*This report was written as part of the requirements for SENG 474 / CSC 578D: Data Mining, offered at the University of Victoria, taught by Dr. Alona Fyshe.

Clickbait Generator – Final Report

(10 Ways to Get an A+)

Dahv Reinhart
V00735279
dahvreinhart@hotmail.com

Chris Carr
V00782360
ccarreng@gmail.com

Andrew Fowler
V00715255
addy.mcfowfow@gmail.com

Trison Nguyen
V00178742
trison.ng@gmail.com

ABSTRACT– Our project is focused on creating a machine learning algorithm for the automated generation of 'clickbait' article titles. We scraped 3618 Buzzfeed super titles and used them to train our algorithm. When human test subjects were shown one of the generated titles alongside a real title pulled from our dataset and asked which they though was computationally generated, there was a median error of 29%. This result shows the algorithm's ability to impersonate a human author which hints at its level of intelligent text generation.

***PROBLEM DESCRIPTION AND MOTIVATION* –** Our project focuses on two main goals: to create a dataset of scraped Buzzfeed articles and to use this dataset to algorithmically generate de-novo article titles. The motivation for this investigation revolves primarily around two ideas. First, there are monetary incentives for the existence of such a tool. If we succeed in teaching a computer to generate titles, then there is no need to pay human authors to do the same work. Furthermore, titles can be generated at a much faster pace and, thus, ad revenue can be increased. Second, we observed a distinct lack of previous work having been done in this particular area. This remains an unsolved problem. The few instances of previous work will be discussed in the *Related Work* section.

Given that there is a lack of previous work done on this topic, this project has several interesting possible outcomes. The ability to generate meaningful text using computational tools enables human authors to focus on other, more creative endeavors. This is especially salient in the arena of news articles and article titles where the writing is often formulaic and tedious. Similarly, other monotonous forms of writing, such as government policy, legal documents, song writing and movie script writing could also be automated. Indeed, any literature that follows a somewhat rigid structure could be automated.

Looking even further, text generation has applications in areas such as artificial intelligence, game programming and computational interpretation of real world stimuli (context awareness of images etc…). In essence, good text generation tools are needed if a computer is to express itself fully.

***RELATED WORK* –** As aforementioned, there is a lack of examples of previous work in this area. We were able to find three main instances of people trying to accomplish similar tasks.

First, we found the blog of Lars, a programmer from Norway [1]. In 2015, Lars attempted to use a neural network to generate click-bait articles with a modicum of success. To do this, he trained on ~2 million articles. This was the only instance of click-bait title generation in particular that we could find.

Second, a company called Narrative Science tried to computationally generate full news and sports articles[2]. Unfortunately, Narrative science is a private company who does not open source their software. Thus, we could not study in any real depth their approach to the problem of computational text generation. However, their slogan offers a glimpse into what it is they do: "Our technology generates data-driven narratives that explain, amplify and illuminate significant events and outcomes. Our mission is to enhance human productivity and make people smarter."[2]

Lastly, we found a publication in the MIT Technology Review about a Finnish team who tried to generate rap lyrics[3]. Titled 'DeepBeat', this study aimed to generate grammatically correct poetry in the style of rap music[3]. Their results are quite interesting with the algorithm being able to generate rhyming lines that have very interesting language. However, its ability to remain coherent throughout a stanza left something to be desired. To train the algorithm, 10000 songs were used for learning.

***OUR APPROACH* –** The first step was to scrape Buzzfeed to obtain a MongoDB database of articles and article titles. Specifically, we scraped the list of Buzzfeed superlist articles. These are articles in the "listicle" format which consists of a title and several subtitles. Together with photos, this comprises a typical Buzzfeed superlist article. After the first iteration of scraping, we obtained ~3900 titles and ~39000 subtitles.

Before the scraped data could be put into the database, it had to be sanitized somewhat. Namely, there were non-english posts that had to be removed. To get rid of these, we used the *LangDetect* python library. This brought the final size of the database to 3618 titles.

Step two was tokenizing and tagging our titles. This, as with most of the textual classification, was accomplished using the *Natural Language Tool Kit* (NLTK) available as a python library. When tagging, we used the built in Part of Speech (POS) tagger within the NLTK. After this process was run, each of the words in our database was paired with its POS. This was used in the upcoming automated text generation step. This process took longer than we thought, with each run taking ~4 hours to complete.

Step three was to build a model for title generation. Originally, we attempted to create a custom Context Free Grammar (CFG) to describe all of our possible titles. We quickly realized that this approach would not be fruitful as the titles being generated were grammatically unsound. This is because, when using a CFG, one must handle edge cases using explicit grammar rules. In our case, we would have had to implement too many of these custom rules to make this a feasible approach. Discarding our CFG, we opted for a n-gram, specifically bigram, model. Here, we generated all the bigrams (pairs of words that occur together within a given title) in our dataset and, using these, generated a frequency distribution (FD). The FD for a given word consists of the number of occurrences where each other word follows the given word. Using such a structure, one can obtain a list of which words are the most popular successors to any given word. This was then used to generate a new title.

The fourth and final step was to generate a new title. To begin the process, we chose a 'seed' word and a length cutoff. The 'seed' word was chosen by randomly selecting a start word from an article in our dataset. The usual output of this was a number but not always. The length cutoff was chosen by randomly selecting a number of words ranging from the length of the shortest title in the dataset to the longest. Using the previously discussed FD, we would then choose a successor to the seed word by randomly selecting one of the top three words in the FD of that seed word. At this point, our title would have two words in it. This process was then repeated until the length cutoff was achieved.

Once we had a generated title, we checked its viability against a smattering of heuristics meant to assess readability. These included:

- Checking that the title did not end on a stop word from the NLTK's Stop Word Corpus (I, me, my, then, with etc…)
- Checking that the title did not end on a forbidden POS (preposition, number, etc…)
- Checking that the title was not a complete match to a real title in the dataset (making sure it was in fact a new title)
- Making sure the title had the correct length
- Making sure the POS flow of the title was viable (match cutoff)

The last bullet above requires further explaining. When ensuring correct title flow, we take the sequence of POS tags of the generated title and see if there are any titles in the dataset that are a sufficient match to this ordering. For our algorithm we put the similarity requirement (match cutoff) at 80%. This ensures that, although the words and concepts in the generated title may be novel, the textual flow and grammatical ordering of the words is somewhat proven to work (already existing in the dataset). This greatly improves the readability of the generated text.

If the generated title, in any way, failed the above heuristics, it was thrown away and a completely new title was generated. This process runs in a loop structure. Thus, once a title is returned by the function, we can be confidant that it satisfies each and every heuristic.

One last problem we encountered was that the sentences were not logically making sense. Although a title would fulfill all heuristics, the concepts within the title were often dissimilar and not complimentary. We found that the source of the problem was the number of dataset titles we were training on. In the original version of our generate() function, we evaluated the match cutoff using all 3618 dataset titles. This was giving the algorithm too much freedom with which to match potentially illogical titles to. To circumnavigate this, we began generating titles based on a randomly selected slice of 100 dataset titles. This constrained the match cutoff heuristic so that it was forced to throw away many more titles, thus, narrowing the possibly accepted titles. Since the slices were randomly chosen and were re-chosen every call to the generate() function, the determinability of our titles was not harmed. In fact we saw an increase in possible contextual breadth. Also, the time it took the algorithm to run was shortened and the overall readability of the titles was greatly improved.

RESULTS – Overall, the project was a success. We were able to create novel titles which obeyed the rules of English grammar and which made contextual sense. Furthermore, the dataset we created was instrumental in this success.

Despite our overall success, there are some recurrent problems in the algorithm. Despite our best efforts (outlined above), the generated title still sometimes ends on words which have no logical reason to be at the end of a sentence. This can be for several reasons. First, the NLTK POS tagger is not perfect. Mislabeling does occur and this can cause a word to fill a POS niche that is not concurrent with its normal place in everyday speech. An example of this is that many words, if capitalized, will get labeled as proper nouns, even if they are clearly not. To get around this, we down-cased all our words prior to tagging them, however, mislabeling like this still occurs to a small degree. Second, ending on illogical words can stem from the fact that not every member of a POS can be used to end a sentence. This is impossible to get around without taking into account every single word in the English language which we do not have the resources to do.

Another problem, despite our match cutoff heuristic, is that the sentences still sometimes do not make contextual sense. Due to the FD approach we take when generating titles, this is always a risk. It is unlikely that this could be remedied using the approach we are using. Other algorithmic structures, such as neural networks, perform better in generating coherent context but due to our lack of knowledge on that subject we were unable to explore such options.

Punctuation and plurality were other areas of consternation. Since our algorithm had a difficult time with generating punctuation in the correct places, we decided to simply delete most punctuation from the tokenized dataset titles. We elected to keep apostrophes and hyphens because these marks often link two ‘halves’ of the same word and these two halves must be present in order for that particular vocabulary to make grammatical sense. Plurality was also an issue in that sometimes words such as ‘is’ or ‘are’ that extend expectation upon succeeding words are sometimes out of place. After all, a title such as ‘The dogs <u>is</u> gray and black’ is not a logical sentence.

The last major area of literary incorrectness was in the ‘flow’ of the title and its relation to the title length. The ‘flow’ of the title is comprised of the vocabulary and how it relates to a certain set of ideas. In a correct sentence, the physical length of the sentence allows for the complete expression of the concepts within that sentence. However, in our titles, this balance is sometimes violated. For example we often had titles such as, ’14 things you shouldn’t wear when thinking about’. Here, although the title is (so far) grammatically correct, the length does not match the flow. If there were two or three more words allowed to be in the sentence, it may have worked out. This is a product of how we choose our sentence length. We opt to do it randomly (as outlined above) which increases variability between sentences but gives rise to problems with flow.

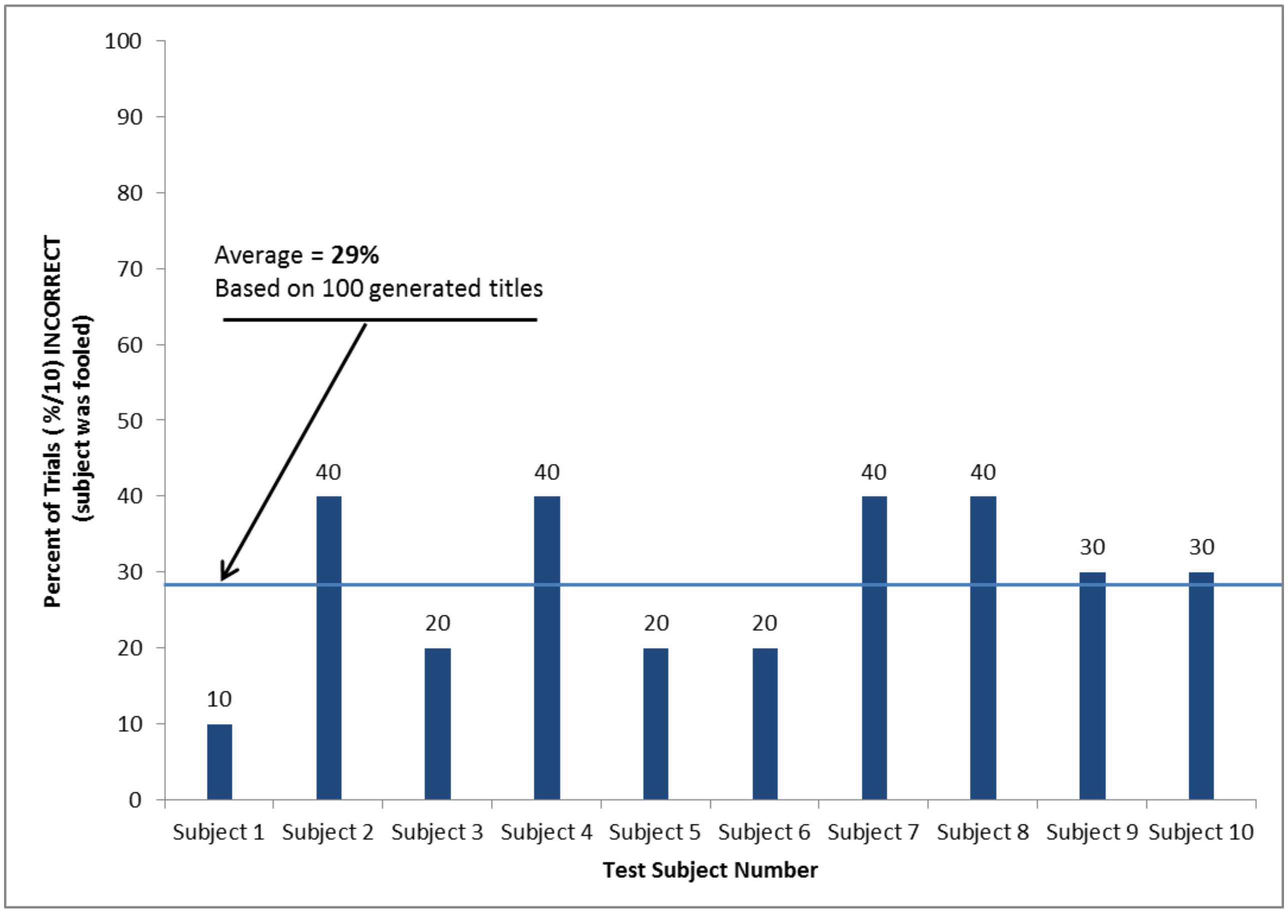
Overall, the problems discussed above were eventualities brought about by the lack of objective validation tools available to us. Since there has been a lack of past work on de-novo textual generation, so too has there been a lack of development of tools to analyze textual generation. Thus, we had very little outside tools to assess if our sentences were ‘correct’ or not. If we had such tools, they could have easily been incorporated during the selection process in our generate() function. Many of the nonsensical titles we got may have been stripped out if this were the case. Unfortunately, we had to develop all the readability tools ourselves. This would be a ripe area for future research seeing as how if good tools were available, automated text generation would be much easier to perfect.

***

Originally, we had scraped entire Buzzfeed superlist articles. However, we ended up not using the subtitles, opting instead to only train based on the super-titles. Thus, the subtitle features in our dataset went unused, although, in future iterations of the algorithm, these could be taken into account in order to generate subtitles or more verbose text.

Noticing we had a lack of objective assessment of our algorithm, we sought to obtain some. This was done by first, creating a sample webpage which, upon activation, generated and printed one of our titles along with a title that was pulled directly from the dataset. A sample group of 10 friends, family members and co-workers was then asked to choose between the two titles and guess which one we had generated. Each test subject underwent 10 trials. The correctness or their responses was then recorded and analyzed. A graph of the data is shown below:

<u>Figure 1</u>: Human-Based Validation Responses to Generated vs. Real Article Titles



As is visible in the graph above, the average percent error of the test subjects was 29%. Note that this is the parameter we were trying to optimize. Also note that this percent error could not meaningfully exceed 50% since 50% would denote a random choice. We are happy with this outcome as it shows that the algorithm has a solid ability to fool humans. This alludes to the level of grammatical correctness and high level conceptual cohesion that the generated titles portray. During this data collection, there was a tendency for test subjects to catch on to the specific 'flavor' of title that the algorithm generated. Thus, the test subject's accuracy began to increase the more titles they were shown.

CONCLUSION AND FUTURE WORK –

This project has shown us the difficulty of generating readable and grammatically correct text. At the outset we greatly overestimated the amount of tools available to us to aid in the completion of our goals. In accordance with this, it is much clearer to us now the amount of edge cases that must be taken into account when generating text. Originally, we thought that simply designing the generation method was going to take the most time. However, now, it is clear that the bulk of the time and effort lies within making sure a generated sentence is correct. This is also an area that should be explored more. Validation tools are an important part of any endeavor as they allow you to know what is going wrong. It was often hard for us to realize what was causing a certain problem. This made it similarly difficult to fix the issue.

If we had more time to design our algorithm, there are some additional features that we would want to implement.

A bigger dataset would enable the algorithm to choose from a bigger pool of possible words. Also of note is that this additional data would not have to come from clickbait article titles. The inclusion of a plaintext corpus such as the Gutenberg Corpus would enable the algorithm to generate titles with more vernacular freedom, the results of which would be very interesting to see.

Although we decided not to use neural networks, this approach has worked in the past. Of particular significance is Google's research into picture captioning. (We couldn't find anything on this before starting our project; hence it did not influence our problem description.) At any rate, it would be an interesting method to try as it is said that neural nets are quite good at maintaining contextual flow through a given piece of generated text.

The last idea that we wanted to explore involved differing n-gram structures. Bigrams seemed to work quite well, although trying to use trigrams, higher order n-grams, or a combination of all three might prove useful.

TASK BREAKDOWN -

Andrew: scraping Buzzfeed data, MongoDB database creation, code implementation of context free grammar generator, implementation of n-gram/FD model, evaluation of generated text using computational methods, refactoring code, website demo, report writing

Dahv: code implementation of context free grammar generator, implementation of n-gram/FD model, evaluation of generated text using computational methods, refactoring code, website demo, report writing

Chris: code implementation of context free grammar generator, implementation of n-gram/FD model, evaluation of generated text using computational methods, refactoring code, website demo, report writing

Trison: code implementation of context free grammar generator, implementation of n-gram/FD model, evaluation of generated text using computational methods, refactoring code, website demo, report writing

***

Our division of responsibility was designed such that everyone had a hand in all aspects of the development process. For creating our algorithm, we would all meet, gather around one computer, and discuss ideas. Thus, no one person was responsible for doing something that the others were not also involved in. This is why there is so much repetition in the duties held by each member above. We feel this strengthened our development process.

BIBLIOGRAPHY –

1. **Lars Eidnes.** 2015. Auto-Generating ClickBait with Recurrant Neural Networks. Available from: http://larseidnes.com/2015/10/13/auto-generating-clickbait-with-recurrent-neural-networks/

2. **Narrative Science.** 2012. Formal article title unavailable. Available from: https://www.narrativescience.com/

3. **Eric Malmi.** 2015. Machine-Learning Algorithm Mines Rap Lyrics, Then Writes Its Own. MIT Technology Review. Available from: https://www.technologyreview.com/s/537716/machine-learning-algorithm-mines-rap-lyrics-then-writes-its-own/