Daniel O’Brien

Multi-tiered text analysis

Text analysis is an increasingly researched field that has just a few barricades holding back its progress. One of these barricades is the intensive computational costs used by its processes. Enterprise-level systems are required to perform any sort of meaningful text-based analysis in a timely fashion. This means that useful text-analysis projects get pushed aside because they lack cost efficiency. The goal for this project is to explore potential efficiency gains that can be found by implementing a multi-tiered check-based analysis system.

Searching for efficiency gains in computer programs is not a matter of saving a few seconds for convenience sake. Companies have realized the economic value of data, and are scrambling to acquire, store, and analyze it to increase profits. Data is being stored and shared at a rate never seen before. In Data Science, more data always leads to a more accurate answer. As such, Companies who can afford it are purchasing incredibly large data sets and handing them off to data scientists to analyze. These Data Scientists are then left with the problem of efficiency. With such large data sources, inefficient functions can render a program useless.

Increasing efficiency of computer programs has always been a fundamental part of the Computer Science discipline. Optimizations are made through clever decisions about how data gets processed. Oftentimes, this involves the pre-processing, concurrent processing, and cleaning of data sets. Pre-processing is useful for applications that face large usage spikes but otherwise spend large amounts of time idle, such as electronic voting systems. Concurrent processing is useful in situations where shared data is important, such as when many people are trying to buy a single concert ticket, but it can also be used for programs that are run on large scale servers with plenty of resources available. Trimming data (the basis of this research) is best utilized when doing analysis against large sets of data.

This project explores potential efficiency gains for text-analysis tools by utilizing multi-tiered systems to trim data sets. Here, article titles are compared before any further analysis is done. The program then decides, based on the title, if the articles are similar enough to proceed with the analysis. If they are similar enough, the program then uses the articles’ sub-titles for the next tier of analysis. Finally, passing all of the preliminary tiers of analysis, the full articles are analyzed for similarity. This allows the program to identify dissimilar articles using minimal computational resources.

Boosting the computational efficiency of text-analysis modules has strong implications for where these modules can be leveraged. One possible application of this system can be found in University credit transfers. When a student transfers colleges, the school they are transferring into must decide which of the student’s credits will transfer. This process involves manually comparing course titles, sub-titles, and descriptions to decide if the proper material was covered. Using a multi-tiered approach as discussed in this paper would allow universities to q efficiently compare the course descriptions submitted by its transfer students. The results can then be used as a guide to help improve speed and accuracy for comparing course descriptions.

Another possible application for a multi-tiered text-analysis module is looking for patent infringements. Many companies encourage their employees to identify and file patents. These companies are then left with large amounts of patents that they must make sure are not being infringed upon. This is a tough task, especially with the tricky legalese that protect certain patents and doom others. By using a multi-tiered approach when comparing patents to product descriptions, companies can sift through tons of information without an unnecessary waste of man power. The product descriptions that make it past the analysis tiers can be handed off to lawyers to be manually inspected, while the dissimilar products are discarded without further need for analysis. This creates a huge boost in both computational and organizational efficiency, and would allow a company to be more vigilant when enforcing their patents.

The module created to conduct this research was made using Python. Python was chosen for its immense data science community and countless open-source packages that can be leveraged to perform data analysis. Also, python is a very versatile programming language. This allowed the program to be structured in a way that is easy to modify and maintain. This versatility was priceless for data gathering. This module utilizes packages for the acquisition, storage, manipulation, and analysis of data, as well as the handling of inputs and outputs from the program. These packages include BeautifulSoup, pandas, Doc2Vec, nltk, and sklearn.

One of the most important packages used in this module is Doc2Vec. Doc2Vec is an algorithm that is used to generate vectors for phrases, paragraphs, and documents. These vectors can then be compared against vectors created from other documents, and the difference between the two can be used to determine the similarity of the documents. Since the module only uses this one method for its analysis, the entirety of the research relies on the accuracy of doc2vec. Using a different analysis algorithm would greatly change the outcome of the results, even if the same dataset was used. Therefore, all results found by this research should be considered in the context of doc2vec based analysis.

Doc2Vec was chosen for its focus on semantics, as well as its flexibility. The vectors are generated by considering the context in which they are used, and not just the specific words used. This means the vectors are semantic representations of the document they were generated from. This is important because of the complexities found in natural language. For example, if one document speaks heavily on banks and stock markets, while another speaks heavily on financial institutions, these documents should be considered similar. However, if we only take the specific words used into account, then the analysis may show that they have little in common. This is unacceptable, as many false negatives would arise. Doc2Vec helps alleviate this problem by considering the context of the entire document. It can recognize that bank and financial institutions are often used in the same context, even if only a few words are shared between the documents. However, this relies heavily on the model built for the doc2vec analysis, and it can only perform accurately in contexts it was properly prepared for.

The doc2vec model for this module was built using articles in the Technology industry. This was done for the purpose of maintaining scope and ensuring accuracy in the data gathered. Building a model is computationally expensive, and the resulting file size is always quite large. This meant that training the model for a more general data set was not feasible. Also, for most applications of this module, a general model would not be needed. For example, if this module were to be used to identify patent infringements in the Technology sector, it would only need to be semantically aware of Technology based terminology. By focusing on one topic to build the model off of, we can maintain the accuracy of the tool while keeping a reasonable size and runtime for the module.