Using Natural Language Processing to Communicate Ideas More Clearly in Writing

Giovanni Flores

Department of Computer Science

**Abstract**

Word-processing software such as Microsoft Word and Google Docs have become the standard when it comes to expressing ideas digitally. However, more often than not, most of what’s written in a rough draft for any sort public document includes a plethora of “fluff”: filler words, run-on sentences, and nonsensical phrases. This is a problem that goes beyond the spell and grammar check capabilities of a text processor. As a result, writers of any kind, such as students and journalists, often have to go back to eliminate or modify redundant sentences in their works. In order to reduce the time that writers spend eliminating redundant sentences and paragraphs, this study explores and investigates algorithms that will parse a corpus to cut off filler content without changing the meaning of the original work and quantifies their effectiveness by counting the amount of filler content removed per each pass of the program. This process is implemented using the Natural Language Tool Kit, a collection of Natural Language Processing (NLP) libraries written in the Python programming language.

**Introduction:**

The importance of Natural Language Processing in software goes oftentimes unrecognized during people’s daily searches throughout the web. What’s important to note is that, without the existence of NLP (Natural Language Processing), inferred online dictionaries couldn’t exist as we know them, there would be no spell check, and intelligible corpora (bodies of text) couldn’t be pseudo randomly generated by a machine. Text processing wouldn’t have gone from specialized machinery from back in the 1960’s to a graphical user interface used for editing. The original idea for this study came from the need to extend the existing capabilities of formatting small text documents.

For the purposes of this study, the descriptions and methodology of this proposal will be focused on keeping the technical details as minimal as possible without changing the overall message of what the study is trying to accomplish. Any algorithm will be explained in simple English and will not get into unnecessary details that may puzzle readers of this work. The algorithm will be created based on the results of this study. The results will come about by collecting data using old rough drafts that I have collected.

**Literature Review**

***Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering***

In this study, the researchers identify the problem that currently exists in the field of Natural Language Processing, that being the need to work with lots of data, and the constant need to optimize algorithms. Drawing from other sources, the researchers deduce that they are going to use what are called LSH (Locality Sensitive Hash) functions to work with larger datasets of nouns.

One of the key important parts of this study was the need for background information (i.e. the theory behind the algorithmic implementation). This is defined in this study by explaining that in order to use the calculation necessary for clustering, they must first find the cosine similarity, which is the cosine between two vectors.

Afterwards, the researchers delineate the steps they took to implement the cosine similarity defined in their theory section. This is listed in numerically short but thorough steps, a procedure that will be necessary for deleting filler content in this study.

This implementation is then tested on a 6-gigabyte newspaper text and is then tabulated to analyze not only the number of vectors (nouns in this case), but the error as well as the time. A similar procedure will go into place for removing filler content.

At the end of the study, this randomized algorithm approach managed to reduce the memory utilized as well as improve the performance from begin quadratic to linear. The researchers end by proposing that this approach can be used on larger documents across the web.

***Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization***

This study analyzes the different graph algorithms and their application to extracting sentences from text to provide synopses of the same. The introduction serves as an overview for what’s to come in the following paragraphs of the next article.

The first algorithm looked at is an algorithm called Hyperlinked Induced Topic Search (HITS) algorithm, which ranks pages on a website based on how many links are going in and how many going out and quantifying a value for each. Other algorithms discussed by the researchers are the Positional Power Function, which ranks by its successive vertices, as well as the famous PageRank algorithm that Google uses, which combines the links going in and the links going out into one score. This background information is important as they lay the foundation in this research’s own review, showing that these researchers in question did some background beforehand.

After establishing these different ways of ranking, the study applies similar methodologies to those used in the above algorithms to rank sentences and basing the connections between them based on the similarities between two sentences. This type of approach would be key to the study of removing filler content, as it presents an idea that could be used to originally base the algorithm by, grounding the study upon already completed research.

Once the ranking is complete, the sentences ranked the highest would compose the summary. For removing filler content, this approach could be extended and built upon to delete the least ranked sentences.

***Multidisciplinary Instruction with the Natural Language Toolkit***

Due to the rather peculiar mix of computer scientists and linguists studying and working on natural language processing, the creators of the NLTK open source library felt it necessary to create a toolkit that would bridge the gap and allow both parties to use this for tasks in the realm of NLP.

Although this article is in the context of teaching, it is quite pertinent to the project of removing filler content from text, as this is the primary tool that will be used to perform the various core NLP-related tasks.

The authors go on to state that simplicity was the priority of the design of NLTK, hence why they chose such a high-level language like Python to be the base language for the library. This was done as an intentional way to reduce the learning time of the actual language for newcomers, and instead focus on the concepts that could be applied with said language.

Due to this current study (the one pertaining to filler content) being tackled from mainly a computer-science approach, it’s important to understand that linguistics is the other half of fully using NLP. This means that NLTK was made with this in mind, as it starts going into how it can be used to show examples and exercises that are included in the NLTK documentation, which should be a point of reference to those using NLTK with Python.

Their research method include contrasting and comparing the distinct motivations of linguists and computer scientists, and seeing how there can be a middle ground, that being NLTK.

***BRAT: a Web-based Tool for NLP-assisted Text Annotation***

This study focuses on creating an annotation tool that will help out humans make comments on different corpora. This was originally another study that included most of the key annotation features sans the ability to edit annotations.

After introducing the basis and the need for this study, the researchers start talking about the implementation of the BRAT (the given name for this tool). They start discussing the Graphical User Interface and the various tools they will use to allow for a good user experience while performing the annotations. This relates to the current study being proposed since, to allow users to check what they want to delete, they will also need to have some way of viewing to see the highlighted text. The BRAT study made this visually appealing by including examples of annotations being made on example sentences.

Then, the tool was run through various studies that also had a human redactor going through what annotations would be made to ensure the effectiveness of the tool, results which were then tabulated to allow for analyzing of not only the annotation type but also the time it took. A similar approach will be taken when the current study seeks to start its implementation. The researchers end on the note that this tool was already used for various projects that prove that the implementation is effective.

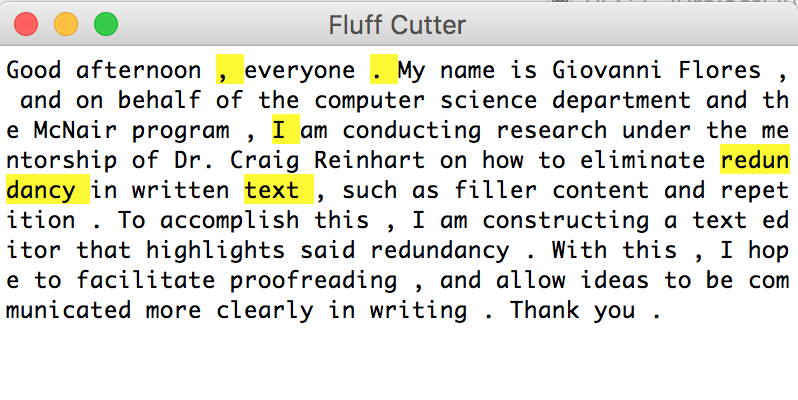
**Research Methods**

The timeline for this project was broken down into the span of five weeks during which the research took place.

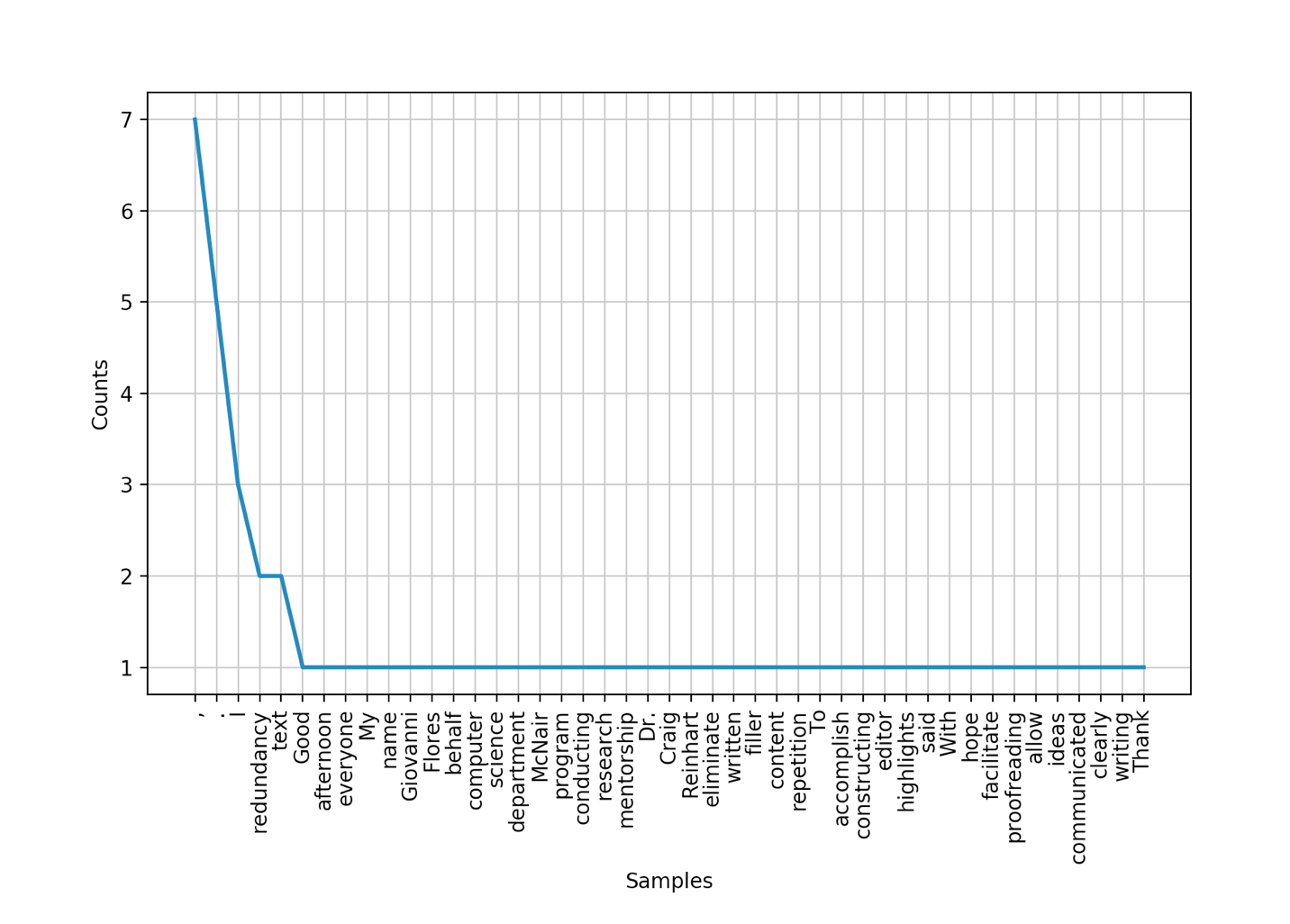
During week one, the current study first sought to begin chunking the different parts of speech of example rough drafts that were written beforehand to allow for examples to test on. The rough drafts came from online sources; for the purposes of this study, what matters is that non-prose was not collected. After identifying the parts of speech, week 1 also sought to label what parts concern a person, and what parts concern organizations that served as breakpoints for the algorithm to work with. However, this was ruled out due to knowing the parts of speech did not affect the algorithm in any meaningful way.

Once this was completed, in week 2 there was a period of time dedicated to building relations between the different parts of text in the drafts so that the algorithm could observe the sentiment of the page, and thus start eliminating repeated sentiments within a three-sentence radius. The reason for including this radius is due to the need to have a bound to recognize what has already been stated.

Starting from week 3, the algorithm was ported to a graphical user interface to allow for a visual representation of results. These results were also graphed to count the amount of words deleted. The amount of words deleted need to be collected to see if they make any actual changes to the already existing content of the rough draft at hand. This process was repeated as long as necessary during this part of the study. The following week (the fourth and last concerning research) was spent on expanding on the algorithm by applying more filters to what would display the results. This was done by including a highlighting mechanism, as shown below.



**Findings**



Above is a sample graph on the words that appear most frequently on a piece of sample, organic, unedited text from a speech. This data was gathered and spread out into one graph per corpus, with there being around 15 corpora used in this project. Noting that the highlighted words (the redundancy) was the same as the occurrences is concerning since this points to the fact that the algorithm is still not in working condition. On an average paragraph, a maximum of 5 words were highlighted. This is worrying since this shows that the current redundancy checker is not filtered enough to see context, even with n-gram implementation.

It is also of note that even after the initial check that the process works effectively, it is not enough to just accept the first result as the approach. A naïve approach is not enough, even if considering the text examples’ length to not exceed that of a few pages. This process will need to be optimized afterwards by timing and counting the number of passes done on the corpus (text) in order to allow for a quantitative, tabulated version of the words and phrases deleted by the algorithm. This process will continue through and by the end of the project until the desired results are achieved with the most effective implementation of the process.

**Conclusion**

A redundancy checker in a text editor allows for there to be numerous applications in everyday use. Students can use this to proofread a paper before they submit it, as well as college professors when doing the proofreading to see what can be cut off and still make sense of the paper in question.

The current version of this study has not reached that point yet. More background work is needed to include some machine learning models to aid with text prediction and analysis. The context of ML was limited to what the already existing NLTK and gensim suite of libraries could already do with their built-in capabilities. However, since this was not a primarily ML-focused study, there needs to be more work done around that.

The largest problem with going forward is the most fundamental truth about NLP and language in general, that being that language is intrinsically ambiguous. Even the most advanced of statistical models have to rely on limiting what can be predicted to come next in a piece of text in order to narrow down the possibilities. This creates more complications for this study due to there being an infinite amount of ways redundancy can occur in a piece of text.

An extension to this project (in the far future) would be to port this to work with other languages. However, there would need to be meetings and consent with language experts from foreign countries to even attempt to make this a possibility. As for now, this remains solely in English.

References

Ravichandran, Deepak, Patrick Pantel, and Eduard Hovy. "Randomized Algorithms and NLP." *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL 05*, 2005.

Mihalcea, Rada. "Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization." *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions -*, 2004.

Bird, Steven, Ewan Klein, Edward Loper, and Jason Baldridge. "Multidisciplinary Instruction with the Natural Language Toolkit." *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics - TeachCL 08*, 2008.

Stenetorp, Pontus, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. "BRAT; a Web-based Tool for NLP-assisted Text Annotation." Accessed April 10, 2018. https://www.aclweb.org/anthology/E/E12/E12-2021.pdf.