volunteers his time with the national group Code for America – contributing data science skills to projects aimed at improving civic and municipal access to data and data products.

Natural Language Processing: A Model to Predict a Sequence of Words

Gerald R. Gendron, Jr.
Confido Consulting
Chesapeake, VA
Gerald.gendron@gmail.com; LinkedIn: jaygendron

EXECUTIVE SUMMARY

This report provides documentation describing the process and decisions used to develop a predictive text model. The model uses natural language processing techniques to accomplish predictive analytics. The main body of the report provides a descriptive approach to predictive modeling by summarizing key considerations encountered during the analysis. Key learning points are included to aid readers interested in reproducing this work and enhancing it. Overall, the discipline of natural language processing is a broad and useful domain of data science. This report includes a brief literature review capturing key concepts that guided this project. The corpus used in this analysis has a sort of personality. More specifically, it is a unique collection of words, phrases, and sentences affecting the resulting prediction model. Exploratory data analysis helped characterize details of the text corpus and determine which parts of the corpus would be most useful for prediction. The initial model created from a small subset of the corpus was accurate, but it was not scalable. Additional research, creative thinking, and persistent modeling alterations resulted in a predictive text model that balanced accuracy with scalability. The model described in this report is a web-based data product made available to readers for review. It allows users to customize results of the analysis to suit their desires. Lastly, the project follows the tenets of reproducible research and all code used in the development of the project is contained in a Github repository (Gendron, 2014).

UNDERSTANDING THE PROBLEM

A most important aspect at the outset of any data analysis project is to understand the problem. With the advent of social media and blogs, the value of text-based information continues to increase. The practical application of extract value from text is increasing seen in areas like click-based web marketing, customer segmentation, and sentiment analysis of Twitter and Facebook comments. The problem in analyzing a large corpus of text is to discover the structure and arrangement of words within the data in order to analyze the corpus using computational methods. The essence of this project is to take a corpus (a body) of text from various sources, clean and analyze that text data, and build a predictive model to present the next likely word based on the prior two words provided by a user. User input could range from formal, professional communication styles to informal, short messages – more typical in social media. Therefore, knowledge of the data characteristics in the corpus is essential. As a concrete example, a user may type into their mobile device - "I would like to". A predictive text model would present the most likely options for what the next word might be such as *"eat"*, *"go"*, or *"have"* - to name a few.

Data sciences are increasingly making use of natural language processing combined with statistical methods to characterize and leverage the streams of data that are text based and not inherently quantitative. There are many techniques available within the R programming language to work quantitatively with text. A key aspect of this project is to discern which techniques best promote accuracy and scalability for large data sets. This project provides a unique contribution. Other models for predicting text are proprietary products used on various mobile platforms. This project makes all the code and algorithms available as open, collaborative code for others to investigate and improve.

LITERATURE REVIEW

An Origin of Natural Language Processing

Alan Turing (1950) opens his influential article "Computing Machinery and Intelligence" with the statement, "I propose to consider the question, 'Can machines think?'" (p. 433). He follows this by outlining something he calls the imagination game played by man A – known as label X, woman B – known as label Y, and interrogator C. The

interrogator is able to ask questions of X or Y in order to attain the objective to properly identify whether "X is A and Y is B" (p. 433) or vice versa. Turing later refines the original question to read, "Are there imaginable digital computers which would do well in the imitation game?" (p. 442). In essence – to predict truth based on features or interrogation. He speaks to a machine teaching process made up of rewards and punishments enforcing orders in symbolic language. Overall, this Turing Test has become a basis of natural language processing – covering a broad array of uses such as spelling correction, speech recognition, author identification, and prediction of words based on preceding words.

Literature Review Purpose and Findings

At the outset of this project, a literature review identified various sources on natural language processing, text mining, and various R programming packages. Further literature review helped extract clues for building a model. Primary goals of the literature review were to understand:

- common issues when analyzing text data
- the body of knowledge that has built up in the domain of natural language processing
- other resources not provided by this course for helping us in the analysis methodology

This section is not a comprehensive overview of over 40 sources reviewed but merely a summary of the two works most influential in shaping the modeling approach taken in this project.

Feinerer, Hornik, and Meyer (2008) provide a good overview of the essential aspects of the R packages on NLP `openNLP` and text mining `tm`. Noteworthy was their information about: reading in corpora into the R environment; explaining functions to transform the data; explaining stemming, stopwords, and tagging parts of speech; considering the issue of text sparsity; and understanding the fundamental of count based analysis.

Jurafsky and Martin (2000) provide a seminal work within the domain of NLP. The authors present a key approach for building prediction models called the N-Gram, which relies on knowledge of word sequences from *(N – 1)* prior words. It is a type of language model based on counting words in the corpora to establish probabilities about next words. Overall, Jurafsky and Martin's work had the greatest influence on this project in choosing among many possible strategies for developing a model to predict word selection. It addresses multiple perspectives of the topics found in Feinerer, Hornik, and Meyer. The following approaches and assumptions were chosen for purposes of building an initial model. They could then be adapted to increase prediction accuracy.

1. Case: corpora words will not be case-sensitive. Although important for spelling correction and part of speech analysis, the words themselves - not their case - are important for prediction.
2. Stopwords: similarly, unlike classification and clustering applications, all words will be included in the model as they represent more than just the primary carriers of the message.
3. Wordform: stemming will not be used as N-Grams are typically based on wordforms (unique, inflected forms of words). Whereas *table* and *tables* are the same lemma, they will be treated as separate words in this model.
4. Punctuation: Jurafsky and Martin treat punctuation as a word and count it as a word. Given the nature of the problem, which is not trying to generate full sentences but only predict a next word, punctuation will be treated slightly differently in the initial model. End of sentence punctuation (e.g., ? ' ! .) will be used to include end-of-sentence tags, as the intuition is they have implications for word prediction.
5. Parts of Speech: the discussion of N-Grams did not imply the inherent value of predication based on syntactically using parts of speech.
6. Numbers: there is no intuition based on the research that numbers will have a great impact on a predication model and they will be removed
7. Sparse Words: all words will be retained. A key concept from Jurafsky and Martin is the idea that even bigram models are quite sparse; however, rather than eliminating those wordforms, they become clues to the "probability of unseen N-Grams" (p. 209, 2000). They include as their fourth of eight key concepts *Things Seen Once* and recommend using the count of wordforms seen a single time to provide a basis to estimate those things not seen in the training set and will likely appear in a test set

8. Whitespace: this was not discussed directly by Jurafsky and Martin. The intuition is that whitespace has little to do with context and excess whitespace will be removed

In addition to shaping the initial strategy, the work by Jurafsky and Martin provided valuable insights on other aspects of implementing a predication model.

- Historical context of the unsmoothed N-Gram and basis of probabilistic approach to predicting words using the Markov assumption to simplify probabilities by looking only at *(N - 1)* or *(N - 2)* previous words
- Importance of having a diverse corpora to improve generalizability of prediction among other corpora
- Comparative analysis of smoothing and discounting techniques to increase predictive power
- Introduction of back off techniques to establish probabilities for otherwise unseen elements of an N-Gram
- Discussion of entropy and perplexity - which may prove to be a good choice as the single measure to help assess quality of the prediction model

DATA PROCESSING

We have to understand the data, determine what should be done with the data, and generate the questions that need to be asked to ascertain whether the data is sufficient to do the job. This section briefly addresses the acquisition, processing, and exploration of the data.

Data Acquisition and Cleaning

SwiftKey was a corporate partner involved with this project. They produce software to aid users in rapidly entering text with higher accuracy on mobile devices. Based on their collaboration with Johns Hopkins University, a dataset (Coursera, 2014) was provided for this project. The dataset is a zip file including blog posts, news articles, and Twitter tweets in four languages (English, German, Finnish, and Russian). The data was downloaded using the R programming language (R Core Team, 2014) and the elements were extracted using the R text-mining package called `tm` (Feinerer. Hornik, & Artifex Software, 2014). The `PCorpus` function was used as it establishes a permanent database instead of a virtual one. This allowed a database to hold over 3.3 million documents in physical disk memory rather than completely in RAM to reserve processing capacity.

A preliminary exploration of the corpus indicated the raw data required a number of transformations to prepare it for statistical modeling. The data set was cleaned using over 20 transformations that pre-processed the data for analysis. In general, those transformations included conversion to lower case, ensuring apostrophes were retained to maintain contractions, removal of numbers, and removal of excess whitespace. Intermittently in the process, the corpus was written back to disk and the database was re-initialized using the `filehash` package to reduce the size of data processing in RAM. Although one-third of the way through processing the corpus had exceeded 1 GB in RAM, use of `filehash` kept this well below that. The final corpus of over 539,573 blog documents after all processing was only 108 MB. It is worth noting the original data set of 899,288 blogs was over 205 MB.

Exploratory Analysis

Clean data is a necessary but not sufficient condition to developing a prediction algorithm. This step is about understanding the relationships between the words and sentences as well as other observable artifacts. Results of the many hours of exploration included a better understanding of relationships between vocabulary size and unique words, distributions of various N-Grams, and information that helped reevaluate the original strategy presented in the literature review. Table 1 provides statistics on the full corpora. It shows the total number of documents in each genre of the corpus.

Table 1: Characterizing the Corpora by Word Count, Type, Ratios, Diversity

Source	Documents	Vocabulary (V)	Word Types (T)	TTR (T/V)	Diversity
Blog	899,288	37,334,131	1,103,548	0.030	127.71
News	77,259	2,643,969	197,858	0.075	86.04
Tweets	2,360,148	30,373,543	1,290,170	0.042	165.53
Corpus	3,336,695	70,351,643	2,123,809	0.030	179.04

Table 1 also shows the total vocabulary (V), which equals the number of total word tokens present in each genre. Just over half of the total Corpus is composed of blog posts. Word types (T) are the number of unique words within the Vocabulary. The Type/Token Ratio (TTR) is a well-documented measure of language comparison. It equals the total word types divided by vocabulary (Richards, 1987). The TTR indicates complexity, with higher numbers indicating a more complex genre. Tweets are the most complex because it takes more unique words to build a smaller vocabulary. This measure was used to make a decision to limit the data for the model to just blog entries. News articles tend to be overly repetitive with non-conversational language and tweets are a language unto themselves with many "created" words.

Diversity is the last column of Table 1. According to Richards (1987), it is a more useful measure because TTR will tend to fall off just from a growing vocabulary. Diversity is defined as "a measure of vocabulary diversity that is approximately independent of sample size is the number of different words divided by the square root of twice the number of words in the sample" (Richards, p.208). It is robust and positively correlated to the tokens. It also flattens out at some point. Table 2 shows the effect on diversity as the size of the vocabulary (document numbers) increases. There is a relative flattening out at 60 percent of the total documents. This corresponds to a common technique for separating data into a 60 percent training set, a 20 percent test set, and a 20 percent validation set. This observation helped solidify the decision to use a training set composed of 60 percent of all blogs in the dataset.

Table 2: Effect of Vocabulary Size on Diversity Measures

Measure	Type	50	500	5,000	50K	60%	80%	Entire
Diversity	Blog	16.38	34.17	57.31	83.07	118.55	123.72	127.71
Diversity	Corpus	22.68	45.22	72.56	103.84	163.43	172.10	179.05

Notice in Table 3 that the widely used measure of Type/Token Ratio shows the similarity in complexity as represented by the blogs and the entire corpora of blogs, news, and tweets.

Table 3: Effect of Vocabulary Size on Type/Token Ratios

Measure	Type	50	500	5,000	50K	60%	80%	Entire
TTR	Blog	0.51	0.31	0.18	0.08	0.04	0.03	0.03
TTR	Corpus	0.49	0.29	0.15	0.07	0.04	0.03	0.03

Understanding the distribution among the word tokens helps shape expectations of the linguistic model. An N-Gram refers to the number of words in a string. This project will work on a 3-Gram model. The basic building blocks of that model are unigrams (N: n=1), bigrams (N: n=2), and trigrams (N: n=3).

Figure 1 provides the distribution of the frequencies of each of these word forms from the 60% set of blogs.

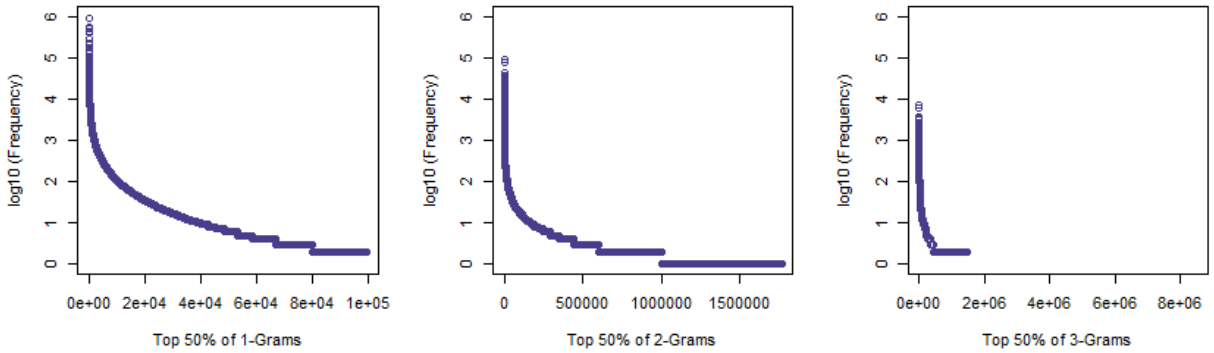

Figure 1: Distributions of Three N-Gram Models

Prior to plotting, the frequencies were sorted in decreasing order and the x-axis is an index. Therefore, the far left of each plot in Figure 1 indicates an N-Gram with the highest frequency. Notice a few features of the data. Although there are far fewer unigrams (198,576) than bigrams (3,525,643) and trigrams (17,021,221), their distributions are much more skewed right as the level of N increases. This indicates very long tails (more frequencies of 1, 2, etc.) but at the same time much higher frequencies of particular words (such as "the", "and", "to", etc.). The three plots are logarithmic in the y-axis and set at a common y-axis limit to aid in comparison. In logarithmic scale $\log_{10}(1)=0$. Figure 1 gives an idea of how many singletons exist:

- within the first half of all unigrams, no frequencies of "1" are seen
- for bigrams, single frequencies occur at about 30 percent (around 750,00)
- for trigrams, single frequencies occur after only 10 percent of 3-grams appear. All single 3-grams were removed from the dataframe to work within RAM – therefore, no plot points appear at $\log_{10}(\text{frequency})=0$)

Jurafsky and Martin (2000) also describe the importance of "frequencies of frequencies". This is the number of occurrences of all word frequencies. As a concrete example, assume the 1-gram "cat" appeared 10 times in the corpus. Also, assume that the words "dog", "flower", and "automobile" appeared 10 times in the corpus. The frequency of frequency for k = 10 is the value four – the frequency 10 appeared four times. Hence, the term frequency of frequencies. According to Jurafsky and Martin, "we examine Nc, the number of N-grams that occur c times. We refer to the number of N-grams that occur c times as the frequency of frequency c" (2000, p. 212). A visualization of these frequencies is provided in Figure 2.

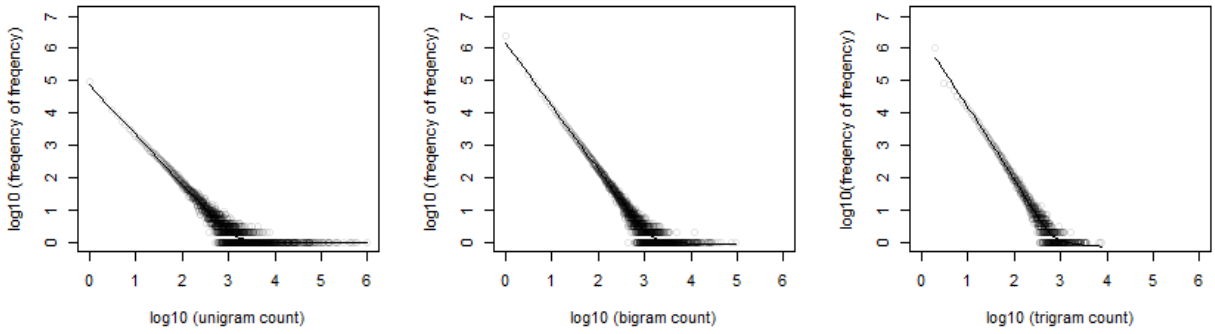

Figure 2: Frequencies of Frequencies of Various N-Grams

The frequencies of frequencies view of the corpus is an important feature for N-Grams because these counts enable predictions of other words. The frequencies reduce rapidly comparing unigrams to trigrams. Information from the literature indicates that these frequency tables respond well to regression analysis to smooth out the information and avoid having to keep all the data in storage.

STATISTICAL MODELING

The overall method used two generate the first model is based a technique of taking the full corpus and breaking it up into a training set, a developmental test set, and a test set (terms from Jurafsky and Martin). The objective of the modeling phase is to create a model that balances accuracy with speed and scalability given that we expect a very large data set in the working corpus. Algorithms of the initial model are revised to improve either accuracy, speed, or both if possible.

Predictive modeling

As noted in the literature review, the strategy selected was to build an N-Gram model augmented with Good Turing Smoothing methods. Initially, it appeared a full Markov matrix would be necessary; rather, the Markovian properties afforded by N-Gram modeling were of true interest. In other words, it does not matter the length of the phrase – one can predict a word on a trigram, bigram, or even unigram (Jurafsky and Martin, 2000).
Continuing with the cleaned and transformed corpus, it was tokenized using the `tau` package in the R programming environment (Buchta, Hornik, Feinerer, & Meyer, 2014). N-Grams were generated for (N:n=1, 2, 3). This proved to be a very efficient way to store the data because it collapsed entire matrices into small number of columns (between three and five), albeit having very long numbers of observations. To give an idea of the number of unique N-Grams, Table 4 shows the number of observations for the 1-, 2-, 3-Gram models.

Table 4: Relative Size of Observations from Various N-Grams

N-Gram	Instances
1-Gram	198,576
2-Gram	3,525,643
3-Gram	17,021,221

Having these N-Gram models in dataframes, a Good-Turing matrix was generated. As noted in Jurafsky and Martin (2000), any N-gram with a frequency less than (K:k=5) is an ideal candidate to adjust the count using Good-Turing methods. N-Grams appearing six or more times are seen to be realistic and do not require smoothing. The power of the Good-Turing method is that it allows us to calculate a probability of the unseen N-Gram. In order to do this it must discount the probabilities of the seen N-Grams in order for the probabilities to add up to one. As mentioned above, discounting in this predictive model was applied to all frequencies of frequencies between the numbers one in five. Those discounted probabilities accumulate and go towards the probability that the N-Gram appears zero times. Table 5 shows the Good-Turing Smoothing matrix for this model.

Table 5: Good Turing Smoothing Matrix for Predictive Model

count	uni	bi	tri
0	1.28	4.74E-05	7.81E-10
1	0.30	0.24	0.17
2	1.32	1.15	0.22
3	2.39	2.13	3.44
4	3.36	3.13	3.21
5	4.30	4.15	4.29

The first prototype of the predictive model was run against test data. At this point, Katz back off approaches were not implemented in the algorithms. The basic flow of the user interface with the algorithm follows this process:

- the user inputs two words (a 2-gram) to the `predict` function – with the goal of generating most likely third words
- the 2-gram is searched for in the trigram model look-up table
- if that 2-gram does not exist, a simple message appears that the word did not exist in the look-up table (in prototype only)
- if the bigram is found, a subset of all 3-grams starting with that 2-gram are generated
- the algorithm next uses the Good-Turing Smoothing model to update the count for each element of the subset where (K:k < 5)
- the probability of the resulting trigram options is calculated by dividing the count of the trigram by the count of the bigram

The algorithm allowed for an accumulation of possible third words within 10 clusters (fixed during prototyping). During early runs of the algorithm with very a small corpus, there were many times when the probability of the most likely third word had up to 14 different options. As the corpus used in the model grew, the stratification of these probabilities was expected to become more granular. It was determined that later iterations of the model would allow the user to adjust the number of clusters presented in the results.

Having created this first predictive model it was run against a test corpus. The early predictive model – using a small corpus of 30,000 articles and documents – resulted in an accuracy of about 20 percent. This compares to the accuracy of SwiftKey at approximately 30 to 40 percent.

Creative exploration

Having a prototype allows for creativity and practicality to merge. First, the corpus used was increased to the full training set representing 60 percent of blog posts. Very quickly – and unfortunately – it was clear the preprocessing to develop the N-Gram matrices was not scalable. In other words, the number of observations (rows) vastly exceeded the physical RAM capabilities of 4 GB of the development platform used in this project. This required an improvement in the algorithmic approach to improve the basic preprocessing. However, it was believed the prediction algorithm would still run as efficiently once the N-Gram models are brought into proper size. Revisions to the prototype data cleaning were implemented to include end-of-sentence and number tags, conversion of various ASCII codes to appropriate language wordforms, removal of punctuation except apostrophes and < > symbols, and removal of web URLS. These adaptations had the desired effect. The 60% training blog set could be fully processed within the R statistical environment.

Another very important change was in developing the trigram table. During processing, that table exceeded the physical RAM memory. The large number of single trigrams was removed from the dataframe. Although the results of this were seen in Figures 1 and 2, the singletons were not removed until after prototyping. Upon inspection, most of the singletons were very rare and odd combinations of words. The algorithm for preprocessing was adjusted to find all the single trigrams frequencies and remove them from the table. This process was not straightforward. One had to keep RAM limitations in mind. Therefore, it was accomplished in a series of iterative chunks – taking a subset of 10,000 documents at a time, processing them, and then using the `aggregate`, `merge`, and `rbind` functions of R. For the 60 percent training corpus, this required 54 "for loops" and generated 54 individual data frames.

The `merge` function aggregated those 54 dataframes to accumulate all of the counts in a way that avoided losing a single trigram if it appeared in one data frame. This approach is similar to the MapReduce approach used in Hadoop style data processing. Once all the data frames were consolidated – and consolidated – and consolidated again, the total number of single trigrams exceeds 15,000,000 in number. Overall, this process reduced the final dataframe of trigrams for the model to just 1,464,904 observations. Reducing the dataframe increased speed and scalability. As for accuracy, this update model was run against the test sets and it increased accuracy to approximately 40 percent.

One last modification is worth discussing. The size of the three dataframes was a concern when loading it into a web-based environment. The initial and revised model used a design relying on a two-gram and a three-gram dataframe. This amounted to 170.8 MB of data. Some experimentation was conducted to discern the effect of adding end of sentence markers. Using them in the data processing had a significant effect and reduced the table sizes by 20 to 30 percent in terms of overall numbers of observations. An idea then emerged to eliminate the two-gram table from the model all together. It served as a look-up table in the initial prototype to determine if there were any 3-grams in the model. Later, it was realized that the three-gram model already contained all the 2-grams and 1-grams within its structure. The reduced dataframe was loaded on the web server. At runtime, a simple `grep` command using regular expression searching extracted the particular bigram and unigram from the dataframe. The Katz back method was added to the algorithm to compensate for the elimination of single trigrams. This eliminated programmed error-checking messages seen in prototyping that "no options could be provided" to the user. All told, these enhancements reduced the web-based model 75 percent to a much smaller 42 MB CSV file that proved scalable on all of the laptops, desktops, and mobile platforms tested during development.

REPRODUCIBLE DOCUMENTATION

All of the data processing, analysis, and results are fully reproducible by executing the code provided in the project repository. It is an R markdown file available as open source at https://github.com/jgendron/datasciencecoursera. Those interested in collaborating on this or similar projects may contact the author at LinkedIn: jaygendron.

RESULTS – CREATING A DATA PRODUCT

All told, enhancements to the initial predictive model reduce the web-based model to a 42 MB CSV file that is scalable on various devices. Once loaded, the methods used to access the data frame allow the model to load in less than two minutes. After the initial load, subsequent queries of the model provide near instantaneous results to the user after they provide two words as input. Results of the data product present a list of possible "third words" to the complete the two-word phrase entered. Additional controls within the data product allow the user to adjust the amount of information they see. The interactive web-based model is available at the Shiny.io web site under the link https://jgendron.shinyapps.io/predictngram/.

CONCLUSIONS

In this analysis, many observations were made of common relationships between the various sub-corpus elements, the vocabulary, and word types. These enabled better understanding of their impacts on predictive power of a model. Using information learned during exploratory data analysis, the algorithms were refined to generate look-up tables that balanced depth of information with speed of processing. There was a subset of the corpus (specifically, 60 percent of the blog corpus) that produced an optimized balance between these competing characteristics. The Good-Turing approach was employed to develop the tables of counts, which enabled prediction. The Katz Back off approach was implemented in the event a particular three-word phrase was not found in the look-up table. This analysis suggests that although a predictive model can be built, the data product is most useful for predicting common word stems as opposed to highly specialized language needs of a user. Next steps for this exploaratory work include refinement of the web-based data product to allow more robust sensemaking of user input. Concretely, enable a Google style "spell checker" that completes user inputs based on the first few letters input. Additionally, the model itself is a proof of concept. It provides a working example to gather other training sets and parametrically tune the model to find an optimal balance between accuracy and computational performance.

ACKNOWLEDGEMENTS

First, I would very much like to thank my wife, Cindy, for all her understanding and support during many weeks and long evenings of optimizing algorithms. She is *amazing* and for that, I am eternally grateful. Thanks also to Dr. Thomas Bock for his explanation of key mathematical concepts. Lastly, I would be remiss if I did not acknowledge the power of collaborative group dynamics. Many of the ideas and solutions found in this paper are rooted in the generous comments from my colleagues on the Data Specialization Discussion Forum.

REFERENCES

Buchta, C., Hornik, K, Feinerer, I., & Meyer, D. (2014, June 11). tau: Text analysis utilities (Version 0.0-18) [Software]. Available from http://cran.r-project.org/web/packages/tau/index.html.

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

Feinerer, I. Hornik, K., & Artifex Software. (2014, June 11). tm: Text mining package (Version 0.6) [Software]. Available from http://cran.r-project.org/web/packages/tm/index.html.

Feinerer, I., Hornik, K., & Meyer, D. (2008, March). Text mining infrastructure in R. *Journal of Statistical Computing*, 25(5). Retrieved from http://www.jstatsoft.org/v25/i05/paper.

Gendron, G.R. (2014). *Word prediction R markdown file* [Data file]. Available from https://github.com/jgendron/datasciencecoursera.

Jurafsky, D. & Martin, J.H. (2000). Speech and language processing: An introduction to natural language processing, computational linguistics and speech recognition. Englewood Cliffs, NJ: Prentice Hall.

R Core Team (2014, July 10). R: A language and environment for statistical computing. R Foundation for Statistical Computing (Version 3.1.1) [Software]. Vienna, Austria. Available from http://www.R-project.org/.

Richards, B. (1987). Type/token ratios: What do they really tell us? *Journal of Child Language*, 14, pp. 201209. Doi: 10:1017/S0305000900012885. Retrieved from http://hfroehlich.files.wordpress.com/2013/05/s0305000900012885a.pdf.

Turing, A.M. (1950). Computing machinery and intelligence. *Mind*, 59, 433-460. Retrieved from http://www.loebner.net/Prizef/TuringArticle.html.