Final Report

Extractive Summarization of a Generalized News Corpus using BERT

1. Introduction

Text summarization is a well known and challenging task in natural language processing (NPL). It seeks to address the eternal need for the succinct and relevant compression of information required by many professionals to keep current with developments in their fields.The busy doctor or scientific researcher cannot possibly read all the new papers within their specialization and the busy investor or executive has no time to read news publications or trade journals from beginning to end.

Automatic summarization is typically divided into two strategy types: extractive and abstractive. Extractive summaries seek to identify the most relevant sentences in the text and concatenate these as the summary. Abstractive summarization attempts to identify the key ideas and express these in new words delivering a more flowing summation. Extractive strategies are a pure classification problem where the goal is to label sentences within the full text as either belonging to a summary or not. Approaches can either be supervised or unsupervised. TextRank is an example of a classic unsupervised algorithm based on Google's famous PageRank.

In this paper, we will take a supervised approach and look at two features sets with different baselines. In the first, we focus on how our models are able to discover the internal structure and relationships within the embedding representation of the news corpus. These results will be compared to TextRank as a baseline and the features set will be limited to the embeddings and their derivatives. The other approach relies on a well known result of summarization for news corpuses and adds the ordinal, or sequential, information to the features set. This follows from the leading sentence phenomena where the first few sentences do extremely well in summarizing the article. In fact, many news publishers follow this rule and explicitly release the first few lines as the article summary or teaser on their websites. A strategy of choosing the first three sentences (LEDE3) is known to be exceedingly difficult to beat on standard metrics (ROUGE). In this spirit, we will use LEDE3 as the baseline for this second approach.

While the second approach with additional information will likely lead to better scores, it is important to note that we would not expect these models to be generalizable to non-news datasets. They will likely rely on the ordering of sentences specific to news articles and not on the semantic and contextual relationship between the sentences and the article.

2. Database

One of the challenges of summarization is that evaluation (or training for supervised models) requires a database consisting of text and their human written or curated summaries. These so-called gold summaries are difficult to obtain in the wild and research tends to focus on news articles and scientific papers where these are the most readily available. The Abstract serves as the summary for scientific papers and newsrooms typically have a teaser or summary of their articles for online banners or updates. The typical metric for summarization

evaluation, borrowed from translation, is ROUGE. At a high level Rouge measures the n-gram overlap of words in the predicted summary and the gold summary.

While the canonical news corpus for summarization is the CNN / Daily Mail database, this project makes use of Cornell University's Newsroom database which is larger, more diverse and contains interesting metadata labels. The full dataset contains 1.3 million articles and summaries sources from 39 publications and features a wide array of domain subjects and summarization strategies. Cornell has developed a number of metrics to assign three strategy types to each summary: extractive, mixed or abstractive. These metrics and strategy classification is available in the database. Unfortunately, the domain subject is not labelled.

The database was obtained on request, after agreeing to certain copyrite conditions, from Cornell and is delivered in JSONL format across three files: training, development and testing. These files, in turn, contain the articles and summaries in string format and keys for the metadata such as url, archive, date, title, three metadata metrics in numeric format and then three further discretized 'bin' descriptions as categories.

3. Model Pipeline

3.1 Processing

The extractive summarization task will be addressed as a classification problem on the sentences of the article and, resultantly, the database will be limited to those article and summary pairs labelled *extractive*. The next step is to split these articles and summaries into sentences and clean them before obtaining the sentence embedding using BERT.

After processing the features space, we turn our attention to the classification labels. The extractive labels of the database are formed from a continuum and not all articles will be "fully" extractive. Resultanly, we calculate labels by identifying those article sentences with highest cosine similarity to each summary sentence, and labelling those with one. The rest are assigned zeros.

3.2 Modelling

We next establish a linear benchmark using logistic regression and baseline using the Lead3 strategy. Lead3 follows from the observation that the first three sentences of a news article typically do a good job as a summary. In fact, and this is evident in our dataset, many publications typically default to this. The Lead3 is a strategy that simply summarizes with the first three sentences and is proven to be an exceedingly difficult baseline to beat for extractive summarization evaluated with ROUGE scores. Following these baselines, we explore various neural net architectures for the classification problem using cross-validation.

The modelling process has two nuances. Firstly, the train-test split needs to occur at the document level so as to ensure we can calculate meaningful ROUGE scores and have sensible sequences to feed into the recurrent neural networks. Secondly, classification prediction is not the end of the story as every article needs a summary and, even after considering thresholds, there may be some that return empty. We need to define a strategy

regarding the minimum (and maybe maximum) number of sentences to return. After obtaining the in-summary sentences, we then concatenate these and score with ROUGE. All these nuances require custom functions: (i) train-test-split, (ii) retrieval and strategy specification for returning in summary sentence text, and (iii) custom evaluation function for ROUGE scoring.

4. Data Wrangling

The implementation of the code can be found on Github. Many of the steps have been written into multi-output or compound functions in order to apply vectorwise to the text and limit full passthroughs of the dataframe. The steps below are more schematic of logic than necessarily true to the order of implementation.

The dataset was truncated to 5,000 extractive articles with over 166,000 sentences.

4.1 Processing Features Set

Step 1: Convert the downloaded JSONL data into a pandas dataframe

The provided JSONL files contain the dictionary of data for each article-summary pair per line. These files are converted into a dataframe using the JSON library to read the file by line, appending the dictionary to a list and then wrapping in a dataframe.

Step 2: Filter for extractive summaries only

Given the project focus, we are only interested in the extractive summaries and all other article-summary pairs are filtered out using the summary strategy “density_bin” provided in the database.

Step 3: Split each article and summary into a list of sentences using spacy

The summaries and articles are read into the spacy tokenizer format using the large English model. These are then split into sentences using spacy’s *.sents* method. The spacy tokenizer format carries a lot of memory overhead and the list is converted to a list of strings before completing this step.

Step 4: Clean the lists by excluding sentences shorter than a defined minimum

Removal of stopwords, lemmatization and stemming are purposely ignored and this step solely focuses on removing very short sentences that would not be appropriate for the summary and ensuring that there are no empty items in the sentence list. The decision to ignore the other steps follow a more modern approach that these words and nuances provide semantic and contextual information that may be important for the NLP task. For example, the use of “and” versus “or” can give a sentence entirely different meanings. This is particularly true when using embeddings as sophisticated as BERT.

Step 5: Embed each sentence using BERT

Each sentence is embedded using the Hugging Face's sentence_transformer library and specifying the *'bert-base-nli-mean-tokens'* model.

Step 6: Add document label to each sentence
As discussed in model setup, sentences need to be assigned to the document in order to recreate summaries and evaluate using ROUGE. Document labels will also be required for train-test-split and when using RNN models. Document labels are set using the index number of the original dataframe and looped through each document sentence to form a list of document labels that correspond to each sentence.

Step 7: Add document mean as an additional feature to each sentence

While BERT is very good at embedding context in absolute space, we can hypothesize that summarization requires each sentence to have some knowledge about the article as a whole or its relative embedding to the other sentences. The latter is implicit when using RNNs but the use of a simple neural network and the linear models will require this to be manually fed into the model. This is the purpose of adding the article mean as a feature for each sentence. This is simply calculated as the mean of the sentence embeddings for the article.

Step 8: Add sentence number as and additional feature to each sentence

We have already mentioned the generic propensity of the leading sentences of news articles to contain more information relevant to the summary when discussing Lead3. Clearly, sentence order is important and adding a feature of the sentence number is a way to hardwire this information into the non-sequence input models. The sentence numbers are assigned to the sentences by looping through the sentence list for each article.

4.2 Processing Classification Labels

Step 10: Calculate Cosine Similarity between each summary sentence and the article sentences

In order to account for the fact that not all articles are *fully* extractive, cosine similarity is used to identify the closest match to each summary sentence. A cosine similarity matrix is calculated for each summary sentence where each entry represents this measure between each article sentence with that summary sentence.SK-Learn's inbuilt cosine similarity function is used.

Step 11: Label the article sentence with the highest similarity to each summary sentence as a positive event

The highest cosine similarity of the matrix for each summary sentence is then identified and marked as a positive event (*1*) while the rest are assigned zeros.

5. Data Exploration

5.1 Article and Summary Length

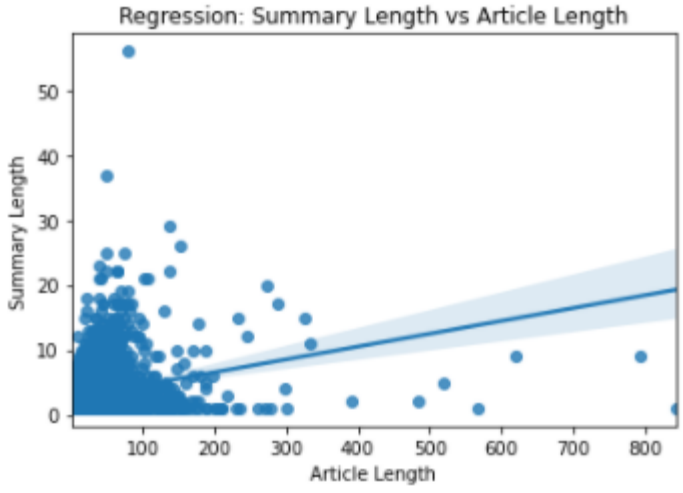
We start by looking at the descriptive statistics of the article and summary lengths as measured in number of sentences. The average and median article length is 33 and 26 while those for the summaries are 3 and 2 respectively. The average length is in line with the LEDE3 strategy already discussed. We also note the 75% of summaries have only 4 or less summary sentences and that the articles contain heavy outliers with the 95th 100th percentile ranging from 151 to 843.

Number of Sentences Percentile	Article	Summary
0.00	1.00	1.0
0.25	15.00	1.0
0.50	26.00	2.0
0.75	42.00	4.0
0.95	76.05	9.0
0.99	151.01	14.0
1.00	843.00	56.0

	mean
Article	33.270
Summary	3.248

It would also be interesting to examine the relationship between article and summary length. Instead of selecting the number of summary sentences as a user input, the article sentence length could then be used by the model to predict how many sentences to output.

```
r_squared =  0.056
```



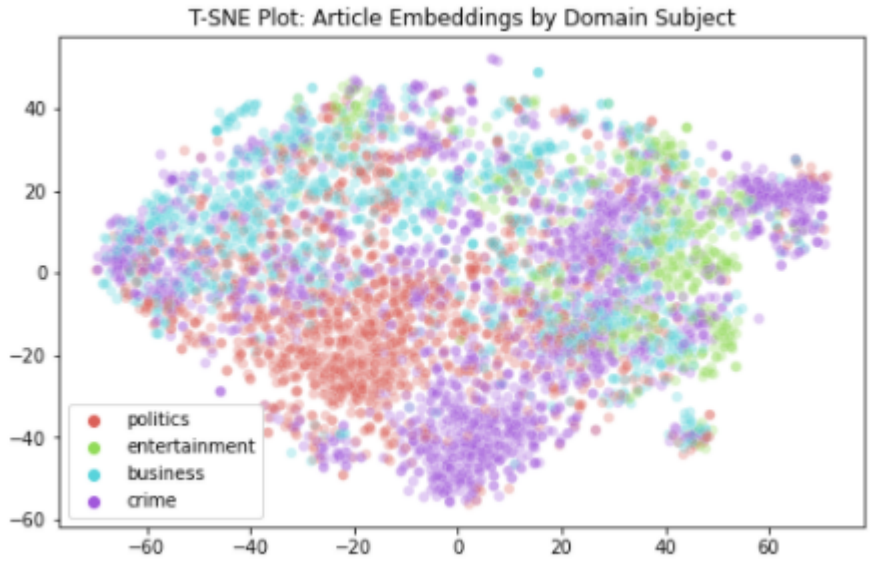
With a meagre r-squared of 5% and the heavy dependence on a handful of outliers, we see that there is little stable relationship between article and sentence length.

5.2 Embedding Visualization by Domain Subjects

We visualize the article embeddings (mean of sentence embeddings) using T-SNE and coloring by domain subject labels.

The domain subject labels were obtained as follows:

- Calculate BERT embeddings of “entertainment”, “crime”, “business” and “politics” as the four domain reference embeddings
- Calculate the cosine similarity of each article to the four references
- Label the article with its closest reference (highest cosine similarity)



Given the vast information loss of the representation (from many sentence embeddings to one article embedding to 2 dimensions) and the simple label assignment, we still clearly see regions of the embedding space dominated by different domains: bottom left clusters with politics, bottom right with crime, top right with entertainment and top left with business. The wide dispersion is also not surprising given that these categories are far from separable and some articles will mix topics like fraud (business crime), movie revenues (entertainment business), etc. On this note, we can also spy a binary split in the data along the top left to bottom right diagonal. This puts politics and crime in the bottom half and business and entertainment in the top half. This could be interpreted in a number of ways (!) but the generous version would be that politics and crime both have a common focus on social and legal matters while entertainment and business are united by their commercial orientation.

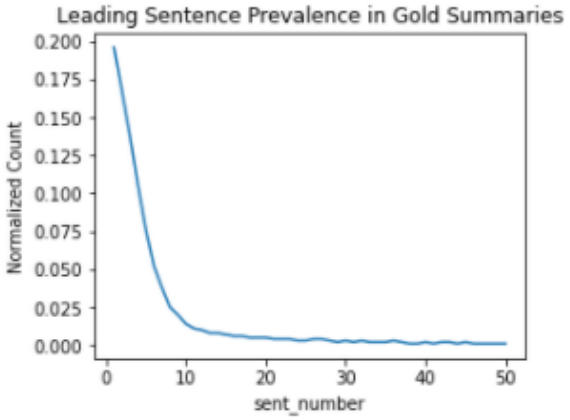
We might also suspect that apparent dispersion is influenced by imbalance in the sample. Low and behold, we see that crime represents nearly 38% of the sample and this could also explain why its dispersion appears more prominent in the plot.

	Normalized Count
crime	0.3760
politics	0.2648
business	0.2568
entertainment	0.1024

5.3 Leading Sentence Distribution

In order to quantify the presence of the leading sentence phenomena in our particular corpus, we look at the distribution of sentence index numbers found in the gold summaries. In aggregate, over 50% of summary sentences can be found in the first three sentences. We also note how the % prevalence of each sentence in the summary tails off quickly to zero with increase in distance from the start.

sent_number	Normalized Count	Cumulative
1	0.196	0.196
2	0.167	0.363
3	0.138	0.501
4	0.105	0.606
5	0.076	0.682
6	0.052	0.734
7	0.038	0.772
8	0.025	0.797
9	0.020	0.817
10	0.014	0.831



Finally, we look at the percentage breakdown of publications across the corpus and track the leading sentence distribution across the top 10 news vendors. The top 10 account for 80% of the corpus and the NY Times accounts for nearly 25%.

	% Breakdown	Cumulative
nytimes	0.2376	0.2376
bostonglobe	0.1334	0.3710
nydailynews	0.0870	0.4580
cnbc	0.0826	0.5406
9news	0.0514	0.5920
p://fortune	0.0472	0.6392
sfgate	0.0450	0.6842
p://nypost	0.0414	0.7256
tmz	0.0402	0.7658
aol	0.0370	0.8028

While there is some variance across leading sentence propensity across the publications, we note that the top3 have slightly lower leading sentence bias compared to the rest which makes us more comfortable that stratified sampling will not be necessary and that these publications do not introduce a non-representative bias to LEDE3 among news outlets.

Normalized Count	nytimes	bostonglobe	nydailynews	cnbc	9news	p://fortune	sfgate	p://nypost	tmz	aol
1	0.191	0.214	0.166	0.264	0.524	0.256	0.031	0.433	0.535	0.262
2	0.170	0.181	0.165	0.177	0.302	0.249	0.029	0.273	0.206	0.230
3	0.169	0.137	0.128	0.119	0.102	0.198	0.027	0.090	0.111	0.142
4	0.143	0.105	0.125	0.088	0.033	0.090	0.025	0.032	0.045	0.065
5	0.103	0.089	0.092	0.065	0.011	0.028	0.025	0.015	0.025	0.049
6	0.066	0.065	0.074	0.055	0.011	0.016	0.025	0.015	0.025	0.032
7	0.042	0.051	0.048	0.044	0.007	0.015	0.025	0.013	0.019	0.026
8	0.028	0.026	0.027	0.032	0.004	0.015	0.024	0.009	0.017	0.016
9	0.019	0.025	0.021	0.030	0.004	0.010	0.024	0.009	0.006	0.016
10	0.012	0.014	0.012	0.024	0.004	0.010	0.024	0.009	0.006	0.013

6. Machine Learning Implementation and Results

We take two separate approaches to modelling the summarization task. The first focuses on learning the internal embedding structure of each article and relates closest to our intuition that a good summarizer is one that is able to parse meaning and classify the salient points of a document. The supervised models considered here limit the features set to the sentence embeddings and their derivatives. The results are then compared to an unsupervised baseline using TextRank. The other less generalizable approach takes advantage of the known ordering structure particular to news articles and exploited in LEDE3 already discussed. We add sequential information to the embeddings and compare the results to LEDE3 as the baseline.

6.1. Metrics

The standard evaluation metrics for summarization tasks is the Rouge-N family of metrics that measure the n-gram overlap between the predicted and gold summaries. Rouge recall is defined as the number of n-gram overlaps between the summaries normalized by the number of n-grams in the gold summary. One potential drawback of this measure is that very long predicted summaries could contain all the words in the gold summary (earning a perfect 100% recall) but also have many words that are not. Rouge precision captures how concise the predicted summary is by changing the denominator to the number of words in the *predicted* summary. Rouge F1 is defined as the harmonic mean between Rouge recall and precision and, striking a balance between recall and precision, is the primary measure typically reported for summarization tasks.

We will use Rouge-1 as our primary metric throughout this report and will also track Rouge-L. Rouge-L considers the longest matching sequence of words using the longest common sequence (LCS). This stricter criteria, factoring in order, means that Rouge-L is always less than or equal to Rouge-1.

Rouge is not standard to the SK Learn library and we will use the *rouge-score* package to implement metric calculations.

6.2 Summary Length Selection

Unsupervised summarization tools usually require the user to input the desired number of sentences for the extractive summary. In line with the average length of summaries and LEDE3, we force all models to return the top 3 sentences by prediction probability and order them as they appear in the full article.

6.3 Model Caveats

The choice of the Rouge metric and the hardcoded selection of the top 3 sentences by predicted probability has a number of implications that are worth keeping in mind as we work through the models

- **Ranking vs threshold**: choosing the top 3 sentences by predicted probability ignores whether the predicted probability crosses the classification threshold or not and has consequences when choosing between models.
    - Model fine tuning such as incorporating class weights or hyperparameter tuning are likely to change thresholds but not sentence rankings and are expected to be of little utility.
    - Rouge scores and the standard confusion matrix may not be consistent between models. It is quite possible that a model with an inferior confusion matrix can produce the same or better Rouge scores based purely on sentences that are included in the summary despite not meeting the classification matrix threshold. An example might be that a simpler model produces better Rouge scores than a sophisticated one despite producing an inferior confusion matrix based on standard classification metrics

- **Overlap vs binary:** although more nuanced, another source of relative disagreement between Rouge scores and the confusion matrix can occur when one model's false positives have better n-gram overlap with the gold summaries than the correctly classified sentence. Given that we are using sentence embeddings (not word frequency measures) and max cosine similarity (not always 1), this may not be as unusual as it sounds.

6.4 Supervised Learning Using Only Embedding Information

In this section, we measure the ability of supervised learning to uncover the internal structure of the embedding space. The goal is to classify each sentence as either belonging to the summary or not. The sentences are the samples and the features set is the BERT sentence embedding and the article mean embedding (mean of sentence embeddings).

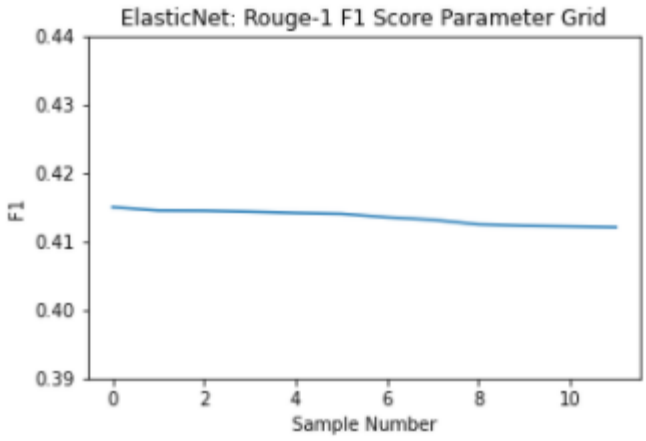
6.4.1 Logistic Regression

Using an 80% / 20% train test split, we obtain the following results:

ROUGE-1 LogReg	f1	recall	precision
Default	0.411	0.396	0.561
Balanced	0.407	0.396	0.551
ElasticNet	0.415	0.401	0.566

As expected, there is little variation across the default setting and class weights set to balanced. Hyperparamter finetuning using elastic net also yielded very little benefit but is still the optimal model and will be taken forward for comparison to the baseline. This occurred with the L1 ratio set to 0.25 and the inverse of regularization at 0.5.

Examining some of the caveats further, we see a very low range of 41-42% in Rouge F1 scores across the gridsearch (custom implementation as Rouge is not an evaluation option in SKLearn).



The vast difference in the confusion matrix between the default and balanced cases is also evident: despite a better Rouge score, the default model barely predicted any positive events at the 50% threshold level.

Confusion Matrix by Model

Default

Predicted / Actual	Out	In
Out	29253	65
In	3032	75

Balanced

Predicted / Actual	Out	In
Out	19352	9966
In	1156	1951

ElasticNet

Predicted / Actual	Out	In
Out	29234	84
In	3013	94

6.4.2 Neural Networks

Using the same train test split, we train various neural net architectures using 50 epochs and a Keras implementation. A selection of attempted one and two layered dense networks is provided below where *No Bal* indicates no adjustment for imbalance data and *Bal* indicates where class weights were set as balanced:

ROUGE-1 Neural Network (Dense Dense)	f1	recall	precision
NN 25 No Bal	0.400	0.391	0.548
NN 25 Bal	0.405	0.397	0.547
NN 50 Bal	0.398	0.392	0.540
NN 75 Bal	0.403	0.393	0.549
NN 25 25 Bal	0.409	0.389	0.571
NN 25 50 Bal	0.402	0.405	0.538
NN 50 50 Bal	0.400	0.392	0.543

Just as in the case of logistic regression, there was little variation in Rouge scores across model architectures. The best score was delivered by *NN 25 25 Bal*- a two layered dense dense network with 25 neurons per layer and class weights set equal to balanced.

Somewhat surprisingly, the neural network provides no improvement on the logistic regression Rouge score (40.9% vs 41.5%). One reason for the underperformance could be that we have not provided the neural network with enough contextual embedding information at the article level for its nonlinearity to take advantage of. After all, each sentence only has the article mean to map itself against. Addressing this with the use of clustering is discussed under future work.

A comparison of the optimal standard confusion matrix for each model algorithm (NN 25 25 Bal vs Log Reg Balanced) shows that the neural network offers an almost identical macro score breakdown but with substantially better accuracy.

Standard Confusion Matrix Scores (Macro)	f1	recall	precision	accuracy
NN 25 25 Bal	0.601	0.638	0.568	0.769
LogReg Bal	0.595	0.644	0.554	0.657

6.4.3 Comparison to TextRank Baseline

We implement TextRank on the same test data and compare to the optimal logistic regression (*ElasticNet*) and neural net (*NN 25 25 Bal*) models:

ROUGE-1	f1	recall	precision
TextRank	0.340	0.326	0.469
ElasticNet	0.415	0.401	0.566
NN 25 25 Bal	0.409	0.389	0.571

ROUGE-L	f1	recall	precision
TextRank	0.271	0.255	0.384
ElasticNet	0.371	0.356	0.510
NN 25 25 Bal	0.360	0.341	0.506

The supervised models substantially outperform TextRank across all measures of Rouge in the test set. The f1 improves by over 7% and both recall and precision are higher. This is important as it rules out the scenario where the supervised model was scoring higher on recall by returning longer sentences.

We also provide the following example summary from the test set:

Original
By CORKY SIEMASZKO DAILY NEWS STAFF WRITER Mother Teresa believed she was posse
ssed by the Devil, the archbishop of Calcutta said yesterday. So the revered nu
n, whom the Vatican hopes to make a saint, underwent an exorcism and afterward
"slept like a baby,"he said. Archbishop Henry D'Souza's bizarre revelation came
as millions yesterday marked the fourth anniversary of Mother Teresa's death. B
ut D'Souza told CNN and The Associated Press in India he truly believed

Elastic Net
Archbishop Henry D'Souza's bizarre revelation came as millions yesterday marked
the fourth anniversary of Mother Teresa's death. The Catholic cleric said he di
agnosed the demon in Mother Teresa shortly before she had a fatal heart attack
Sept. 5, 1997, and died at age 87. Mother Teresa won a Nobel Prize for her lif
e's work, and Pope John Paul has begun the process of declaring her a saint.

Neural Net
Mother Teresa believed she was possessed by the Devil, the archbishop of Calcut
ta said yesterday. So the revered nun, whom the Vatican hopes to make a saint,
underwent an exorcism and afterward "slept like a baby," he said. Mother Teresa
won a Nobel Prize for her life's work, and Pope John Paul has begun the process
of declaring her a saint.

TextRank
Mother Teresa believed she was possessed by the Devil, the archbishop of Calcut
ta said yesterday. So the revered nun, whom the Vatican hopes to make a saint,
underwent an exorcism and afterward "slept like a baby," he said. D'Souza said
he suggested an exorcism, and the elderly nun, who had devoted her life to help
ing the poor and downtrodden, quickly agreed.

An interesting observation from the above is that our text cleaning process does a good job in removing the irrelevant publication specific information (first few uppercase words in *Original*) that appears in many of the article summaries. While cleaning the gold summaries for this information would not result in changing the ranking of the implemented models, the scores are actually better (across the board) than indicated here.

6.5 Supervised Learning Including Sequential Information

We now add sequential information to the features set and compare the results of various models to LEDE3. The sentence number is added to the previous features set for logistic regression where the samples remain the individual sentences. Long short term memory is chosen as the neural net architecture in this case where the samples are now the full article fed into the model as a sequence. The features set is identical to that in the previous section and only contains embedding information.

6.5.1 Logistic Regression

In this case, the default logistic regression model performs materially better than the balanced case across all Rouge-1 metrics. We also observe that the addition of the sentence order saw large improvements as measured against section 1. In particular, Rouge f1 has already increased by over 10%.

ROUGE-1 LogReg	f1	recall	precision
Default	0.525	0.527	0.684
Balanced	0.499	0.505	0.647

6.5.2 Long Short Term Memory (LSTM)

In order to deal with the different article (sequence) lengths, we decided against padding and used a batch size of one to feed the sequences into the network. Full Keras implementation code is available on Github. A variety of unidirectional and bidirectional networks were tested with epochs of 50 and a selection of these are provided below:

ROUGE-1 LSTM	f1	recall	precision
LSTM Uni 25	0.599	0.593	0.760
LSTM Uni 50	0.597	0.586	0.767
LSTM Bi 25	0.595	0.583	0.767
LSTM Bi 50	0.603	0.594	0.769

There was very low variance between LSTM model scores but the best score was provided by a bidirectional network with 50 neurons. In this case, the neural network architecture provided a marked improvement on the logistic regression case and improved the f1 of section 1 by almost 20%. The improvement was also consistent across both recall and precision.

6.5.3 Comparison to LEDE3 Baseline

LEDE3 (return first 3 sentences in all cases) was implemented across the same test sample and compared to the best logistic regression and LSTM models:

ROUGE-1	f1	recall	precision
LEDE3	0.567	0.599	0.698
LogReg Def	0.525	0.527	0.684
LSTM Bi 50	0.603	0.594	0.769

ROUGE-L	f1	recall	precision
LEDE3	0.552	0.582	0.678
LogReg Def	0.498	0.500	0.648
LSTM Bi 50	0.587	0.577	0.746

Given the high bar set by LEDE3, LSTM's ability to beat it on both Rouge-1 and Rouge-L f1 is impressive. Digging a little deeper, we see that LSTM's recall is marginally below LEDE3 but its precision is far higher. This suggests that in the few cases where it does diverge from returning the top 3 sentences, it does a good job in picking out relatively shorter or more concise sentences. In this respect, it should be noted that all models (including LEDE3) are cleaned of the irrelevant publication specific information discussed earlier and the better precision of LSTM is not an artificial consequence of this choice in model pre-processing.

6.6 Future Work

The following areas are identified for further development:

- **Supervised TextRank:** An interesting riff on the logic of TextRank could be using the rows of the cosine similarity matrix as the features set for the sentences and running this through the supervised models to uncover a competing ranking algorithm to TextRank.
- **Additional Features Engineering**: The comparative performance between the neural net and logistic regression models were slightly disappointing where we expected the non-linearity of nets to be able to uncover deeper relationships in the internal structure of the articles. One reason this was not the case is that the only article level contextual information we gave each sentence was the article mean. One way to provide more information without computation overload would be using a simple clustering algorithm to calculate the centroid of 3-5 clusters and use this together with an assignment cluster label for the sentence to the features set.
- **More Training:** Another potential reason for the relative underperformance of neural nets could be that more training was required. The underlying database is generous in size and this could certainly be accomplished with access to adequate computational power.
- **Attention:** No examination of a challenging NLP task is complete without adding the attention mechanism to the neural net architecture. As a first step, this could be added to the LTSM model or a new GRU. The next step would be to aim the power of transformers on the task.
- **Mixing Corpuses:** One challenge of the chosen news corpus is that its overwhelming sequential structure limits the generalizability of the model.

This could be tackled by mixing corpuses from other domains with a less ordinal structure.