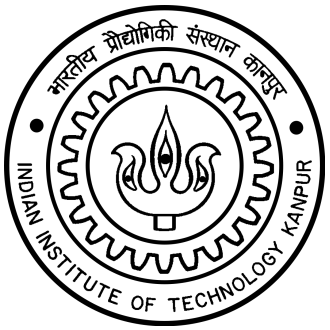

UGP - Report

Topic - Query-Focused Multi Document Abstractive Summarisation

Advisor:
Prof. Arnab Bhattacharya, CSE, IIT Kanpur

Undertaken by:
Akshat Jindal - 150075 - akshatj@cse.iitk.ac.in
Amrit Singhal - 150092 - amrits@cse.iitk.ac.in

Abstract

We present a method to perform query-focused multi-document summarisation, using abstractive techniques. 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. Finally, an abstractive summarisation approach is used to generate abstractive summaries for our information need from the extracted relevant passages of the documents.

Contents

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.

Query independent summarisation is an extensively researched field and great results have been obtained for extractive summarisation in a query independent manner[10]. While results for abstractive summarisation have also been satisfactory, it is an ongoing field of research, with the current state of the art being RL based deep networks by Paulus et al[6].

However, query dependent summarisation requires extra work. We propose a method to get an abstractive summary for a given query together from multiple documents together, without actually having to process all of those documents together simultaneously, 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 substantially lesser data.

2 The Proposed Method

We propose a pipeline to achieve query-focused multi-document abstractive summarisation. The basic steps included are as follows:

1. Retrieve relevant documents for the given query from the entire corpus of documents.
2. Extract relevant passages from each document.
3. Collate all such relevant passages and perform redundancy removal to keep the length of such a collated document reasonable
4. Abstractive summarisation approach is used to generate abstractive summaries for our information need.

This following sections describes each step in the proposed pipeline in detail.

3 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.

4 Relevant Passage Extraction

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.

4.1 TextTiling

TextTiling[3] 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.

4.2 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.

4.2.1 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

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

4.2.2 Luhn's Clustering Approach

According to [9], 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

4.2.3 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[8] 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 [8], 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.

5 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. We use this SuperDoc for our final step of abstractive summarisation.

6 Abstractive Summary Generation

Now, the last step in our proposed pipeline is to generate an abstractive summary of the extracted relevant passages. The first method we apply for the same is to perform a direct abstrative summarisation of the SuperDoc created above. We employ multiple existing methods of abstractive summarisation after performing a extensive literature review of the same, which will go through in the rest of this section.

6.1 Seq-to-seq RNN

6.1.1 Structure of the model

Idea The basic idea behind abstractive summarisation is to map an input sequence of words to an output sequence of words. This basic structure is common across various NLP tasks like Machine Translation. A common deep learning architecture used in such scenarios is the attentional Recurrent Neural Network (RNN) encoder-decoder model proposed in Bahdanau et al[1].The encoder consists of a bidirectional GRU-RNN (Chung et al., 2014), while the decoder consists of a uni-directional GRU-RNN with the same hidden-state size as that of the encoder, and an attention mechanism over the source-hidden states and a soft-max layer over target vocabulary to generate words.

Architecture The tokens of the article w_i are fed one-by-one into the encoder (a single-layer bidirectional LSTM), producing a sequence of encoder hidden states h_i. On each step t, the decoder (a single-layer unidirectional LSTM) receives the word embedding of the previous word (while training, this is the previous word of the reference summary; at test time it is the previous word emitted by the decoder), and has decoder state z_t. The attention distribution α_t is calculated as in Bahdanau et al[1].

The decoder basically outputs the summary, one word at a time. At each time step *t*, it calculates the hidden state z_t as :

$$z_t = f(z_{t-1}, c_t, y_{t-1})$$

where the function f is some non-linear function, and c_t is the context vector at time step *t* where :

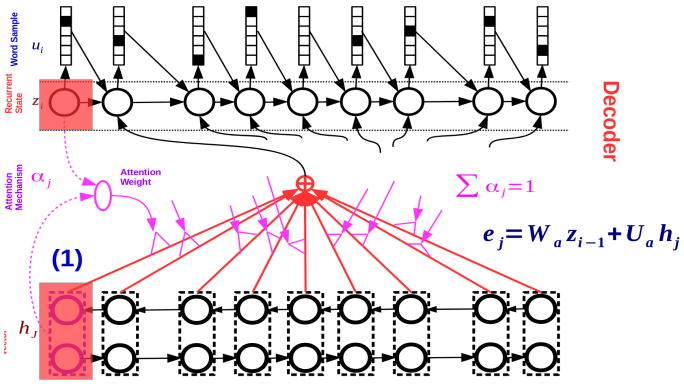

Figure 1: Decoder with attention architecture

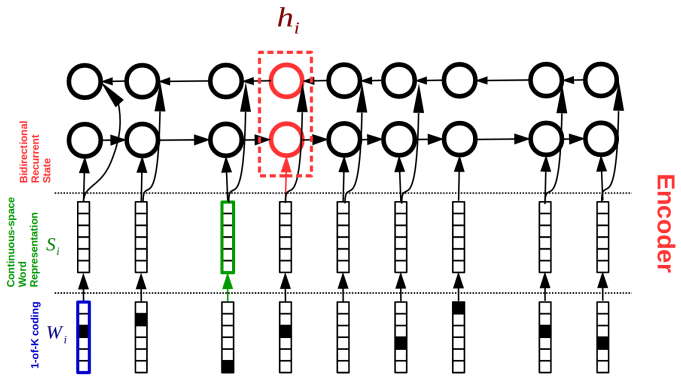

Figure 2: Bi-directional RNN encoder

$$
\begin{aligned}
e_{it} &= a(z_t, h_i)\texttt{OR} \\
e_{it} &= v^T tanh(W_z z_t + W_h h_i + bias) \\
\alpha_{it} &= softmax(e_{it}) \\
c_t &= \sum_i \alpha_{it} * h_i
\end{aligned}
$$

a is a FNN model whose weights are also learnt along with the other parameters of our model. We have also mentioned the exact form used in the paper for completeness.

Realise that the attention distribution can be viewed as a probability distribution over the source words, that tells the decoder where to look to produce the next word. Then, z_t is the passed through a linear layer and then transformed to softmax probabilities via :

$$P_{vocab}(w) = softmax(V * z_t)$$

which gives us the probability distribution over the Vocabulary from which the words are sampled by the decoder for summary generation.

Loss and Training The loss function used is the negative log-likelihood of the target word at each time step :

$$L = -\sum_t \log P(w_t^*)$$

The training is done by calculating the gradients via BPTT and then using SGD to update the weight of the model.

6.1.2 Implementation and Results

Implementation Details An existing implementation of the Seq-to-Seq RNN architecture was taken from: `https://github.com/rtlee9/recipe-summarization`. We adapted it for use with the CNN dataset, and trained it on the same.

Results The model generated a ROUGE score of 26.73 on the CNN test set. But, when the trained model was used to generate summaries from multiple instances of created SuperDocs, the results weren't satisfactory at all. The summaries obtained thus lacked coherent structure and mostly failed to present all the required information as well, presenting some junk instead. This was only checked qualitatively, as a quantitative measure for the same could not be agreed upon.

6.1.3 Problems

The baseline model as pointed out in Nallaptali et al[5] has a lot of problems :

1. The model failed terribly when asked to produce summaries of length longer than a few words.
2. It produced summaries that contain repetitive phrases. Since the summaries in this dataset involve multiple sentences, it is likely that the decoder 'forgets' what part of the document was used in producing earlier highlights.
3. Finally, often-times in summarization, the keywords or named-entities in a test document that are central to the summary may actually be unseen or rare with respect to training data. Since the vocabulary of the decoder is fixed at training time, it cannot emit these unseen words. Instead, a most common way of handling these out-of-vocabulary (OOV) words is to emit an 'UNK' token as a placeholder. However, this does not result in legible summaries.

6.2 Pointer Generator Networks

We first aimed to solve the problem of out-of-vocabulary words by using Pointer Generator Networks proposed by A. See et al.[4]. This model is an 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.

6.2.1 Structure of the model

Mechanism In the pointer-generator model, as shown in Figure 3, the attention distribution α_t and context vector c_t are calculated as is. In addition, the generation probability $p_{gen} \in [0, 1]$ for time step t is calculated from the context vector and the decoder state :

$$p_{gen} = \sigma(W_z z_t + W_c c_t + bias)$$

This p_{gen} is used as a soft switch to choose between generating a word from the vocabulary by sampling from P_{vocab}, or copying a word from the input sequence by sampling from the attention distribution α_t. For each document let the extended vocabulary denote the union of the vocabulary, and all words appearing in the source document. We obtain the following probability distribution over the extended vocabulary :

$$P(w) = p_{gen} P_{vocab} + (1 - p_{gen}) \sum_{i|w_i=w} \alpha_{it}$$

The ability to produce OOV words is one of the primary advantages of pointer-generator model.

Coverage Now, to handle repetition of phrases in the summary See et al[7] proposed a coverage mechanism. This involves, maintaining a coverage vector cov_t which, intuitively, is a (unnormalized) distribution over the source document words that represents the degree of coverage that those words have received from the attention mechanism so far.

$$cov_{it} = \sum_{t'=0}^{t'=t-1} \alpha_{it'}$$

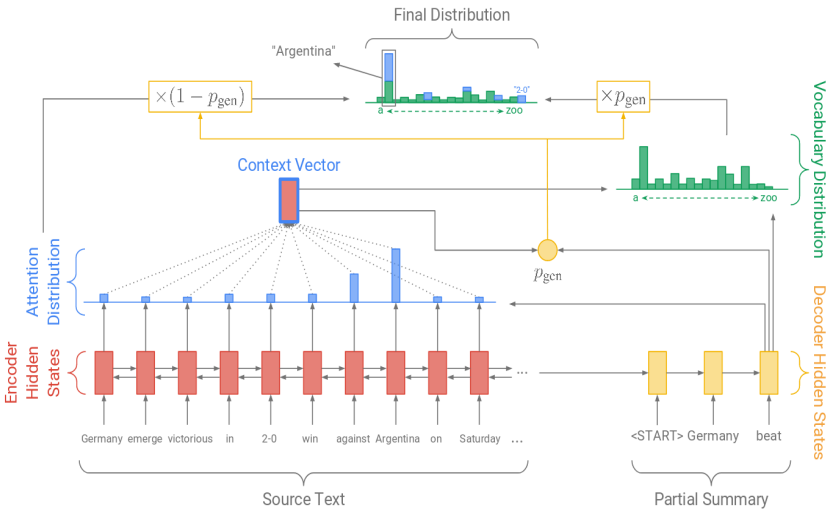

Figure 3: Pointer-Generator Mechanism

This coverage vector is used as extra input to the attention mechanism :

$$e_{it} = v^T tanh(W_z z_t + W_h h_i + W_{cov} cov_{it} + bias)$$

This ensures that the attention mechanism's current decision (choosing where to attend next) is informed by a reminder of its previous decisions (summarized in cov_t. This should make it easier for the attention mechanism to avoid repeatedly attending to the same locations, and thus avoid generating repetitive text.

Loss Function A slight change to the loss function is now needed to penalize the model for attending to the same weights again and again :

$$covloss_t = \sum_i min(\alpha_{it}, cov_{it})$$

$$Loss = \sum_t (-\log P(w_t^*) + \lambda \sum_i min(\alpha_{it}, cov_{it}))$$

6.2.2 Implementation and Results

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. We used this model for the abstractive summarisation step of our model. The results obtained are not grammatically perfect, but can be seen to match the information needed.

6.2.3 Problems

The pointer - generator model although good, does have some problems:

1. First, even though the coverage mechanism is used, repetition is still seen in the summaries. This can be attributed to the fact that the the current word being generated also depends upon what previous words were generated. So Paulus et al[6] used Intra-Attention on Decoder outputs which we'll see in a moment.
2. Secondly, the model, despite of giving a fair chance to both abstractive and extractive aspects of the summary, ends up being heavily biased towards always throwing out an extractive phrase resulting in a more or less extractive summary.

6.3 Reinforcement Learning based techniques

6.3.1 A conundrum

A very big problem with the baseline summarization models is during the sampling stage. During training, we always feed in the correct inputs to the decoder, no matter what the output was at the previous step. So the problem that this gives rises is that the model doesn't learn to recover from its mistakes and assumes that it will be given the golden token at each step in the decoding. This works fine during training but during test time, we sample the next word from the output of the previous step, so if the model produces even one wrong word then the recovery is hard. A naive way to do rectify this problem is to toss a coin with $\mathbb{P}[heads] = p$, during decoding at training time and choose the token produced at previous step with probability p and the golden output token with probability $1 - p$. Now the network can't always assume that it'll be given the correct summary and hence learns to generalize better. In practice, this method gives only slight improvement but impacts convergence.But very recently Paulus et al[6] have come up with a Reinforcement Learning based training method that gave huge improvements.

6.3.2 Reinforcement Based Learning

The training in our baseline model is simply word-level supervised training. The model's aim is to output the reference summary, so we define a cross entropy loss between the target and the produced word. But this approach is fundamentally flawed. There are various ways in which the document can be effectively summarized. The reference summary is just one of those possible ways. Hence,the model's aim shouldn't be just restricted to outputting only the reference summary. There should be some scope for variations in the summary. This is the essential idea behind the Reinforcement Learning based training approach. In this approach, during training, we first let the model generate a summary using its own decoder outputs as inputs. This is essentially sampling as described above. After the model produces its own summary, we evaluate the summary in comparison to the reference summary using the ROUGE metric. We then define a loss based on this score. If the score is high that means the summary is good and hence the loss should be less and vice-versa.

6.3.3 Self-Critical Policy Gradient

For this training algorithm, we produce two separate output sequences at each training iteration: y^s which is obtained by sampling from the $p(y_t^s|y_1^s, \ldots, y_{t-1}^s, x)$ probability distribution at each decoding time step, and $\hat{y}$, the baseline output, obtained by maximizing the output probability distribution at each time step, essentially performing a greedy search. We define $r(y)$ as the reward function for an output sequence y, comparing it with the ground truth sequence y^* with the evaluation metric of our choice. (Here, we take it as ROUGE).

The loss function as can be seen below is designed in such a way, that minimizing L_{rl} is equivalent to maximizing the log-likelihood of the sampled sequence y^s when the reward for y^s is greater than the reward for $\hat{y}$. This, therefore prods the model to earn higher reward for y^s and therefore a better ROUGE score is obtained.

$$L_{rl} = (r(\hat{y}) - r(y^s)) \sum_{t=1}^{n} \log p(y_t^s|y_1^s, \ldots, y_{t-1}^s, x)$$

6.3.4 Augmenting the Loss Function

Simply using L_{rl} as the loss function tends to bias the model towards simply learning to maximise ROUGE scores. Because of the inherent problem with ROUGE, this may lead to summaries that are not human readable. So, to ensure that, we infuse the original teacher-forcing training loss too.We can interpret it as, RL training giving the summary a global sentence/summary level supervision and Supervised training giving a local word level supervision. :

$$Loss = \gamma L_{rl} + (1 - \gamma) L_{tf}$$

6.3.5 Intra-Temporal and Intra-Decoder Attention

The RL based model also had 2 more unique features that allow this model to overcome the problem of repetitive summaries that we faced in earlier models. This it does by not only adding an intra-temporal attention mechanism in the encoder to ensure that the same input word is not attended over too many times, but a novel decoder attention is also added as mentioned in the problem section of Pointer-Generator networks.

Intra-temporal Attention The only change this encoder design has when compared to the Seq-to-Seq RNN model, is that after calculating the raw attention weights e_{ti}, we normalize these weights using the previous attention weights before applying softmax to get α_{it}:

$$e_{it}^{'} = \begin{cases} \exp e_{it} & t = 1 \\ \frac{\exp e_{it}}{\sum_{t'=1}^{t-1} \exp e_{it}} & otherwise \end{cases}$$

Intra-Decoder Attention Another change that Paulus et al[6] came up with was adding an attention mechanism in the decoder.The attention weights are calculated $\forall z_{t'}, t^{'} \leq t-1$ by the standard method and a context vector $c_t^{decoder}$ gets created now. This slightly changes the equation for calculating the deccoder hidden state:

$$z_t = f(z_{t-1}, c_t, c_t^{decoder}, y_{t-1})$$

6.3.6 Implementation and Results

The implementation of the model was taken from `https://github.com/oceanypt/A-DEEP-REINFORCED-MODEL-FOR-ABSTRACTIVE-SUMMARIZATION`

The model was trained on the CNN dataset and obtained a ROUGE-1 score of 37.68 on the results. Clearly, the model outperforms all its predecessors in terms of generating a abstractive summary.

7 Issues with SuperDoc-based summarisation

Although the abstractive summarisation models worked decently for generating summaries of news articles from the CNN dataset, their extension to SuperDoc was not very good. It did extract some relevant details, but was not coherent at all. Also, the abstractive step uptil now only acts to summarise this created SuperDoc, with no importance given to the query at this stage.

Our SuperDoc approach consists of filtering the input documents according to relevance and then pass the filtered relevant passages to an abstractive model. We infer from the grammatical incoherence of the results that this approach cannot adapt well for abstractive methods because the input that is generated by the filtering process is quite different from the type of documents on which the abstractive model was trained: it is not a well structured coherent document. Abstractive models rely critically on the sequential structure of the input to take decision at generation time.

Therefore, we update our proposed method for preserving the document structure while infusing relevance into the abstractive model during decoding.

8 Injecting Query Relevance in Abstractive Summarisation

All abstractive summary approaches discussed above simply summarize a given input document with no regard to the query-bias critical to our pipeline. Adding a query bias to the abstractive summary step is done in the following manner as inspired from Baumel et al[2] :

1. Given a document and a query, we calculate the relevance of each sentence to the query (which we obtain from the initial part of our pipeline) and use this relevance as an additional

input to the network. The relevance model predicts the relevance of a sentence given the query.

2. We project the relevance score of sentences to all the words in the sentence to obtain a word-level relevance score.
3. At each decoding step in the model, we multiply each (unnormalized) attention score of each word calculated by the model by the pre-computed relevance score.

$$e_{it}^{'} = Rel_i * e_{it}$$

where, Rel_i is the relevance score of w_i.

9 The Updated Proposed Method

We use the following simple eager algorithm to produce summaries from multiple documents and control the length of the output.

1. Retrieve relevant documents for the given query from the entire corpus of documents and sort the input documents by their relevance score to the query, as calculated by Pylucene.
2. Perform sentence level scoring techniques, as mentioned in Section 4.2, to obtain sentence level relevance scores and therby obtain word-level relevance scores, as mentioned in section 8.
3. Iteratively summarise the documents, as a whole to maintain the document structure, till the pre-designated budget of the word limit is reached.
4. Perform redundacy removal steps, as mentioned in section 5, while combining these abstractive summaries (Rather than combining the relevant passages).

10 DataSet

The dataset used for this project is the CNN news dataset, which is available at `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. The queries thus generated were highly likely to include multiple documents containing relevant information.

Also, for purposes of abstractive summarisation models:

1. The news story in each article was treated as the text document.
2. The collation of all the highlights was treated as a multiple line summary for that document.
3. For the purposes of getting a smaller summary, only the first highlight was taken as the summary of the document, while the rest of the highlights were discarded.

11 Future Work

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

- We are still working on perfecting the implementation of the added query bias in the RL models. This the immediate work that needs attention.
- **Hierarchical Attention :**Based on the idea that for the summary some sentences are more important than others. So they use two Bi-Direction RNN to scan the source text, one at word level and another at the sentence level. Then the calculate word-level attention using first encoder and sentence level attention using the second encoder. Word level attention is then weighted by corresponding sentence level attention.

$$P_j^a = \frac{P_w^a(j)P_s^a(s(j))}{\sum_{k=1}^{N_d} P_w^a(k)P_s^a(s(k))}$$

Here $P^a(j)$ is the attention given to word j where it's corresponding sentence is $s(j)$. P_w^a is the word level attention and P_s^a is the sentence level attention.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[2] Tal Baumel, Matan Eyal, and Michael Elhadad. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *CoRR*, abs/1801.07704, 2018.

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

[4] 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.

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

[6] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304, 2017.

[7] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368, 2017.

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

[9] 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.

[10] 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.