Project 1 Milestone 3

Business Problem

The advent of big data comes hand in hand with the growth of content. With that growth, some data can be analyzed by a machine, however for some purposes human consumption of the document is required. Recommendation engines help end users determine what results are relevant and useful from their search by ranking results and preforming other under-the-hood calculations to provide the most ‘relevant’ results. However, when the complexity of the subject matter of documents within search results increases so does the end user’s need for identification of the content within said documents. The objective of this project is to create a visualization of the subject matter contained within lengthy documents and thus allow end users to identify the main focuses of the document and where within the document they occur.

Background

When analyzing large text documents, understanding key features, their frequency and where in the document they occur can help an individual quickly determine a documents' relevance and sentiments. When on a web page control + F is often used to locate keywords, however this is an effective method to search across a database of document databases such as research articles or books. A document may be suggested to an end user but does not contain what is needed. Additional metrics and visualizations are needed to further streamline searches that require greater complexity.

Data Explanation

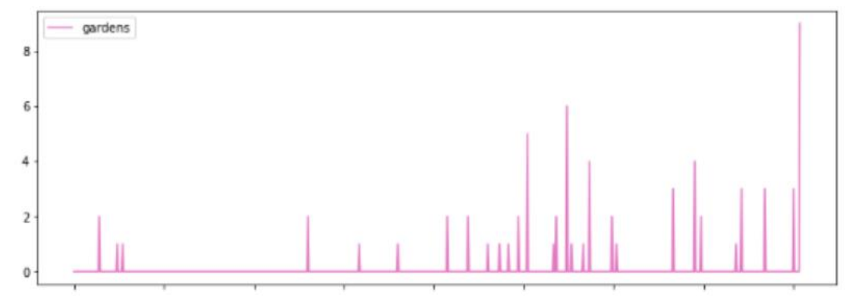
The data used in this project has been retrieved from a free online website that hosts web novels. This data spans over 800 text documents with each document containing a chapter of a novel. For copyright and ethical reasons, the words within the text files have been shuffled. This does not affect the overall metric as the shuffle is applied to each file individually. During the data cleaning process stop words are removed from the text and punctuation and symbols are stripped.

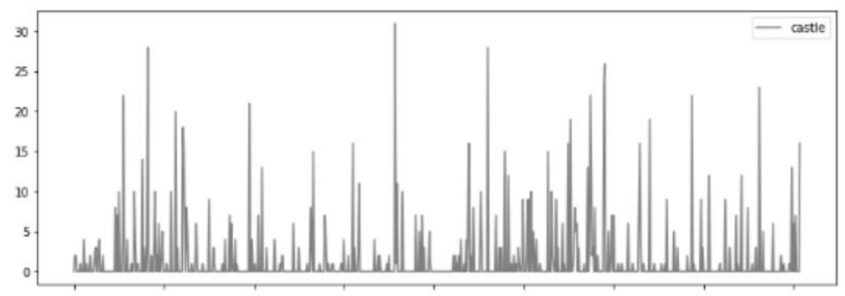
Methods and Analysis

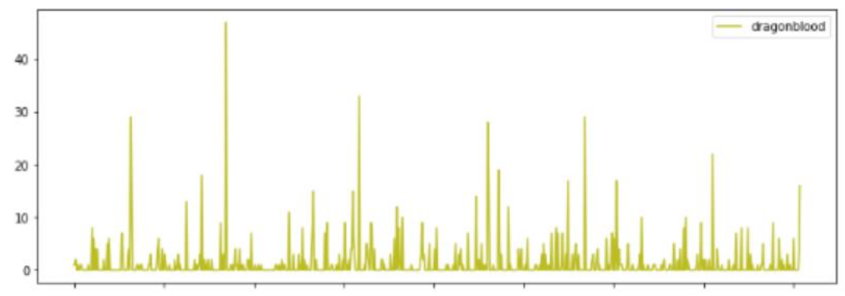
The data files are first read via Pyspark utilizing Apache Spark to create a document database. When each file is read into the system some basic manipulation is conducted to generate the tokens, raw features, and features of the data. This is then aggregated into the document database. For ease of manipulation the Pyspark SQL database is then converted to a Pandas Dataframe. A TF-IDF fit and transformation is then applied to the contents of each chapter of the overall document. Numpy is then used to retrieve the feature names as well as the frequency of the features within all documents. At this point the top 20 features are captured from this information. The top 20 features here now equates to the top 20 most frequently used words across all documents within the database.

Now that the top 20 most frequently used words are known we can then retrieve how many times they occur per chapter/document within the document database. This calculation is performed via the use of a Count Vectorizer applied to each chapter individually. Because we have already determined the top words within the book, we can pass this list to the count vectorizer as a vocabulary. This will allow us to discard all other values to save storage space and potentially processing power. The count of each term per document/chapter is then added to the document database.

Once the top 20 feature count per document is retrieved it can then be visualized. The visualization is created utilizing Matplotlib’s line chart within Pandas. Each term is then plotted separately with a legend to identify to which feature the plot belongs.







Use case

When an end user comes across a large document this model can be applied. The model will return a visualization of where and how many times that term occurs within the document. This allows a user to determine overall themes of the document as well as where within the document they may need to look to find the most data most relevant to their search. Alternatively, the model could be provided with a set of vocabulary and return the related information.

Assumptions and Limitations

The model used by default assumes that the subject of the text will be explicitly stated. If a topic is covered within a document briefly or indirectly referenced, this will be lost by this generalized approach. This could be overcome by specifying a vocabulary for the model rather than simply using the top x features.

One limitation of the model is the requirement for a machine-readable text document. The text must also be segmented or partitioned prior to modeling. Its efficacy can also be diluted if the document covers many subjects. If the text does not cover a single subject the term frequency-based approach is only able to identify subjects that are common across all documents. For example, if you apply the model to an academic journal. If the Journal covers a variety of one disciplines' subtopics the model may only identify that the document database is related to the overall discipline and thus miss the mark when looking to identify subtopics.

In this situation, if all articles often use a term like machine learning or AI the model will highlight this overarching term but not capture the more specific terms related to subsets of the individual articles within the document database. It would identify that the journal focuses on AI and machine learning. However, the end user may be more interested in the subtopics covered in the individual articles.

Challenges

This model is most useful when aggregating information from many long and verbose documents in a singular document database contained within a greater database of document databases. With this large-scale application comes a requirement to read in and process all of the data contained in, not only a document database, but each document database within the overarching database if used as a metric for a recommendation engine.

When utilizing the top 20 results for each document this can be computed prior when the document database is added to the content which the recommendation engine searches. If a set vocabulary is to be searched across all documents regardless of term frequency, the model would need to be applied on the spot to every article. In turn we can assume that scalability of this more complex application of the model will be challenging or possibly not worth the resource cost required to process this model.

Recommendations

We believe that the model is quite useful in efficiently conveying the overarching topics of a document and where within the document the end user may wish to review. The model would be most useful when computed prior to an end user's search. The addition of applying part of speech tags would greatly assist in filtering and optimizing results. Retention of the count vector of terms for each document database within the top-level database containing all document database would allow for searches to be performed efficiently when provided a custom vocabulary. This is because the recommendation engine would only need to generate visualization after searching for the terms within the count vector of each document database.