A Study of Automatic Text Summarization System based on MMR and TextRank

Khushhal Jindal, Muhammad Uzair Tariq and Rohan Singh Rajput
Department of Computer Science
University of Central Florida, Orlando, FL 32716
jindalkhushhal@gmail.com (3908033) | uzairtq@gmail.com (3672637) |
rohan.rajput4@gmail.com (3914579)

Abstract

In this paper, an automatic text extractive summarization system is constructed to generate text summaries of the given dataset. Two techniques are proposed for the task of automatic text summarization: Maximal Marginal Relevance (MMR) and TextRank. The MMR criterion strives to reduce redundancy while maintaining query relevance in re-ranking retrieved documents and in selecting appropriate passages for text summarization. Initial results highlight some pros of MMR diversity ranking in document retrieval and in single document summarization. The clear and distinct benefit is demonstrated in constructing non-redundant multi-document summaries, where MMR results are clearly superior to non-MMR passage selection. On top of it we also implement TextRank – a graph-based ranking model for text processing, and show how this model can be successfully used in natural language processing applications. In particular, we used MMR and TextRank to attempt better performance for text summarization on given standard dataset.

Keywords– MMR, TextRank, Text Summarization, Extractive Summary

1 Introduction

An automatic summarization system is a system which is used to generate summaries as similar to the human summaries given a collection of text documents by means of a computer. Goal of such system is to present text of document in shorter form. Summary created will keep most important points from the original document. When a user search something on the web, the response obtains vast amount of information which is very difficult for him to read completely. Summarization is quite helpful for the users as it reduces this reading time. Text summarization is useful in order to create an abstract of an entire document, by including the most informative sentences. Text summarization is an important step for the task of information management. Text summarization is a challenging task because it requires understanding the long sentences and jargons in a document. Another major challenge of text summarization is to maintain coherence i.e. whether different parts of a summary comes together to create a consolidated sequence or not. For making this summary as accurate as possible, there is a need to establish an 'ideal summary' so that the system summary can be matched against it. However, it is very difficult to construct such an 'ideal summary'.

One existing approach which resembles our approach for automatic text summarization is LexRank - an approach to define sentence salience based on graph-based centrality scoring of sentences. It constructs the similarity graph of sentences compared to the centroid approach, which is prone to over-generalization of the information in a document cluster.

Online information has been growing rapidly with time and hence it has become increasingly important to provide improved mechanisms to find information quickly. The conventional IR systems rank and assimilate documents based on maximizing relevance to the user query. When we have few relevant documents, or in the cases where very-high recall is necessary, pure relevance ranking is apt. However, when there is a large number of potentially relevant documents, highly similar to each other or even containing partially or fully duplicative information one must look for solution beyond pure relevance for document ranking. A new document ranking method is one where each document in the ranked list is selected according to a combined criterion of query relevance and novelty of information. The best is the method called Maximal Marginal Relevance (MMR) which provides precisely such functionality.

Our major contribution is towards applying a baseline state-of-the-art system MMR to given data set for text summarization along with introduction of a new approach towards automatic summarization to try to achieve better result than the existing MMR system.

Graph-based ranking algorithms like Kleinberg's HITS algorithm (Kleinberg, 1999) or Google's PageRank [8] have been successfully used in citation analysis, social networks, and the analysis of the link-structure of the World Wide Web. A graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. Applying a similar line of thinking to lexical or semantic graphs extracted from natural language documents, results in a graph-based ranking model that can be applied to a variety of natural language processing applications, where knowledge drawn from an entire text is used in making local ranking/selection decisions.

We specifically make use of the TextRank graph-based ranking model for graphs extracted from natural language texts. It is an unsupervised learning approach for sentence extraction which eliminates the need of training data and uses the Graph-based ranking approach for the summarization; voting and recommendation are the two basic idea behind it. Linkage of one vertex to another cast a vote to that vertex. Higher vote reflects higher importance to the graph and then the model is ranked according to that. Two important unsupervised approaches have been taken in this state-of-the-art system; keyword and sentence extraction. These approaches can yield better result for text summarization.

2 Problem Formulation

Given, an input document Ɗ containing a set of 50 topics T. Each topic from the set T is related with a set D of 10 news documents referred to as a "meta-document". After generating the meta-document, the pre-processing has been applied, this processed data then propagates towards two approaches which we are going to discuss next. Various standard libraries have been used to process the data for good and credible result.

3 Our Approach

3.1 Maximal Marginal Relevance

The first approximation to measure relevant novelty is to measure relevance and novelty independently and provide a linear combination as the metric. The paper [1] mentions this linear combination as "marginal relevance" - i.e. a document has high marginal relevance if it is both relevant to the query and contains minimal similarity to previously selected documents. Our aim is to maximize-marginal relevance in retrieval and summarization, hence this method is labelled as "maximal marginal relevancy" (MMR).

$$MMR \stackrel{def}{=} Arg \max_{D_i \in R \setminus S} \left[\lambda (Sim_1(D_i, Q) - (1-\lambda) \max_{D_j \in S} Sim_2(D_i, D_j)) \right]$$

where R = IR(C, Q, theta), i.e., the ranked list of documents retrieved by an IR system; given C (a document collection or document stream), Q (a query or user profile); and a relevance threshold theta, below which it will not retrieve documents (theta can be degree of match or number of documents); S is the subset of documents in R already selected; R\S is the set of as yet unselected documents in R; Sim1 is the similarity metric used in document retrieval and relevance ranking between documents (passages) and a query; and Sim2 can be the same as Siml or a different metric. Given the above definition, MMR computes incrementally the standard relevance-ranked list when the parameter lambda=1, and computes a maximal diversity ranking among the documents in R when lambda=0. For intermediate values of lambda in the interval [0,l], a linear combination of both criteria is optimized. Those wishing to sample space around the query, should set lambda at a smaller value, while those who want to focus on multiple potentially overlapping or reinforcing relevant documents, should set lambda to a value closer to 1. [1] indicates that a particularly effective search strategy (reinforced by the user study discussed below) is to start with a small lambda (e.g. lambda = 0.3) in order to understand the information space in the region of the query, and then to focus on the most important parts using a reformulated query possibly via relevance feedback) and a larger value of lambda (e.g. lambda = 0.7).

First approach is to implement the basic MMR technique [1] to extract and rank the sentences out of a meta document according to a certain criterion. The criterion requires that a sentence should have a high similarity with the document and at the same time a low similarity with already selected sentences. The following equation calculates the MMR score to rank each sentence according to the above criterion

$$MMR(s_i) = \lambda Sim_1(s_i, D) - (1-\lambda) \max_{s_j \in S} Sim_2(s_i, s_j)$$

The implementation of this technique requires to have a bag-of-words representation of si, sj and D. This means that the meta document has to be preprocessed before calculating the MMR score. Following are the steps that are taken for the preprocessing:

1. Firstly, all the documents from all the folder are fetched.
2. A meta-document with a 10 document would be created by concatenating all the sentences.
3. Then word preprocessing would perform with following pipeline:
   a. Sentence Segmentation: NLTK toolkit [2] has been used for segmenting the text into sentences which is important before tokenizing the text into words.
   b. Word tokenization: NLTK toolkit has been used to derive word tokenization.
   c. Word lowering: We use in built function to convert words to the lowercase.
   d. Stop Word removal: We use NLTK toolkit to remove the stopwords for the documents.
4. After this step we match the current document (s_i) with meta-document (D) by a cosine similarity function (Sim1) Again, for second term we apply the Sim2 similarity function for the previous and next sentences s_j and s_i respectively.
5. After that we vary the different values of lambda for getting maximum value of MMR.
6. This value of MMR will be saved in an array from which we will pick up the highest MMR value sentences up to 100 words for creation of the summary.

3.2 TextRank

An application of graph-based ranking algorithms is entitled to the texts in natural language to be able to construct a graph representing texts and interconnected words. When an entity is added to the graph, following are the main steps of application of these graph-based ranking algorithms as mentioned in [2]:

1. Identify text units which are most important since they can ideally define the task at hand, and add them as vertices in the graph.

2. In a graph, relations are used to draw edges between vertices that connect by first identifying these relationships such as text units. These edges can be directed or undirected, weighted or unweighted.

3.Iterate these graph-based ranking algorithms used until convergence.

4. Finally, vertices are sorted based on their final score. Ranking decisions are made according to these values associated with each vertex.

TextRank's application is evaluated on natural language task of sentence extraction which is based on ranking of texts. Through this sentence extraction we are able to find the most important sentences in the texts used to build extractive summaries later on.

1. Ranking Algorithm

Terminology:

- G = (V, E) a directed graph with vertices V and edges E
- In(Vi) = predecessors of Vi
- Out(Vi) = successors of Vi
- d is the damping factor ∈ [0,1] (usually 0.85) that indicates the probability to jump to a random page

$$S(V_i) = (1-d) + d \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

Assign a random initial value to each vertex in the graph and iterate the scoring function until convergence (on text: 25-30 iterations). Scores are based on PageRank [8]

Ranking algorithms are traditionally applied on directed graphs. However, they can also applied be to undirected graph if the convergence is more gradual. *Weights* can model the *strength* of the relations between textual units. Original definition of ranking algorithms assumes unweighted graphs. [2] introduces new ranking formula to take into account edge weights

$$WS(V_i) = (1-d) + d \sum_{j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

TextRank approach is the second approach used for the task of text summarization. In this paper we are taking TextRank approach for the creating automatic summarization. TextRank basically is a graph ranking algorithm in which sentences are treated as vertices and edges are the weights are how similar the sentences are in that graph.

The first step towards TextRank approach is tokenizing all the text into sentences, Stanford standard NLTK library has been used for this preprocessing. After tokenizing the sentences, each sentence is tokenized into collection of words. 'PunktSentenceTokenizer' module has been used for this. After completing these steps, the

sentences are converted into graphs, we use Scikit-learn 'Tf-idfTransformer' module for changing the sentences into graphs. After converting sentences into graphs the PageRank algorithms has been used for marking the score to the graph. The NetworkX pagerank function has been used for this scoring. This provides the mapping of sentences along with the indices to the scores. After getting the scores we sort them out to get the higher scoring sentences. These sentences are taken as the summary of the corresponding meta document.

TextRank approach is supposed to work well because it takes into account information recursively drawn from the entire graph rather than depending on a vertex's local context. TextRank is capable of identifying connections between various entities in a text, by constructing the graphs on the whole text & implementing recommendation concept [2].

4 Dataset

There is currently much interest and activity aimed at building powerful multi-purpose information systems. The agencies involved include DARPA, ARDA and NIST. Their programs, for example DARPA's TIDES (Translingual Information Detection Extraction and Summarization) program, ARDA's Advanced Question and Answering Program and NIST's TREC (Text Retrieval Conferences) program cover a range of subprograms. These focus on different tasks requiring their own evaluation designs.

Within TIDES and among other researchers interested in document understanding, a group grew up which has been focusing on summarization and the evaluation of summarization systems. Part of the initial evaluation for TIDES called for a workshop to be held in the fall of 2000 to explore different ways of summarizing a common set of documents. Out of the initial workshop and the roadmapping effort has grown a continuing evaluation in the area of text summarization called the Document Understanding Conferences (DUC). Sponsored by the Advanced Research and Development Activity (ARDA), the conference series is run by the National Institute of Standards and Technology (NIST) to further progress in summarization and enable researchers to participate in large-scale experiments.

The successor of DUC [11] is TAC (Text Analysis Conference). The Text Analysis Conference (TAC) is a series of evaluation workshops organized to encourage research in Natural Language Processing and related applications, by providing a large test collection, common evaluation procedures, and a forum for organizations to share their results. TAC comprises sets of tasks known as "tracks," each of which focuses on a particular subproblem of NLP. TAC tracks focus on end-user tasks, but also include component evaluations situated within the context of end-user tasks.

5 Experiments

Experiments are conducted using python (2.7.11) programming language. It starts with preprocessing of the data with Stanford NLTK library [6]. Preprocessing pipeline includes the removal of start tags & white space. After removal of these tags we start sentence tokenization, the sentence tokenization along with different other preprocessing is done by NLTK Stanford libraries. After sentence tokenization, other processes are word tokenization, word lemmatization, and stopword removal. After carrying out all these preprocessing steps, we append the all the words to a list to create a sentence list. This process goes on to continue for every 10 documents. This combines the whole one meta document. After completing preprocessing steps we use word to vectorization technique to calculate similarity. The Scikit-learn library [7] has been used for the Cosine Similarity, MMR function has been evaluated after that with various values of λ.

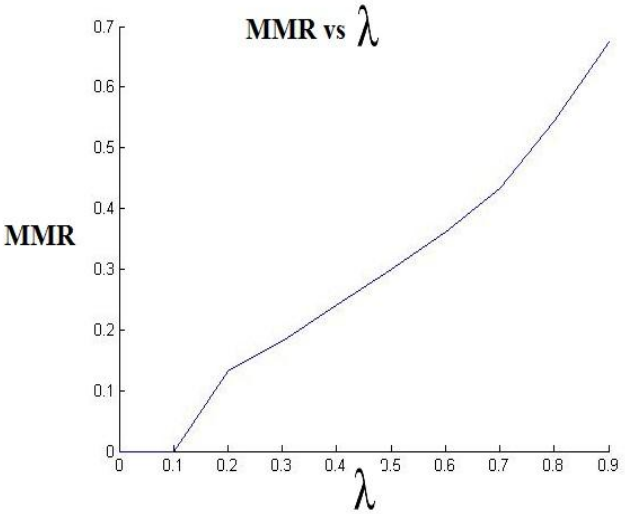

Fig 1: MMR score vs λ

This λ parameter is used to amount of diversity and redundancy to the selection, the wide range of λ has been used ranging from 0 to 1. For each value of λ, maximum value has been calculated and top scoring sentence has been selected as summary sentence. For the current experimental setup, the sentence summary has been restricted to less than 100 words. Similarly, for all the 50 data set, the summaries have been deduced after

calculation of the summary we evaluate our score with Rouge toolkit [9]. ROUGE is an automated summarization evaluation package which is very highly correlated with the human evaluations. This Toolkit compares the data set with human summary and provide various Rouge Scores which is a defining metric used for the comparison with different natural language processing approaches. Five state-of-the-art systems like Centroid, DPP, ICSISumm, LexRank, Submodular are the compared with the MMR approach. Table 1 reports the ROUGE scores obtained for MMR and TextRank against these state-of-the-art systems.

As seen from the experiments, we are collecting different MMR values with different λ values as MMR values is seen as a function of λ. This relationship can be seen in Figure 1.

The MMR approach has fairly produced the similarity with Human Summaries, however there exist some sentences that have been missed. This could be improved by doing more preprocessing and assigning good similarity functions. Here, we are using our own word2vector approach, there are several other word2Vec toolkits such as Kaggle Word2Vec. There has been a fine score generated by our TextRank approach, however there is still a room for improvement by providing some supervised learning approach like labelling of small portion of the data. There exist other approaches like deep learning neural network which can be combined with TextRank to produce better results.

Table 1: Summarization results evaluated by ROUGE (%)

	ROUGE-1			ROUGE-2			ROUGE-SU4		
System	**P(%)**	**R(%)**	**F(%)**	**P(%)**	**R(%)**	**F(%)**	**P(%)**	**R(%)**	**F(%)**
CENTROID	35.93	36.41	36.61	7.85	7.97	7.91	12.23	12.40	12.31
DPP	39.70	39.79	39.74	9.59	9.61	9.60	13.82	13.85	13.83
ICSISUMM	38.57	38.40	38.48	9.83	9.78	9.80	13.68	13.61	13.65
LEXRANK	35.89	35.95	35.92	7.44	7.49	7.45	11.89	11.91	11.90
SUBMODULAR	39.25	39.18	39.21	9.36	9.34	9.35	13.76	13.74	13.75
MMR	33.8	34.25	33.49	6.62	6.58	6.744	11.33	11.21	11.47
TEXTRANK	33.7	33.92	33.51	6.95	6.98	6.92	11.35	11.42	11.29

6 Related Work

There are various approaches which can be used to extract summaries. While implementing summarization approach of our own, motivation to choose 'TextRank' approach was the study by Rada Mihalcea and Paul Tarau from University of North Texas on "TextRank: Bringing Order into texts" [2]. Most existing systems assumed feature independence and relied on Naive-Bayes methods, others have focused on the choice of appropriate features and 3 on learning algorithms that make no independence assumptions. significant approaches involved hidden Markov models and log-linear models to improve extractive summarization. TextRank is an unsupervised learning approach which does not require any training data. Infact, it relies on inherent properties of text to produce the output by determining keyphrases which seems crucial in the text.

Automated document summarization dates back to Luhn's work at IBM in the 1950's [10], and evolved through several efforts to the recent TIPSTER effort which includes trainable methods [5], linguistic approaches and our information-centric method [4], the first to focus on anti-redundancy measures.

7 Conclusion and Future Work

Automatic text summarization is an important and challenging task in Natural Language Processing, this paper we attempt to create automatic extractive summary. Two approaches have been taken for creating extractive summary, first state-of-the-art system is MMR (Maximal Marginal Relevance), this is a simple but powerful approach to create automatic summary.

The experiment shows the MMR depends upon the choice of diversity and redundancy in the data. As from derived result it can be seen that the MMR provides good summary of given DUC data set, the performance measure was done by using Rouge ToolKit. Second approach is our approach that is TextRank, as TextRank is a graph ranking unsupervised approach, the performance improvement was expected with TextRank. The results show quite comparable outcome with MMR and other state-of-the-art systems. We used standard library to calculate the summary which provides credibilities in the result also the time of execution is also quite less than other state-of-the-art systems. There are many possible extensions to this work. The unsupervised learning is quite popular among the NLP community, we would like to explore further partial involvement of supervised learning by labelling a portion of data. This approach can be tested for various other datasets to provide more insight on the results. Along with this, one ambitious approach we would like to explore is using Deep Learning techniques, there exists good libraries and tool kits which are easily available today allowing us to have a deep analysis of NLP summarization. TensorFlow from Google is one good example. Also, we can utilize parallel programming to explore more amount of data set to train our Neural Network to provide better results.

References

[1] J. Carbonell and J. Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. *In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336

[2] R. Mihalcea and P. Tarau. TextRank: Bringing Order into Texts. *In Proceedings of EMNLP 2004*, pages 404-411, Barcelona, Spain Association for Computational Linguistics.

[3] D. Das and A.F.T. Martins. A Survey on Automatic Text Summarization. Literature Survey for the Language and Statistics II course at CMU, November 2007.

[4] J.G.Carbonell, Y. Geng, and J. Goldstein. Automated query-relevant summarization and diversity-based reranking. *In 15th International Joint Conference on Artificial Intelligence, Workshop: AZ in Digital Libraries*, pages 9- 14, Nagoya, Japan, August 1997.

[5] J.M. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. *In Proceedings of the 18th Annual Znt. ACM/SZGZR Conference on Research and Development in ZR*, pages 66-73, Seattle, WA, July 1995.

[6] Natural Language Toolkit. http://www.nltk.org/.

[7] Scikit-learn: Machine Learning in Python. http://scikit-learn.org/stable/.

[8] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Technical Report 1999-66, Stanford InfoLab*, November 1999. Previous number = SIDLWP-1999-0120.

[9] Chin-Yew Lin. Rouge: A Package for Automatic Evaluation of summaries. *In Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004*, Barcelona, Spain.

[10] P.H. Luhn. Automatic creation of literature abstracts. *IBM Journal*, pages 159-165, 1958.

[11] DUC Dataset. http://www-nlpir.nist.gov/projects/duc/intro.html