DUJASE Vol. 6 (2) 39-48, 2021 (July) https://doi.org/10.3329/dujase.v6i2.59217

An Automatic Abstractive Text Summarization System

Nasid Habib Barna and Hasnain Heickal*

Department of Computer Science and Engineering, University of Dhaka, Dhaka, Bangladesh

**E-mail: hasnain@cse.du.ac.bd*

Received on 01 April 2021, Accepted for publication on 06 September 2021

ABSTRACT

Abstractive text summarization is one of the most interesting problems in the research field of Natural Language Processing. Recent advances in sequence to sequence model have made it possible to apply new approaches for abstractive text summarization and perform significantly. But most of the existing systems suffer from some drawbacks like word repetition, producing inaccurate or irrelevant information etc. In this work we propose a novel architecture incorporating advanced word embedding layer and topical feature with a pointer generator network to generate more topic oriented summaries in a logically sequenced way. Adding a word embedding layer with the model can capture semantic features of words in the input sequence more accurately. Also our proposed system with incorporated topical features ensures that the summaries focus on the most important parts of the source document. We applied our model to the CNN/Daily Mail dataset and outperformed the baseline model by all the ROUGE scores.

Keywords: Text summarization, Abstractive text summarization, Word embedding, Topical feature.

1. Introduction

As information is available in abundance for every topic on the internet, it is most important to have an improved mechanism to extract the information quickly and efficiently. Condensing the important information in the form of a summary would benefit a number of users. Manual summarization by humans would become very overwhelming. So the system has to be automatic while giving a short summary of a large document, thus creating an automatic text summarization technique. Extractive text summarization (ETS) systems can generate summaries quite successfully by extracting important sections of the source. But summaries from ETS systems often lack the ability to cover all the semantically relevant aspects of data in an effective way. Here comes the challenge of the abstractive text summarization system (ATS) that needs to generate the summary with novel sentences and represent the summarized information in a human readable form. Another challenge of the ATS system is to preserve the soundness and readability of the created summaries by putting more attention to the semantic features of the source texts. The logical flow of the generated summaries also needs to be maintained.

Automatic text summarization systems have a wide range of applications. When researching documents, summaries make the selection process much easier. It can be a great help for the researchers to identify the most important ideas of any research paper. Also automatic summarizers can assist in writing the abstract of a research work. Google's featured snippets are short selections of text that appear at the top of search results to answer a searcher's query in a short form. This process is done through an extractive approach which sometimes can result in delivering insignificant or irrelevant data. An abstractive approach can help in solving this problem by representing the overall context of the article in a summarized form. In many cases, reviews (of movies, books, products etc.) can be long and descriptive. Automatic text summarization systems can become a great help to bring out the outline of the reviews through a summary.

Some extractive text summarization systems proved to be quite successful in extracting the most relevant parts from a document. But automatic abstractive text summarization is a still developing research area. No system can completely justify its accuracy in representing the gist of a large text document with generated sentences. So, in the process of trying to solve the problem of coping with the ever-growing amount of online data we want to build an abstractive automatic text summarization system. This system is intended to reduce the limitations of existing systems and produce a concise and fluent summary while preserving key information content and overall meaning. Even with a small vocabulary dictionary, this system will be able to handle rare or out-of-vocabulary words. We aim to preserve the soundness and readability of the created summaries by handling semantic features of the source texts. We also intend to ensure maximum topic coverage by incorporating topical features with the system.

Automatic text summarization has always been a major topic of interest in the field of NLP (Natural language Processing). When we humans write a summary, we read the entire document to get an overall understanding and represent that in a short form only highlighting the important points. Computers lack this language capability and thus makes it extremely difficult to automatically summarize text from a large document. Research interest in automatic summarization gained attention in the late 50s (Luhn et al. [1]) and there has been great improvement in this field ever since.

Among the two approaches for automatic summarization, the great majority of past work has been extractive. This is because of the high level of difficulty of abstractive summarization. In this case, the computer has to come up with novel words to write the summary while preserving the meaning of the original input. This requires advanced

natural language techniques. Extractive summarization depends on the extraction of important parts of the source text. So the fundamental task here is to point out the most important sentences from the input document. The research on extractive text summarization began with the method of extracting important sentences using features like word and phrase frequency [1]. This paper proposed to extract the sentences of a document consisting of high-frequency words while ignoring the very high frequency common words. Recently, using neural networks for extractive text summarization has been showing effective results in understanding more context in a source document [2] [3]. The neural models work to combine only the important sentences while maintaining the semantics. Narayan et al. [4] proposed an encoder—decoder architecture based neural model consisting of RNNs and CNNs. They applied single-layer convolutional neural systems over sequences of word embeddings to acquire portrayals of sentences. Then to make groupings out of sentences they used a recurrent neural network.

On the other hand, abstractive summarization methods can be divided into three different approaches. In a structure based approach, important information of the text is assigned to some predefined structure to create abstractive summaries. There are different methods performed to approach this problem that are based on tree, rule, ontology or graphs. For semantic based approach semantic representation of the input texts is created before feeding it to a natural language generation system. The representation is attained by creating semantic graphs or by finding the informative part, argument-predicate structure. This system outputs the ultimate summary by using the noun and verb phrases [5]. In deep learning and neural network based approach, it requires training and learning data in order to extract features from the text. Abstractive summarization has become one of the most important topics for current researchers with the development of deep learning technologies in recent years.

Recent success in sequence to sequence encoder-decoder model [6] has mostly paved the way of abstractive text summarization. Rush et al. [7] first introduced a neural attention sequence to sequence model with an attention based encoder and a neural network language model decoder for text summarization. Chopra et al. [8] further proposed an improvement to this model by replacing the feed-forward neural network language model with a RNN. Systems using these methods still could not reproduce factual details due to uncommon words. To solve this problem, different copy-mechanism approaches have been proposed by researchers [9][10][11].

Song et al. [11] proposed an LSTM-CNN based abstractive text summarization frame-work (ATSDL) that can construct new sentences by exploring semantic phrases. First they performed phrase extraction on the source document and then used deep learning methods for generating summaries. To improve the performance of the text summarizer they combined LSTM and CNN together. They considered both semantics and syntactic structure for summary generation. After extracting phrases the model learns the collocation of them using LSTM. The summary is formed by a sequence of phrases that the model generated after training. To handle out of vocabulary words they used phrase location information.

See et al. [9] proposes an approach to solve these two issues: (i) avoid reproducing factual details inaccurately and (ii) avoid summaries repeating themselves. This model creates a pointer mechanism that allows it to switch between generating text and copying text from source. The model uses a hybrid pointer-generator network that can copy words from the source text via pointing. This method helps to deal with out-of-vocabulary words which also retains the ability to generate new words. The overall mechanism helps to reproduce factual details accurately. They also proposed a coverage to keep track of what has been summarized to discourage repetition. Their model consists of three major parts- (i) baseline sequence to sequence model (ii) pointer generator model to handle rare words and (iii) coverage mechanism to avoid word repetition problems. In the context of multi sentence summaries, their model shows great effective results. Though their pointer module gives considerable accuracy advantage, the overall model cannot attain a higher level of abstraction -- which means it has a tendency to often copy words from the original sentence which is mostly seen in case of extractive text summarization.

We built the model with a sequence to sequence encoder-decoder architecture using recurrent neural networks. Topical feature learning is done by extracting topical words from the source text and integrating this information with the neural network.

Preliminary Concepts

For abstractive summarization, generating new sentences from the original texts while preserving the overall meaning requires advanced NLP techniques to form an entirely new summary. Some terminologies and concepts that are integral part of our proposed system are discussed here for better explanation.

1.1 Sequence to Sequence Modeling

A sequence to sequence model works with sequential information. In the case of text summarization, the input in this model is a long sequence of words and the output is a short sequence of words which is the summary. The architecture of a sequence to sequence model consists of two major components--encoder and decoder.

Both of them are basically two neural network models where different kinds of Recurrent Neural Networks (RNNs) can be used. In the case of text summarization, Gated Recurrent Neural Network (GRU) or Long Short Term Memory (LSTM) have shown better performance.

1) *Encoder:* The encoder is fed the entire input sequence, one word at a time. Each time it first transforms the input sequence into a vector representation and then it

updates its hidden states based on the current word and the previous words that it has already read, using a multi layer neural network. The encoder extracts information from the given input sequence which is forwarded to the decoder in order to help it to create the probability distribution over the next word.

2) *Decoder:* The decoder generates the summary one word at a time. It predicts the next word in the output sequence given the previous word. It creates a probability distribution over the next word and the word with maximum probability is selected. So the encoder here converts overall information from the input texts into a fixed length vector which is used by the decoder to predict the next word. This can only work for short sequences of input texts as memorizing a long sequence into a fixed length vector can be quite hard. The attention mechanism can solve this problem.

1.2 Attention Mechanism

Attention mechanism decides how much attention should be paid to each word while generating the next word at a given time. The decoder increases the importance of specific parts of the input by accessing the intermediate hidden states in the encoder and predicts the next word using all that information.

1.3 Word Embedding

Word embedding is a technique in Natural Language Processing (NLP) which allows words with similar meaning to have a similar but unique representation. Here every word in the dictionary of a language is represented as real-valued vectors in a predefined vector space. Firth et al. [12] first proposed the idea of capturing information from a source based on the relationships among the words. Global Vectors for Word Representation (GloVe) [13] is one of the most significant word embedding systems. It consists of 300 dimensional pretrained vectors that have been trained on Wikipedia and Gigaword. This embedding system is based on matrix factorization of word co-occurrence statistics. It considers the relationships between word pair and word pair rather than word and word. By giving lower weight for highly frequent word pairs, this embedding system prevents the meaningless stop words (eg: “the”, “an” etc.) dominating the training progress. It creates a global co-occurrence matrix by estimating the probability of a given word co-occurring with other words. In order to come up with word vectors, GloVe captures both global statistics and local statistics of a corpus.

2. Proposed System

We are proposing an abstractive text summarization system, a method to generate summary with new sentences from the original texts while preserving the overall meaning. Our method for generating text summary is performed through some major steps:

1. Building a sequence to sequence model
2. Choosing words between vocabulary set and the source document using pointer
3. Ensuring coverage to solve word repetition problem
4. Using a pre-trained word embedding system to preserve semantic similarity.
5. Integrating topical features to generate context aware summary.

2.1 Data Preprocessing

The sequence to sequence model takes the source document by splitting the input texts into words. So as the first preprocessing step, the sentences are tokenized into words based on space. Punctuation marks are also considered as words. Then the tokens are converted to lowercase to get uniformed data. A vocab file is created with all the words with their frequency from the dataset. To point out the start and end of a sentence, unique symbols are added with each sentence. IDs are given to all the words as well as to the unknown tokens, ‘start decoding’ token and ‘stop decoding’ token. This data preprocessing step is very important as using jumbled or confusing words to train the model can create catastrophic results.

2.2 Building Model

Now we have to build our model by training them on the preprocessed data.

2.2.1 Attention based sequence to sequence model

We are aiming to read a document and compress its information to generate a meaningful summary. So to generate an output sequence from a given input sequence. It means we need a sequence to sequence model. This model aims to generate a fixed-length output vector from a fixed-length input vector where their lengths can differ. Here our basic sequence to sequence model has an encoder-decoder architecture with a bidirectional LSTM encoder, a unidirectional LSTM decoder and an attention mechanism to produce a probability distribution of each word in the source document.

2.2.2 Feeding the input sequence through a bidirectional LSTM Encoder

An encoder processes the input sequence to get concise information and create a context vector of a fixed length. This vector represents a conceptual summary of the meaning of the input document. We are using a bidirectional LSTM encoder which consists of a forward LSTM and a backward LSTM and thus becomes very effective in capturing semantic information of previous and following words. First the encoder RNN will read the source document each word at a time. This will produce a sequence of encoder hidden states. The hidden states capture contextual information from the input sequence. The hidden state and cell state of the last time step initializes the decoder.

2.2.3 Generating output sequence using unidirectional LSTM Decoder

The vector from the last timestep of the encoder initializes the decoder which generates a sequence of its own that represents the output. In case of training, the decoder reads

the entire output sequence each word at a time and predicts the same sequence offset by one timestep. For the testing phase the targeted output is unknown to the decoder. Start and end tokens are added to the targeted output sequence so that the decoder knows when to start or end predicting the summary. Once the encoder has read the entire source text, the start token is fed to the decoder RNN and it begins to output a sequence of words. This uses a unidirectional LSTM layer to generate words one by one using the last word it predicted and the hidden state from the last timestep. The decoder creates a probability distribution over the next word and the word with maximum probability is selected. Each time it updates the decoder's hidden state. This process ends when the targeted length of the summary is reached or when the next word with maximum probability is the end token.

2.2.4 Emphasizing specific parts of a long input sequence using attention mechanism

The encoder converts the entire input sequence into a fixed length vector that tries to capture the context from the source document. The decoder is trained to generate a summary solely based on the last hidden state from the encoder. This design might fail in case of long input sequence as it can become very difficult for the encoder to represent the entire information from the source document through a single fixed length vector. To resolve this problem attention mechanisms are introduced. This mechanism creates an attention vector so that instead of using the information of all the words in the input sequence, the importance of specific parts of the input sequence is increased to generate the ultimate output sequence. This attention vector holds the information of how strongly a word is correlated with other words of the source sequence. Here is a simple example in figure 1 to show how this mechanism actually works:

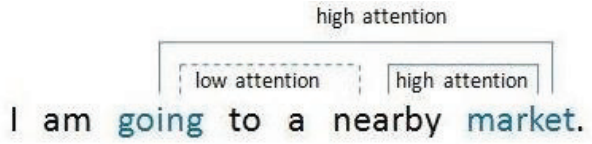

Fig. 1. An example of attention mechanism

The attention distribution or the attention vector is calculated using the hidden states of the encoder and decoder, which is a probability distribution over the words in the source text [14].

$$e_i^t = v^T tanh(W_h h_i + W_s s_t + b_{attn}) \quad (3.1)$$

Where weight vector v, weight matrix W_h, weight matrix W_s and bias term b_{attn} are learnable parameters. Here $W_h h_i$ is the encoder feature, $W_s s_t + b_{attn}$ is the decoder feature and tanh is the activation function with range (-1, 1). Then we pass this probability distribution for each word through a softmax function to get the attention distribution a^t in range 0 to 1.

$$a^t = softmax(e^t) \quad (3.2)$$

The attention distribution a^t_i represents the score of input at position i and output at position t on how well they match. This distribution is used to create the context vector h^*_t -a dynamic representation of what has been covered from the original document for this step. h^*_t is a weighted sum of the encoder hidden states.

$$h_t^* = \sum_i a_i^t h_i \quad (3.3)$$

The context vector and the decoder hidden state are combined and passed through two linear layers of a feed forward neural network to calculate the vocabulary distribution. The output layer consists of a probability distribution over all the words in a large fixed vocabulary. We get the next output using the maximum probability value.

$$P_{vocab} = softmax(V'(V[s_t, h_t^*] + b) + b') \quad (3.4)$$

Here V', V, b and b' are learned by the network while training.

$$P(w) = P_{vocab}(w) \quad (3.5)$$

From P(w) we get the probability to predict the word w. While training, this model uses negative log likelihood of the target word w_t^* for current time step as the loss function.

$$loss_t = -logP(w_t^*) \quad (3.6)$$

Given T as the length of the targeted sequence, the overall loss is calculated.

$$loss = \frac{1}{T}\sum_{t=0}^{T} loss_t \quad (3.7)$$

2.2.5 Adopting pointer-generator network

See et al. [9] proposed a hybrid pointer-generator network which combines the baseline attention based sequence to sequence model and a pointer network to choose words between source document and vocabulary set. As the pointer generator network can copy words from the source via pointing, this system can reproduce factual details more accurately. Suppose the targeted output sequence is "Spain beats Portugal by 3-2." Here 3-2 is treated as one single word which is naturally a rare or out-of-vocabulary term. In this case the pointer generator network is able to copy this word directly from the source retaining the factual information correctly.

So after the previous step we adapted this method [9] and calculated a conditional probability (P_{gen}) to decide whether to generate new words or copy words from the original input document. The calculated value here is a probability scalar between 0 and 1. Based on its value and a given threshold, a novel word is selected from the vocabulary or a word is copied from the source. To calculate P_{gen} for time step t the context vector h_t, the decoder hidden state s_t and the decoder input x_t pass through a linear layer and a sigmoid activation function (range 0 to 1) [9]:

$$P_{gen} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr}) \quad (3.8)$$

where w_h*, w_s, w_x and b_{ptr} are learned by the neural network during training. A word can be generated from the vocabulary by sampling from P_{vocab} or a word can be copied from the source document through sampling from a^t. An extended vocabulary is created by joining the vocabulary set and the words from the source document. Then using P_{gen} for each document, a probability distribution P(w) is calculated over the extended vocabulary. So P(w) is now updated from 3.5 to 3.9

$$P(w) = P_{gen}P_{vocab}(w) + (1 - P_{gen}) \sum_{i:wi=w} a_i^t \quad (3.9)$$

P(w) is the weighted sum of P_{vocab} and a^t. So if w is not in the vocabulary set then $P_{vocab}(w)$ is zero and if w is not present in the source document then $\sum_{i:wi=w} a_i^t$ is zero.

Now the loss function is calculated using the modified P(w):

$$loss_t = -logP(w_t^*) \quad (3.10)$$

$$loss = \frac{1}{T}\sum_{t=0}^{T} loss_t \quad (3.11)$$

2.2.6 Pointer-generator with coverage

Sequence to sequence models often generate repeated words especially when it has to generate multiple sentences. For example, without the knowledge of what has been covered so far the output sequence "Spain beats Portugal by 3-2." can become "Spain beats Spain by 3-2." as both the word 'Spain' and 'Portugal' represent country names.

To solve the problem of repetition, another vector is calculated to keep track of what has been covered in the summary until now. A coverage vector c_t is obtained by adding the attention distributions. Thus it gives us the degree of coverage of the words from the source document. This coverage vector is incorporated with the attention mechanism affecting the attention distribution for current time step [9].

$$c_t = \sum_{t'=0}^{t-1} a^{t'} \quad (3.12)$$

So e^t_i is now updated to

$$e_i^t = v^T tanh(W_h h_i + W_s s_t + W_c c_t + b_{attn}) \quad (3.13)$$

Here W_c is learned by the model during the training process. Calculating loss:

$$covloss_t = \sum_{i}^{X} \min(a_i^t, c_i^t) \quad (3.14)$$

$$loss_t = -logP(w_t^*) + \lambda \sum_i covloss_t \quad (3.15)$$

$$loss = \frac{1}{T}\sum_{t=0}^{T} loss_t \quad (3.16)$$

Here λ is a hyperparameter to reweight coverage loss.

2.2.7 Integrating pre-trained word embedding layer

The simplest form of word embedding could be one where the value of the only dimension of the vector representation of each word is a unique integer, possibly the position of the word in the dictionary. The baseline pointer generator model includes a simple embedding matrix with shape (embedding dimension size × total number of words) for the vocabulary. Here the only information about the words is their indices from the vocabulary set. This simple word embedding makes sure that each word has a unique one dimensional vector representation. But no syntactic, semantic or other contextual information can be extracted using this simple embedding representation. This might result in losing important information. Using a pre-trained word embedding system with the baseline pointer generator network can help to represent the semantic meaning of words more precisely.

We used a pre-trained GloVe embedding model to represent each word as an embedded vector before feeding them to the encoder or decoder. The tensorflow embedding lookup is performed to extract corresponding GloVe representations for every word. GloVe captures both global statistics and local statistics of a corpus in order to come up with word vectors. So incorporating this embedding system on top of Pointer Generator Network helps to maintain the contextual meaning of words in a detailed manner.

2.2.8 Integrating topical words for better topic coverage

The attention mechanism only considers the relationship between original input document and the targeted word. This might fail to get the overall summary of the source document due to less information of the important topics. Here we are introducing some topical vocabulary to get better coverage of the topics that the original document is based on. For example: "Country music, also known as country and western (or simply country), and hillbilly music, is a genre of popular music that originated in the Southern United States in the early 1920s. It takes its roots from genres such as American folk music (especially Appalachian folk and Western music) and blues. Country music often consists of ballads and dance tunes with generally simple forms, folk lyrics, and harmonies mostly accompanied by string instruments such as banjos, electric and acoustic guitars, steel guitars (such as pedal steels and dobros), and fiddles as well as harmonicas. Blues modes have been used extensively throughout its recorded history."

–Source: https://en.wikipedia.org/wiki/Country music

From these texts we can already infer that the main topic here is country music. There could be multiple documents on music but not on country music. So the words 'country music' here gives some additional unique information about the document. So these words here are the topical words. We incorporated the information of topical vocabulary with the attention mechanism and used the updated one to generate our targeted summary.

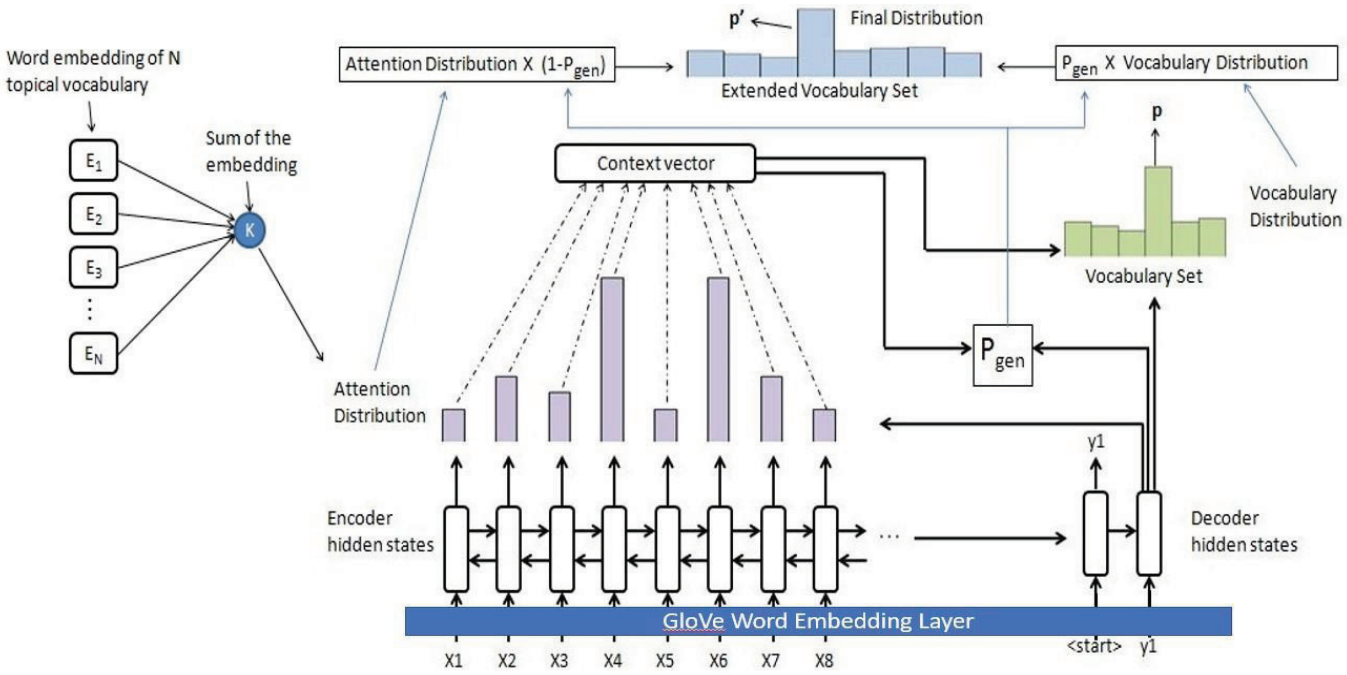

Fig. 2. The pointer-generator model with additional information of topical keywords

2.2.9 Extracting topical vocabulary

There are many text analysis techniques to extract the most important words from a text sequence. Here we are using a statistical approach, TFIDF (term frequency–inverse document frequency) scoring, to extract important words in a document. This approach is fast in measuring how important a word is to a document in a collection of documents as it does not require training data in order to obtain topical words.

The main idea behind this is- if a word occurs in a frequent manner in a document it must be important, so it should get a higher term frequency score. But, if the word occurs too frequently in many other documents, it is not a distinctive identifier for the current document. In that case it should get a lower inverse document frequency score.

Here,

$$TF(x) = \frac{number\ of\ times\ word\ x\ appeared\ in\ a\ doc}{total\ number\ of\ words\ in\ the\ doc} \tag{3.17}$$

$$IDF(x) = log\left(\frac{number\ of\ documents}{number\ of\ doc\ with\ word\ x\ in\ them}\right) \tag{3.18}$$

Now the TF-IDF score is calculated by multiplying these two scores to ultimately assign values to words to identify their importance for a given document.

$$TFIDF(x) = TF(x) \times IDF(x) \tag{3.19}$$

3.2.10 Obtaining topical feature

We have extracted N important words and their word embedding from the original document based on higher TF-IDF score. Let E_i be the word embedding for the i-th topical word. First we calculate the sum of each word embedding of n topical words:

$$K = \sum_{i=1}^{n} E_i \tag{3.20}$$

After calculating their sum we incorporated this information as an input for the traditional attention mechanism. Now the updated formula for the attention distribution becomes-

$$e_i^t = v^T tanh\begin{pmatrix} W_h h_i + W_s s_t + W_c c_t \\ +W_k K + b_{attn} \end{pmatrix} \tag{3.21}$$

Here W_k is a new learnable parameter. This weight matrix is applied to the sum of word embedding of N topical words (K) to get the topic feature $W_k K$ which is added with the traditional attention mechanism.

Figure 2 illustrates the overall process of the pointer generator model with additional topical features to generate summary. Here E_1, E_2 ... E_N are the word embedding of N important keywords of the source document. Their sum K is weighted and added with the attention mechanism as the additional feature. This updated formula for attention distribution results in generating a summary with better insight regarding the source document.

3. Experimental Result and Discussion

To test the proposed text summarization method described in section III we performed extensive experiments. We created four different models based on the additional features of the source texts:

1. Retrained baseline pointer generator model (BPG)

2. Pointer generator model with pre-trained word embedding layer (PGWE)
3. Pointer generator model with integrated topical feature (PGTF)
4. Pointer generator model integrated with both the pretrained word embedding layer and the topical feature (PGWT)

Here the first model (BPG) is a baseline pointer generator model with attention and coverage mechanism. Model 2, 3 and 4 are based on the first model with additional modules. We trained and tested all of these models on the CNN/Daily Mail dataset. This section focuses on the performance analysis of the system which can be divided into several parts.

3.1 Data Collection

To evaluate the performance of the proposed model, we used the CNN/Daily Mail dataset which is a large collection of news articles and modified for summarization. We ran the experiment on 1,00,000 training pairs and 11000 test pairs. These are publicly shared data and are used by many researchers for text summarizer evaluation. We trained all the four of our models on the CNN/Daily Mail dataset. After conducting extensive experiments, we analyzed the output result to understand the efficiency of each of the models. Also we analyzed the data to compare our work with other state-of-the-art text summarization systems. One example of the CNN/Daily Mail dataset is shown in figure 3.

Article (truncated): having been on the receiving end of a 6-1 thumping , a defeat like that could be justifiably met with a backlash by angry supporters . watching a 3-1 first leg aggregate advantage turn into a 7-4 deficit come the end of the reverse encounter too could send many fans *apoplectic* at the capitulation of their side . however that does n't appear the case for those devoted to porto . porto supporters gave their team a hero 's welcome following their 6-1 defeat at bayern munich on tuesday . porto star striker jackson martinez was one of many players to look perplexed by their warm reception . porto boss *julen lopetegui* (left) was hugged by fans congratulating him on their champions league run . police escorts were needed to keep the delirious supporters at bay as the porto team bus drove past . the team bus was met with a cacophony of noise from porto supporters proudly chanting about their club . on their return from a humiliating champions league quarter-final loss at the hands of bayern munich on tuesday night , the squad were given a heroes reception as they arrived back in portugal . in the early hours of wednesday morning , fans mobbed the squad congratulating them on their run in the tournament . star striker jackson martinez and ricardo *quaresma* were one of many porto players who looked perplexed as they were hugged by fans before they making their way on to the team bus - set upon a cacophony of *fiercely-proud* chanting . it was the first time that porto , who had been unbeaten in this season 's tournament up until tuesday night , had reached the quarter-finals of the champions league since the 2008-09 season .

Reference Summary:
bayern munich beat porto 6-1 in their champions league tie on tuesday .
result saw bayern win quarter-final encounter 7-4 on aggregate .
it was the first-time porto had reached that stage since the 2008-09 season .

Fig. 3. One example of the CNN/Daily Mail dataset

3.2 Experimental Setup

The training and testing of the models were conducted in a computer with intel core i7-7700k processor which has 4.2 GHz speed and 32 GB RAM. The operating system was Ubuntu 18.04 LTS. For all the experiments, our model had 150 dimensional hidden states, 400 maximum encoding step size, 100 maximum decoding step size, Batch size of 8, word embedding size 300 and a vocabulary of 50 thousand words. The training was done using Adagrad optimizer with an initial accumulator value of 0.1 and learning rate 0.15. To evaluate our model we installed the pyrouge package to obtain our rouge score.

3.3 Implementation

The major parts of implementation of the experiment are described below:

3.3.1 Training module

During the training and testing phase we limited the length of the input article to 400 tokens and the length of the summary to 100 tokens. These limits were set to exactly match the parameters of the experimental environment in [9], to better evaluate the results. We trained all of our models for about 25000 iterations. Training took almost 26 hours for the 50k vocabulary model.

3.3.2 Summary generation module

In the testing phase the models go through the entire dataset in order and write the generated summaries to specified files. The dataset consists of articles along with their reference summaries. For each article, this module writes these extracted reference summaries and the generated summaries to files in different subfolders in order to run evaluation by the next module.

3.3.3 Performance evaluation module

This module is designed to test and verify the performance and accuracy of the proposed abstractive summarization system. To evaluate the quality of an abstractive summarization system, ROUGE metrics are used as the standard. Because it evaluates how much key information is being preserved between the reference summary and the generated summary. In our paper, we used the Python package pyrouge to run ROUGE evaluation [15]. We evaluated our models with the standard ROUGE metric, reporting the F1 scores for ROUGE-1, ROUGE-2 and ROUGE-L.

ROUGE-1 measures the word-overlap between the reference summary and the generated summary.

For example, if

System Summary: *I went to the nearby shop*

Reference Summary: *I went to the shop*

$$Rouge1_{Recall} = \frac{number\ of\ common\ words}{number\ of\ words\ in\ reference\ summary} = \frac{5}{5}$$

$$Rouge1_{Precision} = \frac{number\ of\ common\ words}{number\ of\ words\ in\ system\ summary} = \frac{5}{6}$$

ROUGE-2 measures the bi-gram overlap.

For example, if

System Summary: *I went to the nearby shop*

Reference Summary: *I went to the shop*

System Summary Bi-grams: *I went, went to, to the, the nearby, nearby shop*

Reference Summary Bi-grams: *I went, went to, to the, the shop*

$$Rouge2_{Recall} = \frac{number\ of\ common\ bi-grams}{number\ of\ bi-grams\ in\ reference\ summary} = \frac{3}{4}$$

$$Rouge2_{Precision} = \frac{number\ of\ common\ bi-grams}{number\ of\ bigrams\ in\ system\ summary} = \frac{3}{5}$$

ROUGE-L measures the longest common sequence.

For example, if

System Summary: *I went to the nearby shop*

Reference Summary: *I went to the shop*

$$RougeL_{Recall} = \frac{number\ of\ longest\ common\ sequence}{number\ of\ words\ in\ reference\ summary} = \frac{5}{5}$$

$$RougeL_{Precision} = \frac{number\ of\ longest\ common\ sequence}{number\ of\ words\ in\ system\ summary} = \frac{5}{6}$$

Now to calculate the F measure of these rouge values,

$$Rouge_{F\_measure} = 2 \times \frac{Precision\ \times recall}{Precision + recall}$$

3.4 Experimental Result

In this section we will present experimental results for all the four different models.

3.4.1 Baseline pointer generator model

We retrained the pointer generator model [9] and after the testing phase we got the Rouge scores shown in table I.

Table 1: Rouge score table for baseline pointer generator model

Rogue Measure	Rouge-1	Rouge-2	Rouge-L
Baseline Pointer Generator Model [9] (retrained)	36.79	13.05	31.22

3.4.2 Pointer generator model with pre-trained word embedding layer

From the pointer generator model with a pre-trained word embedding layer (GloVe) we got the Rouge scores shown in table II.

Table 2: Rouge score table for pointer generator model integrated with pre-trained word embedding layer

Rogue Measure	Rouge-1	Rouge-2	Rouge-L
Pointer Generator Model with GloVe	37.21	15.6	33.29

3.4.3 Pointer Generator Model with Integrated Topical Feature

From the pointer generator model integrated with topical feature we got the Rouge scores shown in table III.

Table 3: Rouge score table for pointer generator model with integrated topical feature

Rogue Measure	Rouge-1	Rouge-2	Rouge-L
Pointer Generator Model with integrated topical feature	37.3	15.86	33.5

3.4.1 Pointer generator model integrated with pre-trained word embedding layer and topical feature

From the pointer generator model integrated with both the pre-trained word embedding layer and the topical feature we got the Rouge scores shown in table IV.

Table 4: Rouge score table for pointer generator model integrated with pre-trained word embedding layer and topical feature.

Rogue Measure	Rouge-1	Rouge-2	Rouge-L
Pointer Generator Model integrated with pre-trained word embedding layer and topical feature	39.01	17.25	35.94

3.5 Performance Analysis

Here is an example of generated summaries by the four different models:

Source Text:

maverick tottenham forward emmanuel adebayor has insisted that he is happy to stay and fight for his place at white hart lane and rejected reports linking him with a move away. taking to twitter on tuesday night, the 31- year-old togo international expressed his gratitude at being able to play in the premier league and labeled the division 'the best in the world.' adebayor joined spurs in 2011 from manchester city, initially on loan before an impressive first season tally of 18 goals, convinced the club to make the switch permanent for 5million the following summer. tottenham striker emmanuel adebayor rides the challenge of liverpool defender dejan lovren. adebayor takes to twitter to pledge his future to tottenham and insists he will fight for his place in the team . adebayor gives an underwhelming performance against league one side sheffield united in the league cup . the striker 's significantly more modest form since then , however , coupled with the meteoric rise of homegrown star harry kane , have left adebayor little more than a bit-part in the spurs machine . this season , he has made just 16 appearances in all competitions , scoring only twice and has been criticised for several underwhelming performances , in particular his showing against league one 's sheffield united in the capital one cup semi-final . yet, the former arsenal , real madrid and manchester city hitman remains adamant that he will see out the remainder of his contract in north london . adebayor 's last premier league goal for spurs came during a 1-2 defeat by newcastle back in october last year. adebayor has made just 10 appearances in all competitions this season and scored only twice. adebayor might be deemed surplus to requirements due to the breakthrough of homegrown star harry kane .

- Source: MailOnline [16]

Reference Summary:

tottenham forward emmanuel adebayor has taken to twitter to insist his immediate future remains at white hart lane . he has one year left to run on his current contract at spurs . adebayor joined spurs in 2011 from manchester city , initially on loan before an impressive first season convinced them to make the move permanent .

unfortunately , adebayor has not managed to rediscover his early form and has been reduced to a bit-part player in mauricio pochettino 's plans .

he has not scored a league goal for spurs since october .

Generated Summaries:

Baseline PGN (retrained):

emmanuel adebayor has insisted that he is happy to stay and fight for his place at white hart lane and rejected reports linking him with a move away .

taking to twitter on tuesday night , the 31-year-old togo international expressed his gratitude at being able to play in the premier league and labeled the division 'the best in the world'

PGN with Glove word embedding layer (PGWE):

emmanuel adebayor says he is happy to stay and fight for his place at white hart lane . tottenham striker emmanuel adebayor rides the challenge of liverpool defender dejan lovren adebayor might be deemed surplus to requirements due to the breakthrough of homegrown star harry kane .

PGN with topical feature (PGTF):

emmanuel adebayor rides the challenge of liverpool defender dejan lovren .

adebayor gives an underwhelming performance against league one side sheffield united in the league cup .

adebayor 's last premier league goal for spurs came during a 1-2 defeat by newcastle back in october last year .

Proposed PGN with GloVe word embedding layer and topical feature (PGWT):

adebayor has insisted that he is happy to stay and fight for his place at white hart lane and rejected reports linking him with a move away.

adebayor 's last premier league goal for spurs came during a 1-2 defeat by newcastle back in october last year .

adebayor might be deemed surplus to requirements due to the breakthrough of homegrown star harry kane .

The rouge scores are calculated by comparing the generated summaries with the reference summaries.

After analyzing the output of the baseline PGN model in the given example it can be seen that it generated some irrelevant or unimportant data. The last sentence in the example for the baseline model is observed to be deviated from the original topic.

The generated summary from the PGWE model captures important parts of the input by handling semantic measures. The GloVe word embedding system tries to find correlation between word pairs and word pairs. So naturally the sentences with 'adebayor' mostly remain in the generated summary.

For the PGTF model, the generated summary seems to be very context aware. The source text is about a sportsman possibly losing his good reputation due to unsatisfactory performance. The model could actually capture the context and generate a precise topic oriented summary.

The proposed PGWT model consists of both the Glove word embedding layer and the integrated topical feature with network. So the summary generated by this model captures both the context and semantic features. As said before, the source text is about a sportsman possibly losing his good reputation due to unsatisfactory performance and the debut of another sports star. The summary generated by this model is observed to give almost all the important information from the source text without creating irrelevant details.

Table V shows our experimental result for all the models with standard ROUGE metric reporting the F measure for Rouge-1, Rouge-2 and Rouge-L.

Table 5: Rouge score table

ROUGE Measure	Baseline PGN	PGWE	PGTF	proposed PGWT
Rouge-1	36.79	37.21	37.3	39.01
Rouge-2	13.05	15.6	15.86	17.25
Rouge-L	31.22	33.29	33.5	35.94

Chart 4 shows that integrating GloVe word embedding representation and topical features has clearly improved the performance of the summarization model. After analyzing the data in table V and chart 4 we can see that our proposed model achieves significantly better ROUGE scores than the baseline. So we can definitely conclude that our model ensures better performance in preserving key information while maintaining a moderate level of abstraction.

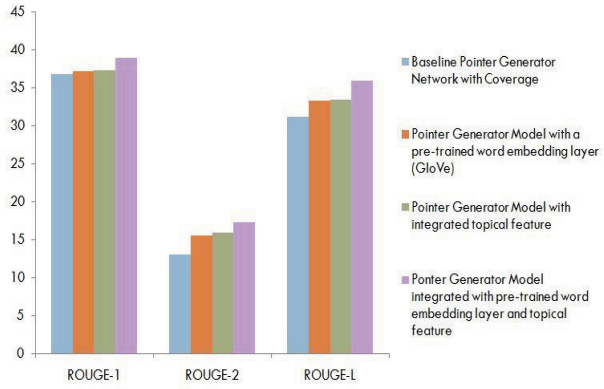

Fig. 4. Comparing Rouge score to evaluate model performance

4. Conclusion

The enlarging growth of the Internet has made enormous amounts of data available. Day by day this is becoming more difficult for humans to extract valuable information from this huge amount of text data due to this abundance of information. Thus the need for an automatic text summarization system cannot be denied. In this book we proposed an automatic abstractive text summarization system using RNN. We feed a sequence of text data into the system and it generates a sequence of text as output in the form of a summary of the source text.

Our system adopted a pointer generator network that helps the system to choose between copying words from the source text and generating novel words using the vocabulary

dictionary. So even if there is a small vocabulary dictionary or too many rare words in the input text, this system can handle the out-of-vocabulary words that ensures accurate reproduction of information. This system can also handle word repetition problems by using a coverage vector to keep track of what has been summarized at each timestep. This method helps to control the flow of the summary and eliminates repetition. We proposed to incorporate a word embedding layer with the model to handle semantic features of the source text. We used Global Vectors for Word Representation systems as the word embedding layer to represent the semantic meaning of words more precisely. Our model successfully incorporated topical features along with the attention mechanism in the pointer generator network. This approach focuses on the most important parts of the source document that ensures improvement in summary's informativeness with better topic coverage.

Our system is trained and tested on the CNN/Daily Mail dataset and after conducting extensive experiments we got a 39.01 F_1 score for Rouge-1, 17.25 F_1 score for Rouge-2 and 35.94 F_1 score for Rouge-L. We retrained the baseline model and got 36.79, 13.05 and 31.22 F1 scores for Rogue-1, Rouge-2 and Rouge-L respectively. We also analyzed the quality of the summaries by human evaluation. Our system outperforms the baseline model in both Rouge measure and qualitative result.

In the future, we want to improve our model to reach a higher level of abstraction while maintaining the accuracy advantage. We also want to work on more advanced methods for extracting topical features from the source. There has been some very recent work on abstractive text summarization using BERT that is showing very promising results. So in future that direction of research might produce better outcomes.

Despite some limitations, our model is already showing encouraging results compared to the state-of-the-art summarization systems and we believe it adds a significant value in the area of text summarization.

References

1. H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 159–165, Apr 1958.
2. Z. Cao, F. Wei, W. Li, and S. Li, "Faithful to the original: Fact aware neural abstractive summarization," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
3. R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
4. S. Narayan, N. Papasarantopoulos, S. B. Cohen, and M. Lapata, "Neural extractive summarization with side information," *arXiv preprint arXiv:1704.04530*, 2017.
5. S. Alshaina, A. John, and A. G. Nath, "Multi-document abstractive summarization based on predicate argument structure," in *2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, pp. 1–6, 2017.
6. I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215
7. A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 379–389. [Online]. Available: https://www.aclweb.org/anthology/D15-1044
8. S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 93–98, 2016.
9. A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *CoRR*, vol. abs/1704.04368, 2017. [Online]. Available: http://arxiv.org/abs/1704.04368
10. R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.
11. S. Song, H. Huang, and T. Ruan, "Abstractive text summarization using lstm-cnn based deep learning," *Multimedia Tools and Applications*, vol. 78, no. 1, pp. 857–875, 2019. [Online]. Available: https://doi.org/10.1007/s11042-018-5749-3
12. J. R. Firth, "A synopsis of linguistic theory 1930-55." vol. 1952-59, pp. 1–32, 1957.
13. J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. [Online]. Available: http://www.aclweb.org/anthology/D14-1162
14. D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014.
15. C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, pp. 74–81, 2004. [Online]. Available: https://www.aclweb.org/anthology/W04-1013
16. [Online]. Available: https://www.dailymail.co.uk/sport/football/article3029323/Emmanuel-Adebayor-vows-contract-Tottenham-striker-rejectsexit-talk.html