Automatic Text Summarizer - Extractive

IT550 - Information Retrieval Project Report

Sana Baid
202011019
M.Tech ICT - ML
DA-IICT, Gandhinagar

Shivangi Gajjar
202011023
M.Tech ICT - ML
DA-IICT, Gandhinagar

Shradha Makhija
202011029
M.Tech ICT - ML
DA-IICT, Gandhinagar

Mantra Sanathra
202011041
M.Tech ICT - ML
DA-IICT, Gandhinagar

E.V.V. Hari Charan
202011056
M.Tech ICT - ML
DA-IICT, Gandhinagar

***Abstract*—Manual summarization loses practicality in the face of the massive archives of news articles, scientific papers, legal documents, etc on the internet. In this context developing Automatic Text Summarization (ATS) techniques becomes extremely significant. Despite the numerous methods and techniques available, summary generation without human intervention has yet to reach the preferable standards.**

***Index Terms*—summarization, extractive, vectorization, K-means clustering, ranking, rouge scores**

I. Introduction

There exists an enormous amount of textual data on the internet in the form of websites, user reviews, news, blogs, social media networks, etc. This textual data grows rapidly on a daily basis. Due to this, users tend to spend more time to find the information they require. It is impossible for the user to go through all the textual contents of their search results. This is where summarizing comes into the picture [5]. But manual summarization is time consuming as well as an expensive task. The solution to this problem lies within the AutomaticText Summarization (ATS) system [3].

Depending on the type of generated summary ATS can be classified into three types namely extractive, abstractive and hybrid. This report primarily focuses on the extractive summarization approach combined with K-means clustering technique for the generation of summary. The effect of two different types of vector models on summary generation has also been shown. The report is divided into the following sections, section 2 deals with the types of approaches available in ATS, section 3 dives deep into the methodology used for summary generation, section 4 demonstrates the experimental results generated and section 5 contains the conclusion.

II. Types of Automatic Text Summarizers

In this section, we explore the classification of an ATS system based on the type of output of the summarization system. As mentioned earlier, this approach can be extractive, abstractive or hybrid.

The main distinction between extractive and abstractive summary is in terms of the sentences of the summary. If the sentences in the summary are picked out as they appear in the original text, then it is called an extractive summary. If the sentences in the summary are different from the original text, then it is called an abstractive summary.

Abstractive summarization handles the input text as an intermediate representation and generates new phrases and sentences that impart the most useful information from the original text. Abstractive summarization overcomes the grammatical inconsistencies of the extractive summarization and it is closer to how humans summarize text [4].

In extractive summarization, sentences of the original text are ranked as per their importance and then, based on the size of summary required, top n sentences are picked out from this ranking. Extractive summarization produces more accurate results as compared to abstractive summarization because sentences are not altered [7], This paper focuses on an extractive based approach. .

After the approach is decided, it is important to explore methods of text selection and summary generation. Various methods such as statistical-based methods, graph-based methods, machine-learning-based methods and many more are used for this purpose.

In this project, we have implemented a clustering-based method which uses sentence centrality to choose the most important sentence out of that cluster. The scoring of sentences is performed by building centroids and then by choosing the central sentences via distance measure from centroids. The closer a sentence is to the centroid, the more important it is [2].

Clustering-based summarization takes into consideration the relevance of a sentence and also tackles the issue of redundancy.

III. Methodology

Breaking down the goals of the clustering-based summarization approach discussed above, three main tasks need to be performed. 1) To cluster the sentences of the input text 2) To rank and order clusters 3) [6] to select the representative sentences for the summary. This section focuses on achieving these goals.

A. Dataset

In order to implement and evaluate our Extractive Summarization technique, we have used a common benchmarking dataset called CNN/Daily Mail [1]. It contains about 312,084 html documents, each with text data and a multi-sentence summary. To extract the data from these files we have used the BeautifulSoup API.

B. Preprocessing

The data obtained after scraping is unstructured, and a structured representation is required for the algorithm to perform efficiently. Various linguistic techniques have been applied for preprocessing of the data. The steps for preprocessing include removing all the special characters and digits from the data and converting the entire corpus to lower case in order to avoid issues arising due to case sensitivity. Then, after tokenizing the corpus, stopwords have been removed. After removal of stopwords. The final and important preprocessing technique is the lemmatization. This technique helps in grouping together the inflected forms of the same word so they can be analysed as a single item making the querying process consistent.

C. Vectorization

Preprocessig of the data was done to obtain a structured form of data but in order to find relation between the sentences or to evaluate the importance of the sentence, we need to convert the data into numerical representations or better, vectors. This process is called vectorization. Out of all the vectorization techniques put there, this project has restricted the scope to two famous vectorzsation methods, TF-IDF and Word2Vec(Continuous Skip Gram).

1) TFIDF: Term frequency- Inverse Document Frequency is a statistical measure that tells us how important a word is in the document. It is a measure calculated by the product of term frequency, i.e the occurrence of a term in the document, and the inverse document frequency, i.e importance of the term has inverse proportionality with occurrence of that term across all the documents at hand. Since we are generating a summary of only one document, at a time, there is no significance of the inverse document frequency, its value for the words will always be one.
In this approach, we have calculated the tf-idf vectors for each sentence in a document. That is, for each document we have calculated the tf-idf scores for all the present words and then for each sentence, a vector is generated by appending these scores, based on the words present in them, in an ordered manner.

2) Word2Vec: It is a neural network based approach to find association between words in a corpus. There are two prominent architectures, CBOW (continuous bag of words) and Continuous skip gram. In our approach, we have used the continuous skip gram approach. In this architecture, during the training, the model takes up a word and finds the context words falling in the window. The context words nearby are weighed more than those that are farther. We have trained a model from the documents that give a 150 feature vector to each word present, based on the associations calculated.
In order to generate sentence vectors for each sentence in a document, we have taken the average of each of the feature vectors of the words present in the sentence.

D. Clustering

Clustering is a crucial part of our summarization approach. After performing vectorization, the next task is to get the clusters of similar sentences. The basic intuition behind clustering is to group the similar sentences into K clusters. Here, the value of K depends on the number of sentences we want in our summary. In other words, K is the sentence count of the summary. The similarity between the sentences is computed using the cosine distance. Lesser the distance, more is the similarity. Hence, as mentioned earlier, this similarity metric is used in the clustering process of this project. From the obtained clusters of sentences, we find the centroid of each cluster. At last, we are left with a K number of centroid sentences. These centroid sentences will form the summary of that particular document.
Having this intuition, we have performed clustering from scratch (without using any inbuilt library) using the K-Means algorithm. The standard K-Means algorithm takes into consideration the euclidean distance between the vectors. However, euclidean distance is not a suitable metric to compute the similarity between the sentence vectors, hence we have used the cosine distance instead of euclidean distance. Firstly, the centroids are initialized to the random sentence vectors of the document in the corpus. Next, we find the cosine distance between all the data-points and the centroids. The data-points closer to a particular centroid are appended to that centroid's cluster. This process is repeated until some specified maximum iterations and the centroids start getting unchanged. In the implementation, we have set the sentence count (K) and the maximum iteration (max_iter) to be variables so that they can changed as per the user needs. This is how the clustering algorithm will be applied to all the documents in the corpus and extract the centroids of all clusters. Now, one challenge usually encountered here is that the centroids may not be the exact sentence vectors of our corpus. For tackling this issue, we have found the least

distance vector with the centroid vector of that particular cluster. This sentence vector is then appended to the selected set of summary vectors. Same process is applied for all clusters. The sentences corresponding to the selected sentence vectors form the summary of that particular document.

E. Ranking

A summary represents the gist of the document and hence, it should be cohesive in nature, that is, it should be ordered. We already have the set of selected sentences that form the summary. So, our next task is to rank them appropriately. For ranking, we have mapped the indices of the selected sentences with the raw document indices. Then the sentences are ranked according to the sorted order of the indices.

F. Summary Generation

After ranking, we are left with a list of ordered sentences. The number of these sentences depends on the sentence count set by the user. These sentences are stored in a text file, which is our final summary. Furthermore, this summary is compared with the provided reference summaries for evaluating the model.

IV. Evaluation of Model

To get the quality of the ATS systems we need to evaluate automatic generated summary. It can be done in two ways 1) Manual Evaluation or 2)Automatic evaluation. For manual evaluation, human judges may be asked to judge the generated summary on the basis of some metrics like readability, content coverage and many more. Manual evaluation consumes a lot of time and effort. Information retrieval oriented tasks can be used for automatic evaluation. We have used ROUGE-N score for evaluation of our ATS system. The main idea of rouge score is that it counts the overlapping N-grams between the generated and the reference summary. We have used ROUGE1 and ROUGE2 to calculate precision, recall and f-score of our automatically generated summary with the reference summary fetched from the data. ROUGE scores can be calculated using

$$Precision = \frac{number\_of\_overlapping\_terms}{total\_words\_in\_system\_summary}$$

$$Recall = \frac{number\_of\_overlapping\_terms}{total\_words\_in\_reference\_summary}$$

$$F\text{-}measure = \frac{2}{precision^{-1} + recall^{-1}}$$

V. Experimental Results

We have compared the ROUGE1 score of summaries generated by using two representations of document vector 1)Tf-idf and 2)Word2Vec and have also generated summaries with different sentence counts ranging from 5 to 19.

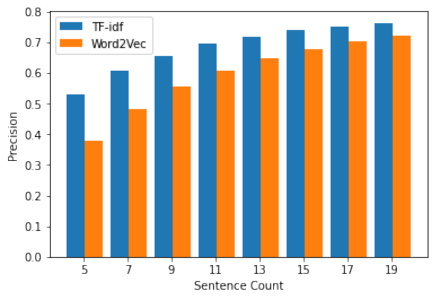

Fig. 1. Rouge-1 precision

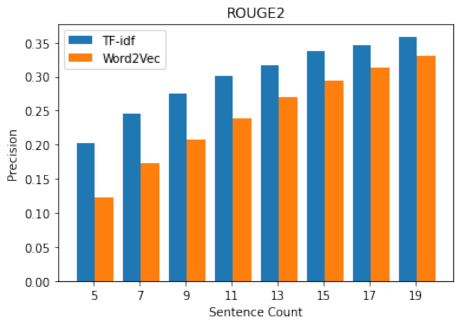

Fig. 2. Rouge-2 precision

Figure 1 shows the comparison of Tf-idf and Word2Vec methods along with their ROUGE1 precision score for summaries with different sentence counts, *Figure 2* shows same for ROUGE2. Results shows that Tf-idf outperforms the Word2Vec representation in both ROUGE1 and ROUGE2. Summaries with higher sentence count have good precision as more number of words would be overlapping with the reference summary.

Score Metric	Sentence Counts	Precision	Recall	F-Measure
Rouge1	5	0.5319	0.2068	0.291
	7	0.6093	0.1699	0.2612
	9	0.6571	0.1441	0.2328
	11	0.6972	0.127	0.2118
	13	0.7178	0.1133	0.1929
Rouge2	5	0.2019	0.0761	0.1082
	7	0.2464	0.0664	0.103
	9	0.2754	0.0586	0.0954
	11	0.3007	0.0533	0.0895
	13	0.3159	0.0491	0.0839

TABLE I
TF-IDF

Table 1 and *Table 2* show ROUGE scores for Tf-idf and Word2Vec representation of documents respectively. ROUGE1

Score Metric	Sentence Counts	Precision	Recall	F-Measure
Rouge1	5	0.384	0.1749	0.2341
	7	0.4814	0.1531	0.2276
	9	0.5502	0.1342	0.2124
	11	0.6004	0.1202	0.1973
	13	0.6453	0.1099	0.1853
Rouge2	5	0.1223	0.0532	0.0722
	7	0.1721	0.0522	0.0786
	9	0.207	0.0481	0.0769
	11	0.2386	0.0459	0.076
	13	0.2693	0.0446	0.0757

TABLE II
WORD2VEC

compares overlapping uni-grams while ROUGE2 compares overlapping bi-grams, so ROUGE2 scores for both Tf-idf and Word2Vec is less as compared to ROUGE1.

VI. Conclusion

Manual text summarization is time consuming and costly. ATS systems are widely used to solve this problem. It is difficult for the computer to find the “most important” sentences in the document as compared to the human generated summary. It is comparatively more difficult to generate abstractive summary as compared to the extractive summary. We have compared the summaries generated using Tf-idf and Word2Vec representations. Tf-idf gives higher ROUGE1 and ROUGE2 score as compared to Word2Vec. Different approaches can be used for representing the documents, ranking the sentences and creating summaries. ATS systems can be tested by using different document representation and by using different distance metrics for ranking the sentences.

VII. Contributions

All team members read various approaches discussed in the chosen paper [3] in detail and four different approaches were decided initially of which one approach was chosen. The algorithm for the chosen approach has been collectively formulated by all the members and the coding part has also been performed in several meetings over the last month where everyone was present and contributed equally.

A. Sana - 202011019

In addition to writing the section of preprocessing and vectorization (Tf-IDF and word2Vec), I have contributed as follows: 1) Worked with Shradha for collection (web scraping) of data and understanding the format of the dataset selected, 2) Wrote the code for word2vec vectorisation and code for preparing the document vectors for the dataset. Worked with Shivangi to integrate the clustering component, written by her and 3) Worked with the entire team for final proof reading and optimization of the code.

B. Shivangi - 202011023

In addition to the report writing sections: Clustering, Ranking and Summary Generation, I have contributed in following implementation components: 1) Helped in finding the suitable dataset(CNN) for the implementation from the given list of datasets in the survey paper provided to us, 2) Scratch coded the sentence clustering algorithm (general for TFIDF and Word2Vec) and with the co-ordination of all other team members, final optimization of the code was done and 3) Co-ordinated for the code of Ranking, which is written by Mantra.

C. Shradha - 202011029

In addition to the report writing section: Types of Automatic Summarizers, I have proof-read the report and made necessary changes for consistency. I have also contributed in following implementation components: 1) Worked with Sana on collection of data from official dataset URL by performing web scraping, 2) Wrote the code for finding ROUGE scores for tf-idf and word2vec and with the co-ordination of all other team members, final optimization of the code was done and 3) Proof read the code for preprocessing the data written by Hari Charan.

D. Mantra - 202011041

In addition to the report writing sections: Evaluation of Model, Experimental Results and Conclusion, I have contributed in following implementation components: 1) Wrote the code for ranking sentences and summary generation for Tf-idf and Word2Vec using the clusters which were formed after clustering, 2) Proof read the code for collection of data written by Sana.

E. Hari Charan - 202011056

In addition to the report writing sections: Abstract and Introduction, I have contributed as follows: 1) Wrote the code for Tf-idf vectorisation and code for preparing the document vectors for the dataset, 2) Proof read the code for computing the rouge scores for evaluation of the summary written by Shradha. 3) Worked with the entire team for final proof reading and optimization of the code.

References

[1] https://github.com/abisee/cnn-dailymail.

[2] Text Summarization using Clustering Technique, international journal of engineering trends and technology (ijett).

[3] Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165:113679, 2021.

[4] Liwei Hou, Po Hu, and Chao Bei. Abstractive document summarization via neural model with joint attention. In Xuanjing Huang, Jing Jiang, Dongyan Zhao, Yansong Feng, and Yu Hong, editors, *Natural Language Processing and Chinese Computing*, pages 329–338, Cham, 2018. Springer International Publishing.

[5] Karel Jezek and J. Steinberger. Automatic text summarization. *Znalosti*, pages 1–12, 01 2008.

[6] Krithi Shetty and Jagadish S. Kallimani. Automatic extractive text summarization using k-means clustering. In *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, pages 1–9, 2017.

[7] Amol Tandel, Brijesh Modi, Priyasha Gupta, Shreya Wagle, and Sujata Khedkar. Multi-document text summarization - a survey. In *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*, pages 331–334, 2016.