Feature-based Extraction Summarizer

Edgar Samudio, Justin Chiu, Marlene Shankar

1. Task Definition

The purpose of this project is to build sentence-based extractive summarizer. Given a text, the summarizer will score each sentence and extract the higher scoring sentence. For example, (from CNN datasets):

> “Can't see Harry Potter's Dumbledore as gay? Harry Potter author J.K. Rowling has a terse message for you J.K. Rowling revealed that the Hogwarts School headmaster was gay after" Harry Potter and the Deathly Hallows , " the final book in the boy wizard series, was released in 2007 Fan Ana Kocovic started the exchange by asking the author , "I wonder why you said that Dumbledore is gay because I can't see him in that way " " maybe because gay people just look like ... people? " J.K. Rowling wrote in her march 24 reply J.K. Rowling 's fans loved it.”

should extract the sentence:

> “Fan Ana Kocovic started the exchange by asking the author , " i wonder why you said that Dumbledore is gay because i can't see him in that way”

2. Motivation

We based our project on a reddit bot known as ‘autotldr’, which takes a text and outputs the top five words found in the text. This bot uses a simple bag-of-words implementation to get these top unigram words. We wanted to make a similar application where the user could input text and receive the most important sentences, summarizing the given text.

3. Accomplishments

- Implemented a feature-based extractive summarizer using Dragomir R. Radev's LexRank text summarization algorithm.
- Evaluated idf values of words in training data.
- Calculated accuracy on our feature-based extractor based on the CNN dataset.
- Learned how to make cosine similarity matrix stochastic, aperiodic and irreducible to interpret it as a Markov Chain.

4. Description

Our application extracts the most important sentences from a given text. Using the LexRank algorithm, we can score every sentence by how similar the sentence is compared to other sentences in the given text. The main purpose of our extractor is to find sentences in the given input and summarize the entire input with the most important sentences by calculating each sentence's probability of similarity. By using a dataset to train, test, and validate our model, we can conclude with accurate results.

Below are two main algorithms used in LexRank:

```
input  : A stochastic, irreducible and aperiodic matrix $\mathbf{M}$
input  : matrix size $N$, error tolerance $\epsilon$
output: eigenvector $\mathbf{p}$
$\mathbf{p}_0 = \frac{1}{N}\mathbf{1}$;
t=0;
repeat
    t=t+1;
    $\mathbf{p}_t = \mathbf{M}^{\mathrm{T}}\mathbf{p}_{t-1}$;
    $\delta = ||\mathbf{p}_t - \mathbf{p}_{t-1}||$;
until $\delta < \epsilon$;
return $\mathbf{p}_t$;
```

Algorithm 2: Power Method for computing the stationary distribution of a Markov chain.

```
MInputAn array $S$ of $n$ sentences, cosine threshold $t$  output: An array $L$ of LexRank scores
Array $CosineMatrix[n][n]$;
Array $Degree[n]$;
Array $L[n]$;
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $n$ do
        $CosineMatrix[i][j]$ = idf-modified-cosine($S[i]$,$S[j]$);
        if $CosineMatrix[i][j] > t$ then
            $CosineMatrix[i][j] = 1$;
            $Degree[i]++$;
        end
        else
            $CosineMatrix[i][j] = 0$;
        end
    end
end
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $n$ do
        $CosineMatrix[i][j] = CosineMatrix[i][j]/Degree[i]$;
    end
end
$L$ = PowerMethod($CosineMatrix$,$n$,$\epsilon$);
return $L$;
```

Algorithm 3: Computing LexRank scores.

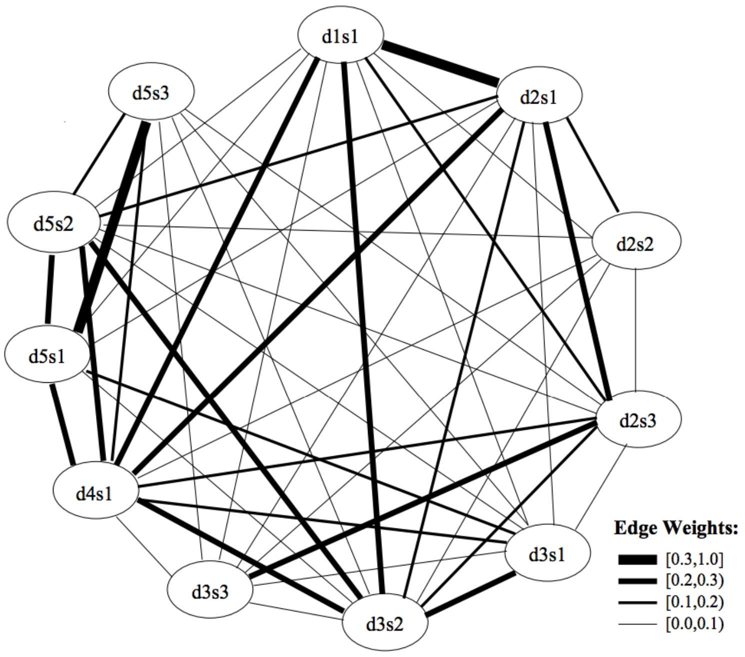

Figure 2: Weighted cosine similarity graph for the cluster in Figure 1.

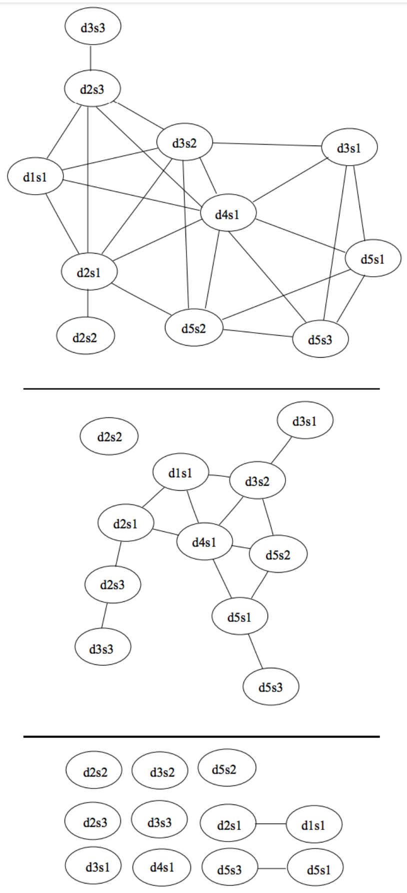

Figure 3: Similarity graphs that correspond to thresholds 0.1, 0.2, and 0.3, respectively, for the cluster in Figure 1.

5. Evaluation

We used the CNN/Daily Mail dataset for training and testing. The link to download this dataset could be found in the ‘readme’ section of our github repository. In the CNN portion of the dataset, there are approximately 90,000 documents, much more than what we needed to complete this project, so we focused on only the CNN portion of the dataset. In our evaluation, we will calculate the accuracy.

6. Results

After training our model, we ran it through over 1000 documents to test and validate our model and compared the results to the results that the CNN dataset provided. The following evaluation statistics are listed below:

Threshold being tested: 0.1

Accuracy: 0.1791

The way we labeled each sentence is by assigning a value of ‘0’ (very unlikely to summarize the text), ‘1’ (somewhat likely to summarize the text), ‘2’ (very likely to summarize the text). We calculated the accuracy of our extractor by taking the number of correctly labeled sentences and divided that by the total number of sentences with that label. We ran this evaluator through the CNN dataset, which summarized more than a 1000 documents in about 18 minutes. Our results from the evaluator prints out the accuracy, precision, recall, F1, for each type of sentence, ‘0’, ‘1’, ‘2’. For this study, we will only focus on the results from sentences labeled ‘2’.

7. Analysis

Looking at the results above, we can see that we obtained an accuracy of approximately 17.91%. Our accuracies are slightly lower because when calculating evaluation statistics, the length of our summary could vary in length, skewing our results negatively. In Dragomir Radev's paper, he calculates his results considering the score and the degree of threshold. The lower the threshold, the more informative and less misleading the similarity graphs are. This is a different approach to how we calculated the effectiveness of our extractor. Dragomir Radev concluded with an accuracy around 35 to 36% with a threshold of 0.1, while we received an accuracy of 17.91% with the same threshold. There are a couple reasons why Radev received higher accuracy. In his study, he summarized multiple documents at once while we summarized each document separately. Overall, summarizing multiple documents at once would lead to better results. Additionally, we used different datasets to train, test, and validate our extractors. Radev used all three DUC 2004 data sets, while we used the CNN dataset. The difference in datasets could negatively affect our results.

8. Conclusions

Some key things we learned from this project is how a computer can process text and extract only the sentences that summarize the given input text. In the LexRank algorithm, Radev scores each sentence by calculating how similar it is compared to all other sentences in the given text. Similarity between sentences is shown above in the weighted cosine similarity graph labeled 'Figure 2'. Similarity graphs shown in 'Figure 3' show how a higher threshold correlates to a less informative and a more misleading similarity graph. Therefore, when we run the algorithm with a threshold of 0.01, we received a 36% accuracy, which is twice the original accuracy that was evaluated with a threshold of 0.1. In the future, we would focus on summarizing multiple documents at once and properly selecting length cutoff for each summary to improve the accuracy on our extractor.