Text Summarization for IR with a Novel Algorithm for Query-Based Summarization from a Large, General Corpus

Ethan Sorrell

May 1, 2019

Contents

1 Introduction

Text summarization is a field which has seen growth for many of the same reasons as the field of information retrieval: that is, the growth of the internet has led to large quantity of information becoming available. As a result, there has been a greater desire to be able to automatically summarize this

information. While text summarization is generally seen as a problem in the realm of natural language processing, the field holds many applications in information retrieval as well as sharing the use of many techniques and ideas [1]. It will be assumed for the purposes of this paper that the reader has some familiarity with the field of information retrieval, but no similar assumption will be made for the field of natural language processing. The field of information retrieval already commonly makes use of automatic text summarization, with most search engines displaying snippets from returned web pages. Additionally, there exist techniques for generating summaries with respect to a query as well as multi-document summarization techniques. However, in both cases the corpus is usually composed of only a handful of documents which usually all focus on a common central topic [2]. While techniques for generating a query-based summary from a large, general corpus may exist, the author was unable to find any at the time of writing. Regardless, it is safe to say that this is not a commonly explored topic. In this paper, we will begin with a brief survey of automatic summarization, drawing parallels to information retrieval techniques where relevant. We will then introduce our chosen problem before walking through the proposed algorithm for tackling this problem. This will be followed with a discussion of the various parameters which dictate the performance of this algorithm, and a look at how these may be chosen. At this point we will briefly discuss the problem of evaluation. Finally, we will conclude with an overview of text summarization and its applications to information retrieval as well as possible future works in this area.

2 Text Summarization

2.1 Abstractive vs Extractive

Text summarization techniques are generally broken down into abstractive summarization and extractive summarization. Extractive methods generate summaries by performing extraction. That is, they identify the most relevant sections of the original text, and produce a summary by concatenating these most relevant sections verbatim [1]. This is generally done by using some scoring or ranking methods to assign a quantitative value to each section, then selecting those sections which maximize this value [3]. Abstractive summarization, on the other hand, aims at producing summaries by developing systems which are able to interpret the text, usually using natural language processing techniques, with the aim of generating a summary composed of entirely new text which reflects the most important information from the

original text [2]. Though the exact aim of summarization is not well agreed upon, the most popular automatic evaluation metrics work by comparison to a human generated summary [4]. Thus it may be considered that the goal of automatic summarization is to create a summary most similar to one generated by a human. To this end, it should be noted that humans tend produce abstractive summaries; they don't simply choose sentences from the text, but rather generate entirely new sentences in order to create a summary. For this reason, abstractive summarization seems appealing. However, in this paper we will focus our attention on extractive summarization techniques. This decision is made for a few reasons. First, extractive summaries have seen more success historically, and indeed a majority of existing summarization methods fall into this category [2]. Additionally, extractive summarization techniques will be more familiar to the intended reader, coming from the field of information retrieval. As yet another consideration, an extractive summary will serve the intended purposes of our proposed algorithm, giving us little reason to consider the large amount of additional complexity implicit in abstractive summarization methods.

2.2 Extractive Summarization

Summarizers can be thought of as being composed of three tasks: 1) creating an intermediate representation of the text, 2) using some scoring metric or otherwise to assign a numeric importance value to the sections of the text (usually sentences), 3) select the most important sentences [4]. It is common to use a greedy, top-n approach to select the most important sentences though other techniques exist.

2.3 Term Frequency-Inverse Document Frequency

Now, we will introduce a relatively simple approach to extractive summarization, and one that relies on a familiar concept from information retrieval: TF*IDF weighting. In some text summarization contexts, there may be a need to draw these TF*IDF weightings from a background corpus as the, oftentimes single document, corpus for the summarization may be insufficient. To formalize this approach, the weight of each word w in document d where $d \in D$ is given by:

$$q(w) = f_d(w) * log\frac{|D|}{f_D(w)} \tag{1}$$

Here, $f_d(w)$ give the term frequency of word w in document d and $f_D(w)$ gives the document frequency of word w among document collection D [2].

TF*IDF is simply a weighting scheme. However, to give an idea of how it could be used in an extractive summarization method, we consider the simplest approach. From this weighting scheme, we could consider the importance of a sentence to be the sum of the weights of the words in that sentence. We could then create a summary using a top-n approach. TF*IDF weighting is sometimes criticized as it has little substantial theoretical backing. However, it is easily computed, requires no additional information such as an ontology, and has the benefits of not requiring a stop-word list. As a result, many existing methods incorporate this weighting scheme [5].

2.4 Clustering to Derive Topics

In multi-document classification there is often the concept of deriving topics from the multi-document corpus before summarization. One method for accomplishing this is to use the, well-known in information retrieval, Rocchio algorithm. However, one drawback of this approach is that the number of topics or clusters must be manually selected. As an alternative, the field of text summarization has CIDR. Here, each document is represented as a TF*IDF vector generated from the words in that document. From here the algorithm begins sequentially processing the documents. The first document is placed in a new cluster of its own. Then each additional document has its cosine similarity computed with all existing cluster centroids. If the most similar cluster has a similarity that is above a pre-defined cut-off value, then the document is placed in the cluster, otherwise the document creates a new cluster of its own [6]. Since this algorithm is able to classify all documents in a single-pass, it is able to perform on very large collections of documents.

2.5 Bayesian Topic Model using Kullbak-Liebler Divergence

One interesting idea that has seen some use in automatic summarization is the use of Kullbak-Liebler divergence for scoring sentences. Though there have been various applications of this technique, we will look at one that use the divergence between a given sentence and the entire document in order to determine the weight for that sentence. However, first we must consider the more general framework of word probability. In this framework, the probability of a given word w is given by:

$$p(w) = \frac{f_I(w)}{|I|} \tag{2}$$

where $f_I(w)$ gives the number of occurences of word w in the input, and $|I|$ gives the overall length of the input in number of words [5]. Using this

framework, we may consider the Kullbak-Liebler diverence. This metric is used to compute the difference between two distributions. Thus, we can we can use our word probability framework, to calculate the divergence of a sentence's distribution from its containing document. This difference is given by:

$$D_{KL}(P,Q) = \sum_{w} P(w) log \frac{P(w)}{Q(w)} \tag{3}$$

where $P(w)$ is the probability of word w is the sentence, and $Q(w)$ is the probability of word w in the document [2].

2.6 TextRank or How PageRank is Applicable

PageRank is one of the most well-known algorithms in information retrieval, and has shown a great degree of success. This gave spawn to TextRank, a PageRank inspired algorithm in the realm of text summarization. This algorithm works by creating a graph $G = (V, E)$ from our corpus. Here the set of vertices V can be composed of a chosen unit of text. For demonstration, let us consider it to be a sentence. Then, edges can be created using a chosen similarity metric. Generally, these edges will be weighted and undirected. Let us define these edges E using cosine similarity between the sentences they connect. Now the score of each vertex V_i can be calculated by:

$$S(V_i) = (1-d) + d * \sum_{j \in C(V_i)} \frac{w_{ji}}{\sum_{V_k \in C(V_j)} w_{jk}} S(V_j) \tag{4}$$

where $C(V_i)$ is the set of vertexes connected to V_i and d is a damping factor usually chosen to be 0.85, though it may take any value between 0 and 1 [7]. Note that compared to PageRank, this algorithm uses a weighted, undirected graph. Like in PageRank, the above scoring is performed successively until the value reaches convergences within a certain threshold [7]. This is the main variable parameter in this algorithm, and should be chosen based upon available computational resources.

3 Proposed Algorithm

For our proposed algorithm, we are assuming a large general corpus, such as the type commonly found in information retrieval problems. From this corpus, we aim to create an extractive summary with respect to a query. As a rough overview, our algorithm will first filter out documents which offer no value to our summary. Then we will extract topics from our remaining

corpus. From these topics, we will select only the few most central topics for our summary. Finally, we will create our summary from concatenating the most relevant sentences from these most central topics. We begin by encoding each document as a TF*IDF vector from the words in that document. We then filter the our corpus to those documents which are somewhat relevant to our query. We do this by using the cosine similarity between the document and query, discarding those documents with a similarity below a certain threshold. We should choose this threshold such that we only discard documents which provide us with no information with respect to our query. This means, the threshold should be higher than would be typical for information retrieval purposes. Next, we apply CIDR to cluster our remaining documents into topics. Recall that CIDR is parameterized by a similarity threshold for creating new clusters. We will save discussion of selection of this parameter for later, but we will label it $T_{cluster}$. We now apply TextRank such that our chosen unit of text is a document cluster. This will score the relevance of each topic. We then select N_{topics} topics using a top-n approach. For each of these topics we compute the relevance of the sentences of that topic using Kullbak-Liebler divergence. Using this score, we select $N_{sentences}$ sentences from each topic again using a top-n approach. Finally, we create our summary by concatenating these $N_{sentences}$ from each of our N_{topics}. This algorithms has several parameters which control its performance, but of particular importance are the highest level parameters. That is, $T_{cluster}$, $N_{sentences}$, and N_{topics}. Both $N_{sentences}$ and N_{topics} control the length of the final summary. Additionally, $T_{cluster}$ controls how homogenous an individual cluster will be. As a result, it could be possible to decrease $N_{sentences}$ and N_{topics} such that the size of the summary does not change. Simultaneously, the threshold $T_{cluster}$ could be modified to expand or reduce the scope of each of these clusters. Thus the selection of these parameters appears somewhat ambiguous, and it will likely vary depending on the specific application of the algorithm.

3.1 Further Considerations

This algorithm is ofcourse not the only possible algorithm for tackling the specified problem. However, the proposed algorithm is easily understood from its components. This process demonstrates how a novel algorithm may be created to address a specific need which may be encountered in the realm of information retrieval. Additionally, it accomplishes this while remaining somewhat accessible. Requiring little prior knowledge, this algorithm introduces many of the concepts common in extractive summarization. Indeed,

there is room for optimization. Many of the chosen algorithms are not state of the art, but rather the author has favored simplicity over cutting edge performance. In the concluding section we will discuss some of the most obvious avenues for optimization as well as other areas of automatic summarization which are noteworthy that we have not discussed thus far.

3.2 Evaluation

There is no universally agreed upon method for evaluating the goodness of a summary. This problem is further complicated by the fact that many summarization algorithms, such as our proposed algorithm are created to address a specific need such that too general of an evaluation metric may not apply. In general, existing methods are divided between automatic evaluation metrics which operate by comparison to a provided human summarization, and manual evaluation which tends to be performed by judges at a relevant conference such as DUC (Document Understanding Conference) or TAC (Text Analysis Conference) [8]. Of the automatic evaluation metrics the ROUGE family is particularly popular. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation and originally included four different metrics: ROUGE-N, ROUGE-L, ROUGE-W, AND ROUGE-S. There are extensions of these such as ROUGE-SU which is an extension of ROUGE-S [4]. We will explore three of the most common of these metrics.

ROUGE-N This is sometimes called the n-gram co-occurrence statistic. It operates by comparison of n-grams in the candidate and reference summary. Note, that an n-gram is simply a sequence of n consecutive words. Let p be the number of n-grams shared between the candidate and reference summary. Let q be the total number of n-grams drawn from the reference summary. Then the score is calculate by [2]:

$$ROUGE - N = \frac{p}{q} \tag{5}$$

ROUGE-L This measure operates using the longest common subsequence. That is, the longest subsequence which is found in both the reference and candidate summary. There are many variations of this measure, but we will look at the F-measure based variation. Let X be our candidate summary of length m and Y be our reference summary of length n, and LCS b the function which returns the length of the longest common subsequence of its arguments. Then we may calculate the score by [4]:

$$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{6}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{7}$$

$$\beta = \frac{P_{lcs}}{R_{lcs}} \tag{8}$$

$$ROUGE - L = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{9}$$

Notice that none of the discussed metrics seem applicable to our proposed algorithm. In general, evaluation of summarization methods requires some form of human judgement. This is difficult when the summarization algorithm targets a corpus such as ours which is sufficiently large such that a human judge could not be expected to read it themself. Similar problems are common throughout the field of text summarization.

4 Conclusion

In this paper, we have attempted to introduce the field of text summarization, especially to a reader with some prior familiarity of information retrieval. We discussed several approaches to extractive summarization, relating these to ideas from information retrieval where relevant. Next, we proposed a novel algorithm addressing the type of problem which may arise in the domain of information retrieval in which text summarization would be relevant. This algorithm was primarily focused on being accessible and understandable. For instance, our topic derivation was performed using CIDR, which is a now dated technique. More modern techniques use latent dirichlet allocation or neural network based representational learning algorithms [2][9]. Additionally, TF*IDF is a useful intermediate representation, but it is not necessarily ideal. Indeed, there has even been research suggesting that binary weighting can produce results which are more stable and reliable [5]. Though, again this is an open problem with no clear answer. More modern representations include word2vec which uses a neural network to create continuous vector representations of sentences which are able to take into account the semantic relationships of words [9]. Text summarization is still a growing field. As it grows, its relevance to other fields such as information retrieval will become increasingly clear. It is already apparent, however, that text summarization offers a unique ability to give order to large quantities of text in a way that information retrieval alone can not.

5 References

- [1] Dipanjan Das and Andre Martins, "A Survey on Automatic Text Summarization," CMU Technical Report. 21 November 2007.
- [2] Mehdi Allahyari et al. "Text Summarization Techniques: A Brief Survey," arXiv:1707.02268, 28 July 2017.
- [3] Niantao Xie et al. "Abstractive Summarization Improved by WordNet-based Extractive Sentences," arXiv:1808.01426, 4 August 2018.
- [4] Chin-Yew Lin, "ROUGE: A Package for Auomatic Evaluation of Summaries," *Text Summarization Branches Out.* ACL.
- [5] Ani Nenkova and Kathleen McKeown, "A Survey of Text Summarization Techniques," *Mining Text Data.* Springer.
- [6] Dragomir R. Radev et al. "A Description of the CIDR System as Used for TDT-2," 1999.
- [7] Rada Mihalcea and Paul Tarau, "TextRank: Bringing Order into Texts," *Empirical Methods in Natural Langauge Processing,* ACL.
- [8] Horacio Saggion and Thierry Poibeau, "Automatic Text Summarization: Past, Present, and Future," *Multilingual Information Extraction and Summarization*, Springer.
- [9] Gaetano Rossiello, Pierpaolo Basile, and Giovanni Semeraro, "Centroid-based Text Summarization through Compositionality of Word Embeddings," *MultiLing@EACL.*