Project Update 1: Text Summarization Using NLP

Mark Trovinger
Purdue Fort Wayne
tromv01@pfw.edu

Atharva Atre
Purdue Fort Wayne
atreaa01@pfw.edu

Navyaprabha Rajappa
Purdue Fort Wayne
rajan02@pfw.edu

1 Results

Our first round of results were trained on the CNN/Daily Mail dataset mentioned in our project proposal. Due to the size of the data, and given that these results were generated using algorithms that are known to be less performant, the decision was made to only use the first 20 entries in the dataset. (Train- 287,113 Validation- 13,368 Test- 11,490). Results below are on the validation dataset only.

1.1 Current

Algorithm	f1 Score
spaCy	0.114012
HeapQ	0.13333
TextRank	0.2565866

As we can see from the table, the results are still in the preliminary stages.We expect siginificant improvements for Update 2.

1.2 Upcoming

We will be looking at three transformer-based models for our next project update round. The first of these models is BERT, which has been referenced in class, both in class content and in paper discussions. We are also interested in exploring a core keras library - KerasNLP, and explore options like BART, and other seq-to-seq models.

Similarly to our current results, we will be examining the results of fine-tuning the three transfomer models.

2 Analysis

2.1 Current

2.1.1 HeapQ

We looked at some of the advantages and disadvantes relating to each algorithm, the main advantages of HeapQ start with its suitability for processing large amounts of text. HeapQ is an efficient algorithm for maintaining a sorted list of items, and it allows us to quickly find the top N items in a collection. This makes it suitable for processing large amounts of text data.

HeapQ allows us to define the key for sorting items in the heap, which means we can use different criteria for ranking the importance of sentences in the text. This gives us flexibility in choosing the most appropriate method for text summarization based on the specific context.

The HeapQ algorithm is relatively simple to understand and implement, which makes it accessible to users with varying levels of programming experience.

We also examined some of the disadvantages related to HeapQ as well, such as a lack of context. The HeapQ algorithm only considers individual sentences in isolation, which means it may not capture the context of the surrounding text. This can lead to summaries that miss important nuances and connections between ideas.

Another downside of HeapQ is related to sentence selection. The HeapQ algorithm relies on selecting the top N sentences based on their importance score, which can be subjective and may not always capture the most relevant or interesting information in the text.

Further, there is a potential issue with the algorithm extracting incomplete sentences. Since HeapQ only selects a subset of sentences from the text, the resulting summary may not provide a complete representation of the original text, and important details may be lost.

Overall, using HeapQ for text summarization is a useful technique that can be effective in certain contexts, particularly for generating summaries from large volumes of text quickly and efficiently. However, it should be used in conjunction with other techniques to ensure that important context and details are not missed.

2.1.2 spaCy

SpaCy is a Python package focused on NLP, that implements a pipeline based infrastructure. spaCy has several advantages, namely that it can be installed with support for GPUs out of the box, which will allow us to train models much faster on GPU-equipped systems.

Another advantage of spaCy is that it excels at large-scale information extraction tasks. While the dataset we are using now is not at Google-scale, it is larger than most datasets we would come across.

The ecosystem of models and tools that can be added to a pipeline is a strong advantage of spaCy, as we can add pre-trained models into the pipeline and compare the results.

One of the disadvantages of spaCy is the pipeline paradigm can be difficult for those new to it to understand. This can make debugging somewhat more difficult, as a new user may be unfamiliar with how to engage with the paradigm.

2.1.3 TextRank

TextRank is an extractive summarization algorithm that is based on PageRank, a graph-based ranking algorithm used by Google to rank web pages in search results.

One of the advantages of the Text Rank algorithm is simplicity. TextRank is a simple and effective approach to extractive summarization that requires no training data or domain-specific knowledge. Text Rank is based on a well-known ranking algorithm (PageRank), which makes it easy to understand and implement.

Another advantage of TextRank is an ability to handle long documents and is langauge independent. In addition, Text Rank is an extractive summarization algorithm, which means that it only uses the existing sentences in the text to create the summary, making it more accurate and trustworthy.

While there are several advantages to the TextRank algorithm, there are also several disadvantages to using it, such as sentence similarity. Text Rank relies on the assumption that important sentences are similar to other important sentences, which may not always be true.

Another disadvantage of TextRank relates to readability. TextRank does not take into account the coherence or readability of the summary, which can result in summaries that are difficult to understand.

Compute requirements may also be a concern, as will the need to fine-tune and adjust the hyperparameters of the algorithm. TextRank can be slow and computationally expensive, especially for large documents.

In conclusion, Text Rank is a simple and effective algorithm for extractive summarization that can handle long documents and is language independent. However, it relies on the assumption of sentence similarity and may not capture the nuances and context of the text. It can also be slow and computationally expensive, requiring fine-tuning for optimal results. Nonetheless, Text Rank is a good starting point for text summarization and can be used as a baseline to compare more sophisticated summarization algorithms.

2.2 Upcoming

Given that the next phase of the project update involves models that use the transformer architecture, the main thrust of our analysis will involve the advantages and disadvantages with different implementations of this architecture.

3 Problems

3.1 spaCy on M2

One of our group member has a MacBook Pro with an M2 CPU, and we ran into compatibility issues with the Python library spaCy. While this issue did not keep us from being able to generate results, it does highlight an issue that can emerge when you have so many different processor architectures and versions of open source software.

3.2 Compute Resources

An issue that will keep recurring as we work on this project is the lack of compute resources on campus. While one of our group members has a GPU-equipped workstation, this will likely be a bottleneck as we start working on larger transformer models.

4 Plan of Action - Update 2

- Final F-1 Score analysis on our three preliminary algorithms (SpaCy, TextRank, HeapQ)
- Explore transformer based models (BERT, BART etc)
- Automated pre-processing pipeline setup for all of our algorithms /models.
- Exploring RoGUE and BLUE metric along with f-1 scores.
- Have all our analysis on github (current analysis already present) - `https://github.com/atharvapurdue/text_summarization`