News Article Summarization

Shashwat Sanghavi, Shraddha Phadnis
sanghavi.s@husky.neu.edu, phadnis.s@husky.neu.edu
CS 6120 Natural Language Processing, Spring 2018
College of Computer and Information Science
Northeastern University

1. Problem Description

Summarization is widely researched field in the domain of Natural Language Processing. In this project we have implemented and improvised some of the widely known text summarization algorithms to summarize web articles.

The automatic text summarization in NLP can be broadly classified into two classes i) Extractive summarization ii) Abstractive summarization. Extractive summarization uses the given text and extracts the important sentences based on the sentence scores, calculated using techniques like word frequency counts, Tf-Idf, word vectorization etc. Here, generated summary contains the important sentences from the original document as it is and there is no rephrasing involved. Abstractive summarization on the other hand once identifying the important sentences, restructures the sentences to get the meaningful summary more in a humanly manner.

In this project, based on the nature of our dataset we will be focusing on Extractive summarization technique. We will be discussing various techniques to improve text summarization algorithms like Luhn's method and TextRank algorithm. We will also discuss how the methods that we have used significantly improved the accuracy compared to existing Luhn's method and TextRank algorithm in the evaluation and result section.

Our algorithm expects input in the form of a single page article web page URL. An article can contain pictures, figures, diagrams which we will preprocess to extract only essential information from the article. The output is the shortened summary of the article which includes maximum of 5-6 sentences summarizing the whole article.

For an example, for the web article on this link, the summary would be as below.

"The Daman and Diu administration on Wednesday withdrew a circular that asked women staff to tie rakhis on male colleagues after the order triggered a backlash from employees and was ripped apart on social media. The union territorys administration was forced to retreat within 24 hours of issuing the circular that made it compulsory for its staff to celebrate Rakshabandhan at workplace. It has been decided to celebrate the festival of Rakshabandhan on August 7."

2. Related work

We chose to perform extractive text summarization using unsupervised approach. We considered Luhn's approach proposed by Prof. Hans Peter Luhn and TextRank algorithm as our baseline models.

2.1 Luhn's method

Luhn's approach is used to obtain the sentence scores using word frequency count. In Luhn's method, first we find the important words. Important words are those which have higher frequencies. Then important sentences are found by finding the number of important words in each sentence. This set of important sentences is then given as a summary for the input text.

2.2 TextRank

Graph-based ranking method such as PageRank is widely known for finding important pages on the worldwide web. TextRank algorithm is derived from PageRank.

PageRank algorithm works on the assumption that important page is the page which is connected with most number of other pages.

Fig. 1: Overview of the summarization

TextRank algorithm works on the intuition that, important sentence in the text is the one which is similar to the most other sentences.

TextRank can be implemented by using following pseudo code which uses PageRank algorithm.

1. Build a graph with each sentence as a node
2. Create links between all the sentences with weights as the sentence similarity scores
3. Run PageRank on this weighted graph to obtain sentence ranking
4. N top ranked sentences will be included in the summary

Here, in step 2, the similarity of sentences is measured using n-grams as a feature. Thus, the similarity between two sentences will depend only on the number of common n-grams.

In this work, we have tried to increase the efficiency of Luhn's approach and TextRank algorithm by changing features (discussed further) used for them.

3. Methodology

Fig. 1 shows the overall process of text summarization. Here the URL of the web article is provided as an input. In Order to obtain the summary, we first need to get the html content of the page and then summarization techniques can be applied on the extracted article text.

3.1 Data Preprocessing

3.1.1 Data Extraction

To extract the news text from the HTML web page, we use Python libraries (i) urllib and (ii) BeautifulSoup. The library 'urllib' dumps the html file into a python variable. BeautifulSoup is used to parse the structured (i.e. html, xml, xhtml) documents.

In our dataset we can get the contents of the news articles by following details.

(i) The Guardian: The content of the news article is located in the with a parameter "itemprop" = "articleBody".

(ii) Hindustan Times: The content of the news article is located in the with a parameter "itemprop" = "articlebody".

(iii) India Today: The content of the news article is located in the with a parameter "class" = "story-details".

BeautifulSoup can find these specific tags and can take out the text from them.

3.1.2 Data cleaning / preprocessing

The obtained text needs further processing and cleaning. The text contains multiple new line characters, punctuations, non ascii characters and unspecific sentence ending. In majority of cases sentences ends with '. ' (dot followed by space),

however there are few occurrences where the sentence ends with '.' (dot followed by no space). Following are the examples.

- Standard case : My name is John Doe. I am working on NLP project.
- Non-Standard case: My name is John Doe. I am working on NLP project.

This text needs to be tokenized and each sentence needs to be separated for the extraction. The simplest approach is to use sentence tokenizer (part of NLTK library) to separate sentences where each sentence is separated by either newline character or a combination of some punctuation and a space. However, this cannot work for the previously mentioned non-standard case.

The other approach to perform tokenizing is by using unsupervised sentence boundary detection proposed by Kiss and Stuck [1]. The same can be performed in Python using NLTK library and punkt_tokenize. This approach gave better results for following situations.

- **Input text:** I am Mr. John Doe. I met Prof. Smith to take his advice.
- **Simple tokenizer output**: ['I am Mr.', 'John Doe.', 'I met Prof.', 'Smith to take his advice.']
- **Punkt tokenizer output**: ['I am Mr. John Doe.', 'I met Prof.Smith to take his advice.'

Before performing the tokenizing task, the punkt tokenizer learns that Mr., Prof. are abbreviations and it doesn't need to separate the sentences from these words. However, punkt also fails in the above mentioned non-standard case as it detects'. '(dot followed by space) and
then decides whether to break sentence from that or not. In non-standard case, '.' cannot be detected as a separator as it is not followed by a space.

Alternative approach: Can we take '. '(dot followed by space) instead of '.' while reading the text from html using BeautifulSoup? Yes, in the html document, each sentence is kept in an individual tag … .

In order to parse through these tags, we can go to the location of article text and then traverse that specific div to obtain each of the tags. This list of tags can be traversed to obtain independent sentences. Simple tokenizer following this method gave desired cleaned data. Examples as follows:

- **Html text**:   My name is John Doe. I am a student. I am studying NLP.
- **Output - simple parser**: My name is John Doe. I am a student.I am studying NLP
- **Output–alternative parser**: ['My name is John Doe. I am a student.', 'I am studying NLP.']

This alternative approach also helps in removing unwanted tags such related to the span, hyperlink, images and figures.

This cleaned data is stored in a csv file with its gold summary. The csv file can be easily processed using Python and pandas.

3.2 Features

We have used different features in the two proposed algorithms. The improvisation of Luhn's method uses TF-IDF, stemming and N-grams. The method that is based of TextRank uses word frequency, N-grams, word context, Word2Vec and stemming.

3.3 Methods

The following are the improvisation over the baseline models.

3.3.1 Luhn with TF-IDF and stemming:

In Luhn's method, the frequency of word occurrence (the word count) has been used as a primary feature. We defined most frequently used words as important words in the text. Then we find sentence score for each sentence using these important words. Sentence score can be calculated by calculating probability of important words in the sentence. With the obtained score for each sentence we will extract sentences with highest score as the text summary.

One obvious problem that we faced with this approach was, if there are large sentences, it will cover many important words and even though the sentence is not very important from the summary point of view, this sentence will be in the final summary result as it gets the higher score. To improvise on this, we tried using

Term frequency and Inverse Document frequency approach.

Luhn's method was implemented by removing the stop words which may occur frequently in the article and are not very important followed by finding the word count in the article. This word count based method doesn't consider the low frequency words as important words. However, it is possible that some of the low frequency words are important. Hence this word count based approach is not very useful to find important words.

Another approach is to find words is inverse document frequency (IDF) which decreases the weight for the commonly used words and increases the weight of the words that are not used very much in a collection of documents. This is combined with term frequency to calculate tf-idf. Thus, words like stop words which are very frequently used in each document will get lower weight whereas low frequency words link nouns and adjectives will get higher weights.

Though this approach is efficient to find important words, it does not take care of similar words in the text. For an example, the words apology and apologize are considered as different words. However, it is very rare that both of these words are referring to different meanings in the same context. To address this issue, we considered only root of the word (i.e. apology for both apology and apologize) for scoring purposes.

3.3.2 TextRank with Word2Vec

Performance of the TextRank algorithm depends on the accuracy of similarity index. In Order to improve the TextRank algorithm, we tried to improve the similarity measurement method.

As discussed in earlier section, the TextRank algorithm uses N-grams as a feature to find similarity between two sentences. While we consider this as a feature, we don't consider the context of the word and thus meaning of that word is not preserved.

In 2014, an algorithm to find Word to Vector was found in which one considers the neighboring words in the window of size N. Converting word to vector in such fashion will also retain the meaning of the word. This algorithm is known as Word2Vec.

Word2Vec takes all the words in the vocabulary as an input. It outputs the probabilities of words to be within $\pm N$ window size from the input word. Word2Vec uses semi supervised machine learning to find these probabilities.

In the neural network shown in fig. 2, input and output layers contains all the words in the vocabulary. After training this network, each row of a weight matrix (of size$|V| \times X$) from input layer to hidden layer denotes the vector for each word in the vocabulary. This vector can be used to denote a word in the space.

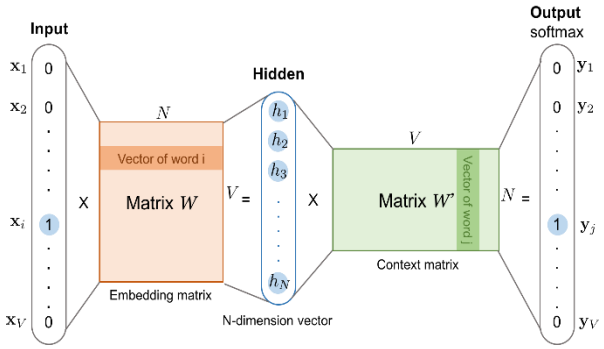

Fig 2. Neural network model for Word2Vec

In below example four vectors, King, man, queen and woman are shown. We can also perform various algebraic operations on these vectors. These operations will give appropriate output vector. For an example, difference between V_{king} and $_{queen}$ is same as difference between V_{man} and V_{woman}.

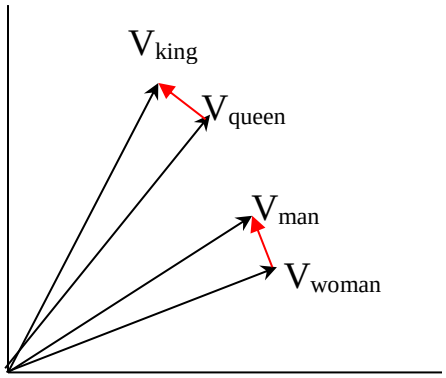

Fig 3. Example of Word2Vec vectors

Once the these word vectors are obtained using word2Vec algorithm, we add all the word-vectors of the words in a sentence to obtain a sentence vector. Now, similarity between two sentences can be found using cosine distance between two sentence vectors.

4. Experiments

4.1 Dataset

For the specific purpose of our project, we required a dataset containing news articles and human written summaries. We are using kaggle dataset with 4500+ news articles from various sources. This dataset contains news articles from 3 english newspapers -- (i) The guardian (~300 articles) (ii) IndiaToday (~3000 articles) and (iii) The Hindustan times (~1200 articles). The dataset contains 4500 rows with 5 columns which includes web-url of the article, author name, publication date, headline and gold summary of this article.

2. Evaluation parameter

The quality of the summary can be measured using extrinsic evaluation as well as intrinsic evaluation. We have done intrinsic evaluation for ~50 articles by reading them by ourselves. For the extrinsic evaluation we have used ROUGE-N score. The quality of the summary can be measured using ROUGE-N score. Here N denotes the N in N-gram. We find ROUGE score by finding the following ratio.

$$ROUGE - N = \frac{X}{Y}$$

X: number of common N-grams in gold summary and machine summary

Y: number of N-grams in gold summary

ROUGE-1 and ROUGE-2 score are widely used scores for evaluating text summaries.

3. Baseline Models

As mentioned in the Related work section of this report we have considered Luhn's text summarization and TextRank algorithm as our baseline models. By using techniques like Word2Vec, TF-IDF, stemming, we have improvised the Rouge scores of the summary. The following section shows the output comparison of different methods that we have implemented in this project.

4. Results

The below table shows the comparison of ROUGE-1 and ROUGE-2 scores for different approaches.

Method	ROUGE-1	ROUGE-2
TextRank with WordCount	0.3648	0.1324
Luhn's method with TF-IDF and stemming	0.4019	0.1583
TextRank with Word2Vec	0.4217	0.2024

The Fig 4 shows the ROUGE-1 and ROUGE-2 score comparison for the different lengths of the document. The length here is measured in terms of number of words. The average number of words in files is ~550. The following graphs shows that TextRank with Word2Vec performs the best for these kinds of articles. From these figures and above table, we can clearly observe that TextRank with word2Vec outperforms the other two approaches.

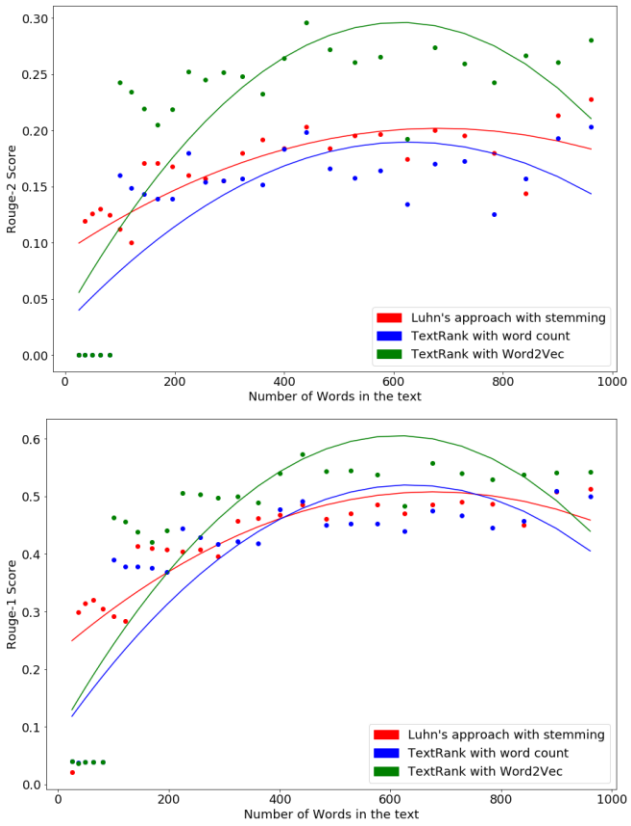

Fig 4. Word Count Vs Rouge-N Score

The main reasons behind this improved performance are following:

1. Word2Vec considers the context of the word, while N-gram doesn't.
2. Constructing a Sentence vector from context aware word vectors preserves the meaning of the sentence. (i.e: similarity with other sentences will depend on the meaning rather than the vocabulary used)
3. TextRank algorithm works on the intuition that important sentence is the one which is similar to the most similar to other sentences.

5. References

[1] Kiss, Tibor, and Jan Strunk. "Unsupervised multilingual sentence boundary detection." Computational Linguistics 32, no. 4 (2006): 485-525.

[2] http://calvinfo.blogspot.com/2010/07/luhns-auto-abstract-algorithm.html

[3] Luhn, H. P. (1958). The automatic creation of literature abstracts.

[4] Lloret, Elena, and Manuel Palomar. "Text summarisation in progress: a literature review." Artificial Intelligence Review 37, no. 1 (2012):1-41.

[5] Bhatia, Neelima, and Arunima Jaiswal. "Automatic text summarization and it's methods-a review." In Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference, pp. 65-72. IEEE, 2016.

[6] https://www.nltk.org/py-modindex.html

[7] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[8] https://medium.com/@acrosson/summarize-documents-using-tf-idf-bdee8f60b