Sentient: Abstractive Text Summarization using Sequence-to-Sequence RNNs

Anitha Ranganathan
CISE Department
University of Florida
anitha19r@ufl.edu

Saugat Chetry
CISE Department
University of Florida
saugatpchetry@ufl.edu

Sweta Thapliyal
CISE Department
University of Florida
sthapliyal@ufl.edu

***Abstract*— Text Summarization is a process of summarizing a large text file by taking input as the text file and producing a precis of that file. In automatic text summarization the process is automatic without human intervention or with minimal human intervention. The output of summarization is in human readable and semantically correct text which maintains the original essence of the document. As of now the extractive text summarization is prevalent in industry which is like highlighting important words in the document. There are no out of vocabulary words in the output. These types of model fail to handle the semantics of the document and the key ideas in the document. Abstractive text summarization, on the other hand, is quite complex but very effective. It is like creating a summary of the text in a very similar fashion to humans. We have tried to achieve this in our project for news summarization. The main summarizer is implemented using deep learning and recurrent neural network. At the core of this news summarizer will be attentional encoder decoder neural network. Since we are using an abstractive model the for summarization, our outputs capture the pith of the input text. Due to the inclusion of pointer networks we are able to handle the out of vocabulary words as well and with the help of coverage mechanism the summary generated is free of repetitive statements. We have provided web application which takes the news, summarize it and display it to the user and on click delivers the full article as well.**

Keywords—abstractive models; neural networks; attentional, Coverage mechanism, pointer networks;

1. INTRODUCTION AND MOTIVATION

There two ways to summarize a given text document. First one is to highlight the important words in the document and generate a summary from those words. Basically, we are extracting the summary from the document itself and hence it is called as the extractive text summarization. There is no way to add the new words in this type of models and hence summary generated from the extractive models lacks the semantic essence of the document. The other way to summarize the document is to generate a summary as done by human brain. It not only takes into consideration the actual content of the documents and the words used but try to replace them with new words whenever possible. Also, the summary generated is free of repetitive and redundant information. The proper nouns are never replaced with a new word even though they are out of vocabulary. This is called as the abstractive text summarization.

Abstractive text summarization is the task of generating a headline or a short summary consisting of a few sentences that captures the salient ideas of an article or a passage. We use the adjective 'abstractive' to denote a summary that is not a mere selection of a few existing passages or sentences or key words extracted from the source, but a compressed and salient paraphrasing of the main contents of the document [1]. The best part of an abstractive summary is that it maintains the essence of a document without generating gibberish in the summary.

As we can see that the abstractive summaries are far more superior to the extractive ones, they are complex to generate as well. Generating an abstractive summary is one of the most popular NLP problems that we face today. There is various research done in this domain. Lexrank[2] and Summarization beyond extraction[3] were one of the first paper to be published on the salient summarization. These models are very difficult to design using the conventional machine learning as it requires to come up with the exact mathematical function for each step. With the advent of deep learning, many started addressing the problems faced in NLP using neural networks and recurrent neural networks. Firstly, neural networks were used for machine translation. Later it was realized that machine translation is very similar to the text summarization and various models and modifications were proposed for the same.

The baseline for such modes is sequence to sequence models. The sequence to sequence model with encoder and decoder were quiet famous for machine translation [4]. In this the input sequence of words are fed into the encoder and it then encodes the data. The decoder then takes the encoded data and then translates the data in desired format. Similar model can be used in the abstractive summary generation with the only difference of input and output language being the same in the summarization model. The summarization problem involves taking input as a sequence of words and generating output of sequence of words is called as sequence to sequence models in deep learning. These types of models are quite successful in solving convoluted problems in deep learning such as speech recognition and video captioning.

In this project, we are going to use attentional RNN with encoder and decoder. For encoder and decoder, we are going to use bidirectional LSTM. Our overall system will be based on the cloud and will use freely available news API.

2. RELATED WORK

There has been a lot of work in the past related to extractive summarization, which involves using words from the paraphrase for summarization. However, when humans are made to comprehend something they use their own vocabulary and come up with simpler words for the same task. In the last decade Banko et al. 2000 [5] used a traditional phase table based mechanism, Cohn and Lapta, 2008 used weighted tree-transformations for abstractive summarization. However, it was not until, Rush et al. (2015)[6] that modern neural networks with NLP were used for abstractive text summarization. They used convolutional neural networks, centered on the feed-forward attention mechanism, and improvised the decoder using a recurrent model on the Gigaword and DUC datasets to provide astounding results. There have been several attempts built up on the same work including (Chopra et al., 2016),[6] Abstract Meaning Representations (Takase et al., 2016)[7], hierarchical networks (Nallapati et al., 2016)[8], variational auto-encoders (Miao and Blunsom, 2016) [9], and direct optimization of the performance metric (Ranzato et al., 2016)[10], further improving performance on those datasets.

All the above work has been performed on the corpus involving smaller dataset. The current CNN/Daily Mail dataset however, is a huge collection to work on and can be used for large-scale data summarization. Nallapati et al. (2016) came up with the CNN/Daily Mail dataset, and provided the first abstractive baselines. A comparison could easily be drawn out between hierarchical RNNs, pointer generator networks and feature rich encoders which handle abstractive, combination of abstractive an extractive and extractive summarization respectively. The neural extractive approach (Nallapati et al., 2017) [11], which uses hierarchical RNNs to select sentences, significantly outperforms other abstractive methods when compared with ROUGE metrics.

The pointer network first came to light in (Vinyals et al., 2015). This model combines extractive and abstractive summarizations as discussed in 4.2.B. It is an end-to-end model which solves the problem of rare words and OOV words in the context of machine translation. This sequence-to- sequence model uses Bahdanau et al. (2015) [12] soft attention strategy to produce an output sequence consisting of elements from the input sequence. This work has been extended to various approaches by NMT (Gulcehre et al., 2016)[13], language modeling (Merity et al., 2016), and summarization (Gu et al., 2016; Gulcehre et al., 2016; Miao and Blunsom, 2016; Nallapati et al., 2016 [14]).

The ideology behind this whole implementation is inspired by the essence of approach specified in Forced-Attention Sentence Compression model of Miao and Blunsom (2016)[15] and the CopyNet model of Gu et al. (2016)[16]. Although our design sheds light on this model, we have implemented some major and minor transformations which provides a guaranteed optimized result when tested on ROUGE scores. Some major highlights which we will further discuss in section 4 are shortly described as below: [i]. In their implementation, Gu et al. have used soft-max as their squashing function which as we know produces a normalized output. However, we have tried to avoid the soft-max function and replaced it with an exclusive switching function for probability generation which is described in much detail in the following sections. [ii] Unlike Gu et al., that exploits two disparate distributions as attention distribution and copy distributions respectively, we have tried to recycle the attention distribution to act as the copy distribution function. (iii) Since words can appear intermittently and have multiple occurrences it can so happen that the word might be trivial, yet it could have a higher probability distribution resulting in wrong output sequences provided by the attention mechanism. We, thus sum the probability mass from all parts of attention distribution as compared to that of Miao and Blunsom, who do not reflect this idea. There are additional reasons for doing this modification as mentioned in the subsequent lines: [i] The pointer network models tend to replicate a word when it comes across multiple appearance of the same word in the source sequence provided as an input, this can tremendously affect the performance, [ii] we have also observed that these approaches individually calculate the probabilities of generated words and/or copy words, which adds up to the runtime complexity of the process, eventually affecting the performance of the whole system. Alternatively, if we use an explicit probability generator function we can make modifications to the probability of generated and/or copy words in one go. This further enhances the performance of the system as a whole, [iii] Having two separate distribution does not solve any purpose other than additional calculations and increasing the complexity. Since they have similar properties and functions we have tried to settle down for one distribution as it simplifies the design and solves the desired purpose.

Our approach is built on that of Gulcehre et al. (2016)[13] and Nallapati et al. (2016)[14]. However, it is different from them as we recommend our pointer to be activated whenever needed whereas they train the network to use pointers only for OOV or rare words. Mixing copy and vocabulary distribution, is better for abstractive summarization. This will enable us to reproduce rare but in-vocabulary words, also the mixture model enables the language model and copy mechanism to work together to perform abstractive copying.

Coverage vectors are another important addition to our proposed solution. Coverage was originally used for NMT by Tu et al. (2016)[17] and Mi et al. (2016)[18], they used a GRU to obtain these coverage vectors. For summarization in longer text having such calculations is tedious and memory intensive. Hence, we found a simpler approach Xu et al. (2015)[19], which uses coverage mechanism for image captioning, and the "distraction" coverage like mechanism come to the rescue. We therefore, have adapted quite similar yet a new technique. In order to obtain the coverage vector suffixes, we end up adding the attention distributions as discussed in 4.2.

Attention mechanism can be applied in various techniques. There have approaches like NMT and summarization (Nallapati et al., 2016) [14] who have explored temporal attention as an alternative. These work the previous attention distribution is summed up to attain the current distribution, which effectively dampens repeated attention. Having researched on it we came to a consensus that this method introduces some distortions thereby affecting the attention resulting in a poor performance. We hypothesize that an early intervention method such as coverage is preferable to a post hoc method such as temporal attention – it is better to inform the attention mechanism to help it make better decisions, than to override its decisions

altogether. We performed evaluations using ROUGE scores using temporal attention for the same task (Nallapati et al., 2016) [14] which gave a smaller boost as compared to that of the coverage theory.

3. System Architecture

We propose a system design which can capture the key ideas of current news articles and provide the summary of the latest news to the end user. The current news will be fetched from various news web APIs as well as from the RSS feeds of the leading newspaper. All the incoming data will be preprocessed and cleansed. Preprocessing involves the removal of unwanted links, images, etc.

Once the data is preprocessed it will be fetched to our Abstractive Text Summarization model based on Sequence-to-sequence RNNs [1]. We propose to make this summarizer independent and automated so that it can begin its processing without any manual intervention. This will give us scope to schedule the operation of summarizer based on events, timings, etc. Once the summarizer has completed the task of summarizing the latest news its output will be sent to our application server which in turn will store it in the database. The application server will be responsible for pushing out the summarized content to the end user using various mechanism such as push notification, web API calls etc.

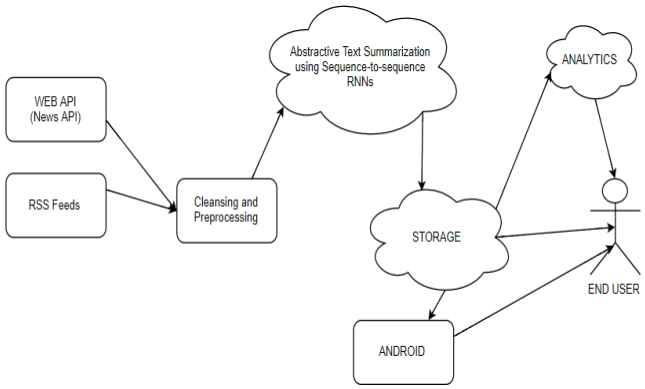

Fig. 1. *Our proposed system design*

4. System Design and Implementation

In this section, we dive deep into our system modules and their interaction in a point to point basis. We are dividing the system into 4 modules namely – 1) Sentient Data Collector and preprocessor, 2) Sentient "the smart" Summarizer, 3) Sentient News Server, 4) Sentient "Smart" Storage.

4.1 Sentient Data Collector and Preprocessor

For the purpose of mode training and testing we would be using the CNN/ Daily Mail dataset having online news articles and their summaries in multiple sentences. This dataset has 781 tokens and the summaries have an average of 3.75 sentences and 56 tokens, consisting of 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs. This can be easily extracted using the scripts provided by Nallapati et al. 2016.

Post training and model deployment, our application will use the latest news articles which we can obtain from the news provides' API such as the CNN/ Daily Mail APIs etc. Sentient Data preprocessor will then cleanse and preprocess the news to remove the non-relevant data like links and advertisements thereby making it usable by "Sentient Summarizer".

4.2 Sentient "The Smart" Summarizer

In the recent past, deep-learning based models that map an input sequence into another output sequence, called sequence-to- sequence models, have been successful in many problems such as machine translation [20], speech recognition [12] and video captioning [21]. This section describes the basic structure of our model. It is divided into three subsections – Sequence-to-sequence unit, pointer generator model, coverage mechanism.

A. Sequence to sequence unit:

Sequence to sequence models are the models which takes an input sequence and returns an output sequence. In our project, the input sequence refers to the original text and output sequence is the summary generated. We can use any RNN for this purpose, however, we choose to feed the article tokens w_i, into a single bidirectional LSTM which acts as our encoder. Like all other feed forward networks, the encoder provides an output encoder hidden state h_i. Since we are using a decoder with attention mechanism as described in Bahdanau et al. (2015) [15], at any time step t, the decoder takes the previous decoder output, the decoder state s_t and the encoder hidden state h_i as input in order to produce the output word embedding. The attention distribution is as below where v, W_h, W_s and b_{attn} *are learnable parameters.*

$$e^t_i = v^T \tanh(W_h h_i + W_s s_t + b_{attn}) \quad (1)$$

$$a^t = \text{softmax}(e^t) \quad (2)$$

As discussed in related work, the attention distribution to obtain context vector is as below:

$$h_t^* = \sum_i a^t_i h_i \quad (3)$$

We have modified the concatenated pairs in the Bahdanau et al to:

$$P_{vocab} = \text{softmax}(V'(V[s_t, h_t^*] + b) + b') \quad (4)$$

During training, the loss for time-step t is the negative log likelihood of the target word w_t^* for that time-step and the overall loss is the weighted average of the individual losses.

B. Pointer generater unit

As discussed in related work we use a pointer generator unit which is well known for its hybrid functionality between extractive and abstractive summarization capabilities. At every time-step at the decoder we use the sigmoid distribution on the context vector, decoded state and the decoder input to obtain a probability generation p_{gen}. As a result of this we allow copy and vocabulary distribution from a fixed vocabulary.

$$p_{gen} = \sigma(w^T_h * h_t^* + w^T_s s^t + w^T_x x_t + b_{ptr}) \quad (5)$$

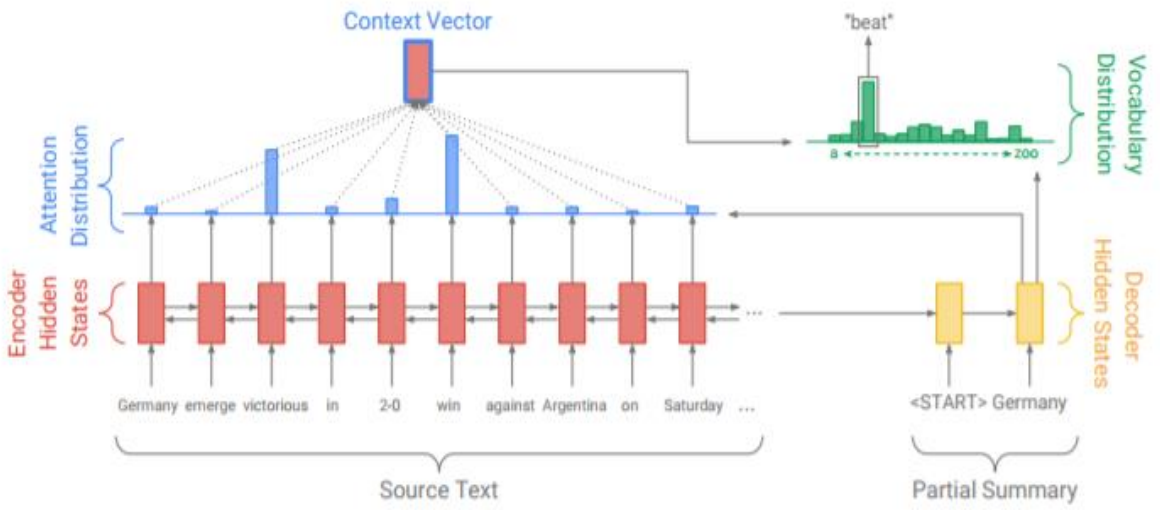

Fig. 2. *Sequence to sequence model with soft attention. Model generates OOV words too. For eg. It scans the iput sequence 'Germany emerge victorious in 2-0 win against Argentina on Sunday' and generates the word 'beat' as part of its abstractive summary by using attention on the word 'victorious'.[26]*

C. *Coverage Mechanism*

To solve the problem of repetition which is present in the sequence-to-sequence model, a coverage vector is maintained. The coverage vector represents the degree of coverage the words in the source document have received so far from the attention mechanism. This coverage vector is fed to the attention mechanism as an additional input which ensures that the current decision of the attention mechanism is based on its previous decision which will in-turn help generating repetitive text. To penalize for repeatedly attending the same location, a coverage loss is defined as:

$$\text{covloss}_t = \sum_i \min(a^t_i, c^t_i) \quad (6)$$

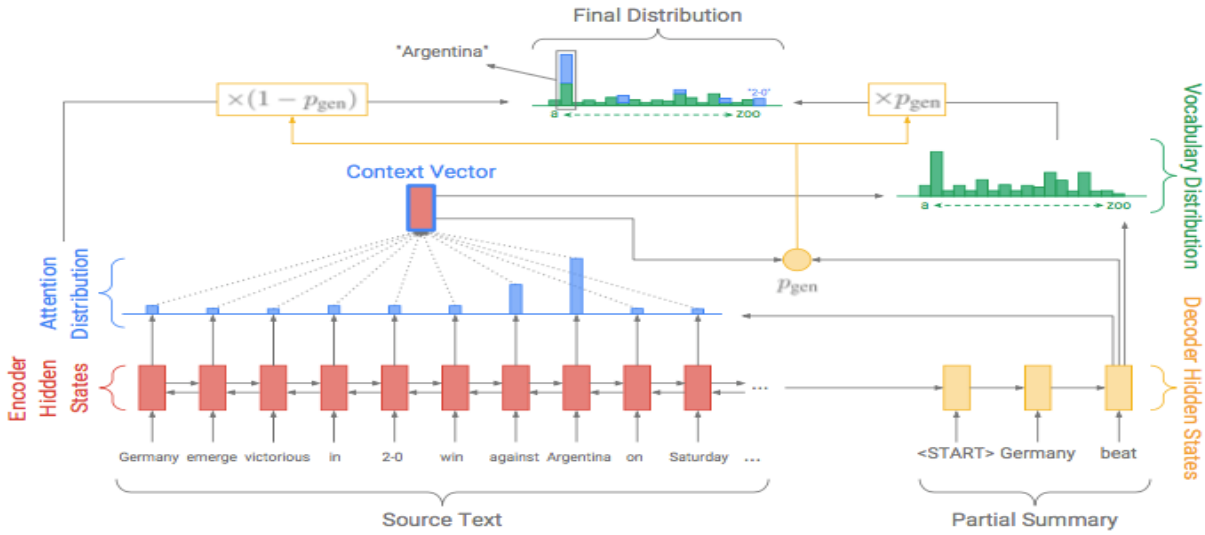

Fig. 3. *In this pointer generator etwork model, each decoder time-step the words are generated using the probability generator function which calculates using the below equations .[26]*

In our final application, we will have the trained model deployed in this module so that our system can provide the summarized news to the Sentient News Server which is discussed in next.

4.3 Sentient News Server

The Sentient News Server in our application will be mainly responsible for providing the summarized news to the client of our application. The Sentient News Server will receive the summarized news from the Sentient "the smart" Summarizer as an input. The input will be received by making a Rest API call to the summarizer module and once the server receives the input, it will save it to the database for future use.

In addition to getting response from the summarizer, the Sentient News Server will also act as the server for all the client applications. Whenever the end user makes a request for latest news, the news server will first check for the validity of the request and then will query the database for the latest news. The result of the query to the database will be made available to the client as a response to its API request.

This design will help us keep the different module loosely coupled which will in-turn facilitate in making out application scalable.

4.4 Sentient "Smart" Storage.

The Sentient "Smart" Storage is the fourth module of our

application. In this module, we will be saving all the summarized news which the Sentient News Server will receive from the Sentient Summarizer. Storing the summarized news is important to make the news available to the end user whenever required. Also, saving the summary will allow us to use this to create new dataset which can be used to train the summarizer further.

Further, the stored news can be used when the user request for older news and also during the time when latest news summaries may not be available due to system downtime or system upgrades.

Our system involves four main classes as stated in section 4.

4.5 Class Diagram

The class diagram gives a brief idea about the properties and members of these four classes and highlights how these classes interact with each other.

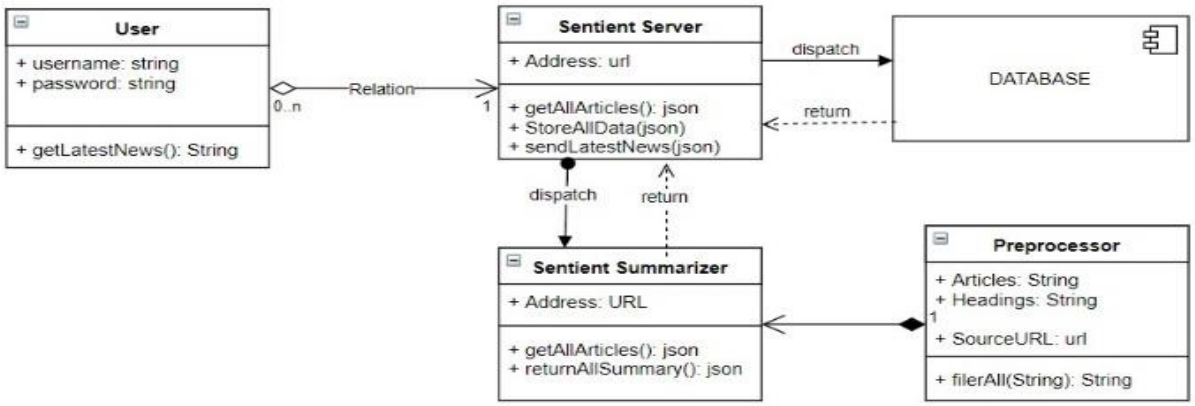

Fig. 4. *Class diagram for the proposed system.*

4.6 Interaction Diagram

Below we have shown the Interaction diagram for Sentient System. The Sentient server will request the text summarizer to provide summaries of the latest happenings across the world. The Sentient Smart text summarizer will begin the background process – (1) query the news providing service APIs such as CNN, Daily Mail, BBC etc., (2) the response of these services will then be passed on to our preprocessing module of the summarizer, (3) the Sentient data preprocessor will cleanse the data as described in section 4.1, (4) this preprocessed data will then be queried onto the trained model "Sentient 'The Smart' Summarizer", which in turn would return the summarized output sequence, (5) this output from the Sentient summarizer will be redirected to the Sentient server, (6) the Sentient server would then will be stored in the Sentient smart storage, (7) On receiving request for summaries from the user, the Sentient server will query this storage for summaries and return the same to the user. We propose to make our system's interaction via Rest APIs. Each module in the system will exist as independent and standalone system and all interaction will be done over HTTP calls. This architecture will make the entire application loosely coupled and hence will lead to increase scalability. Also, this architecture will facilitate further improvement by addition of additional (or replacement) of modules.

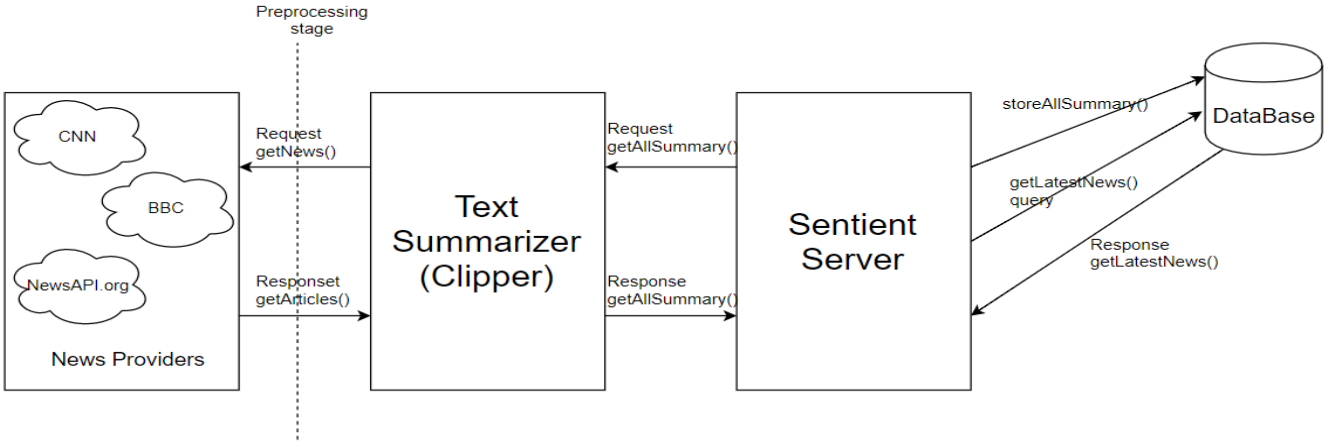

Fig. 5. *System Interation between different modules*

5. PERFORMACE EVALUATION

A. *Experimental Setting:*

For the project we are using 256 hidden states and word embeddings of 128. The vocabulary used is 50K one and not the 150K. The main reason behind this is that the pointer network handles the out of vocab words easily and hence we don't need a more elaborate vocabulary. This saved us more time for training in comparison with the baseline model used in Bhadnau et.al. [12]. The word embeddings are also learned in this code and they are not pre-trained. Hence the training process takes much longer time. The word embeddings are learned using Adagard. To stop the training early we used loss.

The dataset we are using is the CNN and Daily mail data set. The preprocessing of any text article involves tokenizing the data. This means that the articles are fed token by token and all the unwanted characters are removed, and the words are smaller case. This data is then converted into the binary files which is then fed in to the summarizer. The system requirements to execute the summarizer are tensorFlow1.2.1, python 3.6 and pyRouge for evaluation. Although we have used the evaluation code for the purpose of our testing of model and we won't we are using it in the web application as we tend to only summarize the data and not to show the actual evaluation parameters to the user. The code for evaluation is also provided. For the purpose of our project we take input articles till 400 words and train it is using a summary of 4 lines. For testing also, the 4-line summary is generated which is then tested on the pyRouge score. The batch size is 16 whereas the beam size is 4. The base model was trained for 30 epochs and same for the pointer network. For the coverage mechanism the lambda was kept as 1 and further 3K iterations were used to train the model.

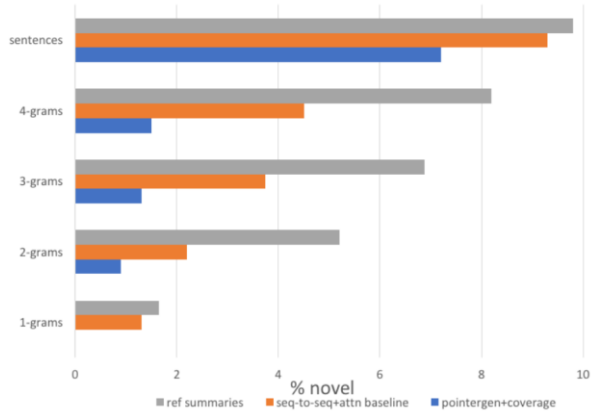

Fig. 6. *Percentage of the novel n grams used in the generated summaries compared*

B. *Evaluation metrics:*

The models are tested with the standard rouge scores. ROUGE stands for Recall Oriented UnderStudy for Gisting Evaluation. It is used to measure the accuracy between the summary generated by the machine and the actual summary generated by the humans. ROUGE metric basically counts the number of n grams that are overlapping in the two summaries to be evaluated. The ROUGE-1 stands for one gram matches and ROUGE-2 for 2 and so on. Rouge L stands for longest common subsequence between the target and the actual summary. The formula to calculate the ROUGE scores is given below

$$\frac{\sum_r \sum_s \text{match}(\text{gram}_{s,c})}{\sum_r \sum_s \text{count}(\text{gram}_s)} \quad (7)$$

Here the numerator counts all the matching n-grams in the target and the reference summary. The denominator counts all the possible n grams in the reference summary. So, we can say that the ROUGE score measures the recall between the n grams that are generated correctly and all the possible n grams in the reference summary.

The rouge scores are obtain using the pyrouge package. The coverage mechanism helps eliminate us the redundancies. We measure repetitions for n grams collectively. The base line model i.e. the encoder decoder scheme with the attention mechanism performs poorly according to the rouge scores. Even by increasing the vocabulary size the performance didn't improve. Although we haven't tested all the related models ourselves as this was time consuming, we had the results from various papers which we compared against over model. We observed that without the pointer generator networks i.e. the model used in Bhadnau et al[12], the rare words were not handled properly and most of the times the summary was redundant gibberish.

Article: cairo (cnn) at least 12 people were killed sunday , and more injured , in separate attacks on a police station , a checkpoint and along a highway in egypt 's northern sinai , authorities said . six people , including one civilian , were killed when a car bomb exploded near the police station in al-arish , capital of north sinai , health ministry spokesman hossam abdel-ghafar told ahram online . he said 40 people were injured . ansar beit al-maqdis , an isis affiliate , claimed responsibility for the attack , which came hours after another operation that the group also claimed . in that earlier attack , a first lieutenant , a sergeant and four conscripts were killed when their armored vehicle was attacked on the highway from al-arish to sheikh zuweid in northern sinai , the military said . two other soldiers were injured and taken to a military hospital . ansar beit al-maqdis has claimed many attacks against the army and police in sinai . a third attack sunday on a checkpoint in rafah left three security personnel injured , after unknown assailants opened fire at them , according to state media . the attacks come as the military announced a reshuffle of several senior military positions , state media reported . among those being replaced are the generals in charge of military intelligence and egypt 's second field army , which is spearheading the battle against the insurgents in the northern sinai . egypt 's army has been fighting a decade-long militant islamist insurgency , which has spiked since the ouster of muslim brotherhood president mohamed morsy in the summer of 2013 . hundreds of police and soldiers , as well as civilians , have been killed in militant attacks in the past months . ian lee reported from cairo . anas hamdan reported from atlanta .

Reference Summary: six people , including one civilian , are killed when a car bomb explodes near a police station . six others are killed when their armored vehicle is attacked on a highway in northern sinai . ansar beit al-maqdis , an isis affiliate , claims responsibility .

Pointer-Generator, With Coverage: six people , including one civilian , were killed when a car bomb explodes near the police station . ansar beit al-maqdis , an isis affiliate , claimed responsibility for the attack . egypt 's army has been fighting a decade-long militant islamist insurgency .

Fig. 7. *A sample output – the summary is abstractive and is not repetetive like some other models*

The pointer generator model has better rouge score. Training was done in less number of epochs. We can see in the summary generated that the out of vocab words is handled in a better fashion. All the facts are copied instead of generating nonsense. When the model is used with the coverage mechanism the rouge scores are even better. Even though we used on 3k iterations to train the coverage the redundancy has been significantly reduced in our results. The ROUGE Score has been better from Nallapati et al[14].

The problem with comparing the extractive the rouge scores of the baseline models is that these extractive models tend to have better scores than the abstractive ones for news summarization since the news articles are structured which has the most important information is at first.

C. *Experimental Results:*

Our key observation of the target summaries was that even when the target summary differs from the reference summary, the original meaning of the summary remains quite relevant to the article. This problem arises when there is only one reference summary. This is not captured in the rouge scores. Also, sometimes model may give a summary that contains words similar to the reference summary but overall is absurd. These are the common weakness of the abstractive models.

The results for the other standard summarizer and the model we are using is given in the table. The table shows rouge-1, rouge-2 and Rouge-L scores for the baseline model Nallapati et al [14], the base line attention model with encoder and decoder, pointer generator and pointer generator network with coverage mechanism[26]. We found out that the pointer generator network along with coverage mechanism performs the best. When we compare it with the extractive model, the baseline lead 3 extractive model performs far superior to the pointer generator network with coverage. As earlier discussed this is due to the structured format of the newspaper articles. Another reason that why extractive models are performing better with ROUGE. The pointer network model is an abstractive model and the abstraction gives us more choices of phrasing and more chances

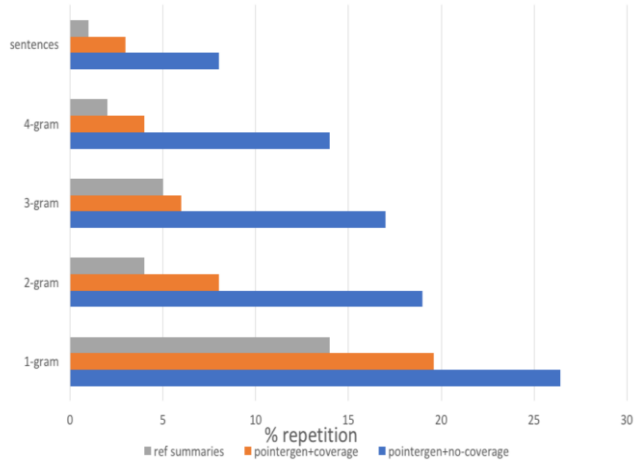

Fig. 8. *Percentage of the duplicate: comparison between all models*

of the generated summary not matching the reference summary. This problem increases here since we have only one reference summary and there can be many abstractive summaries which can be correct but not match the reference summary due to paraphrasing.

In figure 6 we can see the novel n grams produced in the summary generated by the two models we compared i.e. pointer generator with coverage and sequence to sequence base line model. We can see that though the best model is abstractive, it does not produce new n grams as expected by the reference summary. We can see that the base line model produces more n grams that are new. However, this does not signify that the level of abstraction is low in this model. Most of the new n grams produced by the baseline model are erroneous and mostly have all the incorrectly copied words from the original text. Due to the implementation of pointer networks the n grams generated in our summaries are mostly copying the out of vocabulary words from the source text.

In figure 8 we can see that even by dedicating only one percent of the training time on coverage mechanism, we have almost removed all redundancies from the generated summary. We can also observe that as we consider the longer n grams from the summary, better and better our model performs. We can also see that the non-coverage model consists of may duplicate n grams which are undesirable in our summary.

	ROUGE 1	ROUGE 2	ROUGE L
Abstractive Model	35.4	13.1	33.1
Base line model	31.3	11.7	28.0
Pointer-generator	36.2	15.5	32.1
Pointer-generator + coverage	39.1	17.1	36.2

Table 1: Rouge scores of the various summarization techniques

Fig. 9. *The actual snapshot of the output*

6. CURRENT PROGRESS AND PROJECT MANAGEMENT

This section highlights the project progress and work allocation. Its divided into sections to provide some clear and transparent

idea about the work done so far. In this section, the progress of the project and the milestones achieved are explained

6.1 Current Status

Currently we have trained our model that can summarize a given text. The training has been done both on a local machine (using a smaller dataset) and also on AWS Cloud Platform (P2 instance with GPU support). We have also used the trained model in our end application which is a website that give the headlines and summarized news to the user. Whenever the user clicks on a particular news item, the complete story is shown to the user. The website is right now hosted locally but we intend to host it on hosting services like Firebase, Heroku etc. as soon as we are done. We also intend to build a mobile application that can consume the summarized news given we have enough time.

Milestones :

(1) Complete the basic training for the model: Hit (completed)
(2) Tweak and twist learnable parameters and try obtaining a better result: Hit(Completed)
(3) Perform evaluations and obtain experimental results after finalizing the model and its parameters: Hit (Completed)
(4) Deploy the model in cloud and in the server. Perform testing for the application as a whole: 90% Completed (deploying to hosting services is left)
(5) Complete documentation for the final presentation and demo day: 90% completed (only poster is left to be made).

6.2 Team Coordination

In this section, each team member's role and responsibilities have been listed.

We have worked as a team of three where each one of use was actively involved in all the
different aspects of the project. We define each individual's contributions below:

(1) Anitha Ranganthan: She was responsible for training and testing the model for the summarization. This involved implementing the summarization algorithm as well as setting up the environment (both local as well as cloud) to run the training and testing of the code.
(2) Saugat P Chetry: He was responsible for developing the web application and the entire framework that can use the trained models to summarize news content. Creating the backend server that can serve the summarized news as restful response was done by him. Also, he was involved in the implementation and training of the summarization algorithm.
(3) Sweta Thapliyal: She was involved in the training and testing of the model for the summarization. She worked with Anitha during the environment setup and running and testing of the code for summarization. She was also involved in cleansing and pre-processing of the training as well as the test data.

Apart from the aforementioned tasks, each one of us was responsible for paper reviews and individual documentation.

6.3 Milestones, Weekly Plans, and Deliverables

There are approximately five weeks remaining for the final demo and we have split the plan into five short plans:
Week 1: Complete the basic training for the model completed so far.
Week 2: Tweak and twist learnable parameters mentioned in section 5.1 and try obtaining a better result.
Week 3: Perform evaluations and obtain experimental results after finalizing the model and its parameters.
Week 4: Deploy the model in cloud and in the server. Perform testing for the application as a whole.
Week 5: Complete documentation for the final presentation and demo day. We have been able to meet all the miles-stones and check-points which we have set for ourselves during the course of this project. We are right now left with deploying the web-application to a hosting service like Firebase or Heroku.

The github link is https://github.com/Shweta5201/TextSummarizer.git

REFERENCES

[1] Ramesh Nallapati, Bowen Zhou, C¸ a glar Gulc¸ehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond.

[2] Günes Erkan , Dragomir R. Radev, LexRank: graph-based lexical centrality as salience in text summarization, Journal of Artificial Intelligence Research, v.22 n.1, p.457-479, July 2004

[3] Kevin Knight , Daniel Marcu, Summarization beyond sentence extraction: a probabilistic approach to sentence compression, Artificial Intelligence, v.139 n.1, p.91-107, July 2002

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

[5] Banko, Michele and Mittal, Vibhu O and Witbrock, Michael J, Headline generation based on statistical translation

[6] Chopra, Sumit and Auli, Michael and Rush, Alexander M, Abstractive sentence summarization with attentive recurrent neural networks

[7] Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hi- rao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Empirical Methods in Natural Language Process- ing*.

[8] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, C ¸aglar Gulc ¸ehre, and Bing Xiang. 2016. Abstrac- tive text summarization using sequence-to-sequence RNNs and beyond. In *Computational Natural Lan- guage Learning*.

[9] Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sen- tence compression. In *Empirical Methods in Natu- ral Language Processing*.

[10] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level train- ing with recurrent neural networks. In *International Conference on Learning Representations*.

[11] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Association for the Advancement of Artificial Intelligence*.

[12] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2015. End-to-end attentionbased large vocabulary speech recognition. CoRR, abs/1508.04395.

[13] Caglar Gulcehre,Sungjin Ahn,Ramesh Nallapati,Bowen Zhou,Yoshua Bengio. 2016. Pointing the Unknown Words

[14] Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence rnns for text summarization. ICLR workshop, abs/1602.06023.

[15] Miao, Yishu, and Phil Blunsom. "Language as a latent variable: Discrete generative models for sentence compression." *arXiv preprint arXiv:1609.07317* (2016).

[16] Gu, Jiatao, et al. "Incorporating copying mechanism in sequence-to-sequence learning." *arXiv preprint arXiv:1603.06393* (2016).

[17] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, Hang Li. 2016.Modeling Coverage for Neural Machine Translation

[18] Mi, Huaiyu, et al. "PANTHER version 10: expanded protein families and functions, and analysis tools." *Nucleic acids research* 44.D1 (2016): D336-D342.

[19] Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." *International Conference on Machine Learning*. 2015.

[20] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning.

[21] Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence - video to text. CoRR, abs/1505.00487.

[22]] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555.

[23] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. CoRR, abs/1412.2007..

[24] Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In HLT-NAAC

[25] Abigail See, Peter J. Liu ,Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks

[26] Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hi- rao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Empirical Methods in Natural Language Process- ing*.