Query-Biased Multi-Document Abstractive Summarisation

Amrit Singhal
(150092)
Junior Undergraduate
Department of Computer Science and Engg.
IIT Kanpur

Akshat Jindal
(150075)
Junior Undergraduate
Department of Computer Science and Engg.
IIT Kanpur

Abstract

In this paper, we introduce a novel pipeline to implement a query-biased multi-document summarisation using abstractive summarisation. Traditional information retrieval systems return a ranked list of whole documents as the answer to a query. However, in many cases, not every part of an entire document is relevant to the query. Thus, it is desirable to retrieve from the set of retrieved documents, in a succinct manner, a summary of only the required information extracted from across all the relevant documents. The approach proposed involves retrieving relevant documents for each query, followed by extracting relevant passages from each document. This is followed by collating all such relevant passages and performing redundancy removal to keep the length of such a collated document sane. Finally, an abstractive summarisation approach is used to generate abstractive single-line summaries for our information need.

1 Introduction

Search-engines have now become increasingly popular, and have become the common entry-ways for people into the world wide web. With the ever-increasing amount of information available on the internet, and the excessively busy schedules of people, individuals do not have time to read the complete document(s) that is retrieved for their query. What they need is a concise and precise summarisation of the information available in that document. The relevance of this document to the query is judged based on this summary only. Even a extremely relevant document may be marked as irrelevant due to poor summary generated. This leads to the growing importance of document summarisation in modern information retrieval systems.

Summarisation in general comes in two flavours, namely Extractive and Abstractive. Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. In contrast, abstractive methods build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might express. Such a summary might include verbal innovations. Research to date has focused primarily on extractive methods, which are appropriate for image collection summarization and video summarization.

Traditional information retrieval systems return a ranked list of whole documents as the answer to a query. However, in many cases, not every part of an entire document is relevant to the query. Thus, it is desirable to retrieve only relevant passages, as opposed to whole documents, which in effect helps to filter out irrelevant information in a long relevant document.This leads to two different types of summarisation techniques to pursue: query-independent and query-dependent document summarisation.

Information Retrieval Course Project (CS657 2018), IITK.

Query independent summarisation is an extensively researched field and great results have been obtained for extractive summarisation in a query independent manner[6]. While results for abstractive summarisation have also been satisfactory, it is an ongoing field of research, with the current state of the art being Sequence-to-Sequence RNN model described by Nallapati et al[3].

However, query dependent summarisation requires extra work. In this paper, we propose a self designed pipeline to achieve query-focused multi-document abstractive summarisation. The basic steps include firstly retrieving relevant documents for each query, followed by extracting relevant passages from each document. This is followed by collating all such relevant passages and performing redundancy removal to keep the length of such a collated document sane. Finally, an abstractive summarisation approach is used to generate abstractive single-line summaries for our information need.

In this manner, we get an abstractive summary for multiple documents together, without actually having to process all of those documents together in a network, which is the conventional method for multi-document abstractive summarisation, and is computationally more expensive. Our method serves to keep the benefits of abstractive summarsation over multiple documents, while actually doing the task over a single one.

2 The Proposed Pipeline

This section describes each step in the proposed pipeline in detail.

2.1 Document Level Retrieval

The first step is to perform a standard retrieval process to get the set of relevant documents for each query. This step was performed using Pylucene's Information Retrieval platform.

2.2 Passage Retrieval

The next step is to extract from each relevant document, passages that contain information relevant to the query. A critical problem in passage retrieval is to extract coherent relevant passages accurately from a document, which we refer to as passage extraction. While much work has been done on passage retrieval, the passage extraction problem has not been seriously studied. Most existing work tends to rely on pre-segmenting documents into fixed-length passages which are unlikely optimal because the length of a relevant passage is presumably highly sensitive to both the query and document. In this paper, we first explore a few segmenting based approaches and then introduce a new LSA based approach for passage extraction.

TextTiling

TextTiling[1] is a technique for subdividing texts into multi-paragraph units that represent passages, or subtopics. In principle, paragraphs should be coherent, self-contained units, complete with topic sentence and summary sentence. In practice, however, paragraph markings are not always used to indicate a change in discussion, but instead can sometimes be invoked just to break up the physical appearance of the text in order to aid reading. TextTiling is used to partition each document, in advance, into a set of multi-paragraph sub-topical segments.
TextTiling makes use of patterns of lexical co-occurrence and distribution. The algorithm has three parts: tokenization into terms and sentence-sized units, determination of a score for each sentence-sized unit, and detection of the subtopic boundaries, which are assumed to occur at the largest valleys in the graph that results from plotting sentence-units against scores. Each "passage" extracted thus is a coherent, and self-contained sub-document in a way, complete with a topic for itself. We will make use of this primary observation.
The authors for TextTiling used three different scoring strategies, based on blocks, vocabulary introductions, and chains. Also, we ran these against the default NLTK TextTiling to get a comparative result. Finally, the vocabulary introduction based scoring was included in the pipeline.
Now that we have chunked each document into sub-topical coherent passages, we perform a secondary retrieval, on the same query, keeping these passages as our documents. Thus, we are able to get the most relevant passages from the entire corpus.

Sentence Scoring Approaches

Three sentence scoring approaches were tried. In each approach, every sentence in the document is given a score depending on the specific algorithm.Then, a few *top* sentences are picked. For each picked sentence, three sentences above and below it are taken as constituting the *passage* this sentence is a part of. These passages constitute the *relevant* passages to be extracted from the document. We now describe the various scoring techniques used.

TfIdf approach

In this approach, considering the document as the *corpus* and each sentences as a *document* in the corpus, the standard TF-IDF score is calculated for each sentence

$$TFIDF(sen, q) = \sum_{t \in sen \text{ and } q} TF(t) * IDF(t)$$

Luhn's Clustering Approach

According to [5], two issues should be considered for a query-biased summarization: relevance and fidelity. The former shows the relevance of the summary with the query, while the latter indicates the correspondence of the summary with the original document. A good summary should keep both high relevance and high fidelity.

- **RELEVANCE:** Based on the belief that the larger the number of query terms in a sentence, the more likely that sentence conveys a significant amount of the information need expressed in the query, we used the following formula to calculate the score for a sentence

  $$Score_{rel}(sen) = RET * n^2/q$$

  where n is the number of query terms in sentence sen, q is the total number of query terms and RET is a measure of the how important is the Relevance score as compared to the Fidelity score. A value of RET=2 has been used in this paper.
- **FIDELITY:** Based on the assumptions that high frequent non-stop words are significant and sentences with dense cluster of significant" words are important, Luhn[`courses.ischool.berkeley.edu/i256/f06/papers/luhn58.pdf`] proposed a keyword-based clustering method to measure the importance of sentences. More specifically, a word is deemed as "significant" if it is not a stop word and its term frequency is larger than a threshold T (T=2 in this paper).Clusters of significant words are created such that significant words are separated by not more than 4 insignificant words within the sentence. If in that way a sentence contains two or more clusters, the one with the highest significance factors is taken as the measure of that sentence. The significance score for a cluster :

  $$Score_{fid}(sen) = SW^2/TW$$

  where SW is the number of significant words in the cluster (both cluster boundaries are significant words), and TW is total number of words in the cluster.

The final score of a sentence is just a sum of the two scores

LSA based approach

The traditional approach to using LSA for document summarisation is not query biased and no query biased LSA approach has been talked about as far as the authors have researched.

In their paper, Steinberger and Jezek[4] applied the singular value decomposition (SVD) to generic text summarization. The process starts with creation of a term by sentences matrix **A**, upon which SVD is applied to obtain U,S,V matrices. To reduce the dimensionality of the output document vectors(rows of V), only the first **K** columns of U and V are taken and only the first **K** rows and columns of S are taken.

The optimum value of K was set such that the singular values do not fall below half the value of the highest singular value. The authors propose the intuition that each of the K dimensions represents a topic and the singular values corresponding to each topic are a measure of the importance of that topic in the document overall. Using this intuition we propose that each sentence will be given two scores, namely LSA score and Relevance score.

- The LSA score : Instead of simply calculating the norm of every k-lengthed sentence vector as done in [4], we take the weighted norm with the weights being supplied by the query k-lengthed vector.This works because higher the weight, i.e the value of the query vector for some topic k, greater focus is given to that topic while choosing sentences.
- Relevance score : We also add a relevance score for each sentence which is nothing but the cosine similarity of the sentence and query vector.

The final score for a sentence is a weighted sum of the above two scores.

2.3 Redundancy Removal And SuperDoc Creation

The next step is to create, what we call a SuperDoc, that should contain all the information required to answer the query, and no(or not much) irrelevant information. We now have all the relevant passages from all the different documents available in the corpus. These passages do satisfy the information constraint we have on out SuperDoc, but multiple passages may still contain the same information about the topic, as there may be similar information available in different documents, or in different sections of the same document, in both of which cases we will get separate semantically similar passages for the query purpose. This will introduce redundancy in the SuperDoc that we aim to create, which is ultimately undesirable in the summary.
So, rather than simply concatenating these passages one after another, we apply a redundancy removal technique first to avoid repetitive information from entering the SuperDoc. The redundancy removal technique we have chosen is based on simple cosine similarity measure, that provides good enough results for our task. There exists some other more involved methods for the task, but the results from the simple cosine similarity measure are not so far from them.
Based on the method used for passage extraction, we may or may not have a ranking among passages. From now on, we assume that we are picking the passages in the ranked order, from most relevant to least relevant, if we have a ranking (if we used TextTiling approach), or else, we pick passages in the order that they were available in the document, going from the most relevant to the least relevant document.

Algorithm:

1. Construct a vector representation for every sentence in every passage. We have used term-frequency based feature vectors, after removing stopwords.
2. For each passage, picking passages in the above mentioned order:
   - For each sentence in the passage:
     - Check the cosine similarity of that sentence, with every sentence that is already added to the SuperDoc
     - If this comes out to be greater than a threshold, then exclude that sentence from the SuperDoc, else concatenate that sentence to the SuperDoc.

At the end of this algorithm, we will get a SuperDoc that will contain sentences, none of which contain the same information. Surely, there may still be some redundancy of information at the passage level, which will require other advanced redundancy removal techniques which can be added here. For now, this SuperDoc is what we will send to our final step for the abstractive summarisation.

2.4 Abstractive Summary Generation

Now, with the SuperDoc in place, the last step in our proposed pipeline is to generate an abstractive summary of the same. For this step, two different models were implemented.

Model 1

The first model tried was an implementation of a *recipe summarization* model on `https://github.com/rtlee9/recipe-summarization`. Although the model seemed promising as it utilizes a sequence to sequence encoder -decoder, which is among the newer techniques in abstractive summarization. Thus, for texts with a 2-3 word summary, like the name of the dish, the model would perform well, but for the CNN news dataset used in the paper, the results were abysmal. We think that

this was primarily due to the vast difference in the nature of the datasets for which it was designed, namely recipe summaries, and what we used it on. Thus, a second model was tried.

Model 2

The second tried was the *pointer-generator* model, proposed by A. See et al.[2]. This model is am improvement on the simple sequence to sequence architecture models, designed to overcome its shortcomings. The model uses a hybrid pointer-generator network that has the ability to *point* to a word from the document, as well to *generate* new words, as and when needed. Also, this model was trained and tested on the CNN model itself, so it will be safe from any dataset driven disparities. The implementation of the model was taken from `https://github.com/abisee/pointer-generator`.
We picked a pre-trained version of the model, trained on the CNN training corpus. The model gave a ROUGE score of 29.53 when tested on the CNN corpus, which is an improvement in ROUGE score over any of its ancestoral models. So, we finally use this model for the abstractive extraction step of our model. The results obained are not grammatically perfect, but can be seen to match the information needed. We still need quite better methods for this step of the pipeline.

We must note that each step of the pipeline is designed to be independent of the earlier steps, and thus one can use better methods for any of the steps in the process. As better and better method come up for abstractve summarisation, those can simply be used as the last step of the pipeline to get better results.

3 DataSet

The dataset used for this project is the CNN news dataset. `https://cs.nyu.edu/%7Ekcho/DMQA/`.This dataset contains the documents from the news articles of CNN. There are approximately 90k documents.Each document has the following structure:

- The document begins with the CNN story.
- Then, 3-4 **highlights**, which are human generated single-line summaries of the preceding article.

Thus, for retrieval purposes :

1. The **corpus** was created by removing the highlights from each document to give us a corpus of about 90k documents.
2. The query set was created in the following manner:
   - All extracted highlights were placed together. Now, Latent Dirichlet Analysis was used to discover 50 topics within this *corpus* of queries.
   - Top 6 words of each topic were picked and were treated as a query.
   - This gave us 50 query topics to query the corpus on.

4 Implementation

The implementation for the entire project can be found on `https://github.com/amr4i/InfoRetProject`. The readme on the repository explains the steps needed to run the code.

5 Future Work

There is a great amount of work which can still be done on this project.

- Firstly, with Liu et al's[2] state-of-the-art encoder-attention-decoder model for generating long summaries, we can generate much longer summaries from the SuperDoc that we created instead of just single-line summaries.
- Secondly, even with our single line summary approach, the *headline* generated can be used as the updated query for the relevant documents, being used for updating the query for later use.This implements a sort of feedback mechanism.

References

[1] Marti A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, March 1997.

[2] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *CoRR*, abs/1801.10198, 2018.

[3] Ramesh Nallapati, Bing Xiang, and Bowen Zhou. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023, 2016.

[4] Josef Steinberger, Karel Jezek, and Yihong Gong. Using latent semantic analysis in text summarization and summary evaluation. 2004.

[5] Changhu Wang, Feng Jing, Lei Zhang, and Hong-Jiang Zhang. Learning query-biased web page summarization. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 555–562, New York, NY, USA, 2007. ACM.

[6] Y. Zhang, M. J. Er, and M. Pratama. Extractive document summarization based on convolutional neural networks. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 918–922, Oct 2016.