Abstractive Text Summarization using Text-Rank and LSTM

Paras Avkirkar*, Ruchit Modi†, Mohit Gurnani‡
Department of Computer Science, Stony Brook University
*pavkirkar@cs.stonybrook.edu (112685014), † rjmodi@cs.stonybrook.edu (112685342),
‡ mgurnani@cs.stonybrook.edu (112687997)

***Abstract*—There has been tremendous research in the area of text summarization by using extractive approach. Recently, it has shifted to text abstractions using neural models to replicate human paraphrasing capabilities. This has been accomplished by using TOPIARY, phrase-table based machine translation approaches and deep learning models, mainly RNN and CNN. We have taken a paper Nallapati R. (2016) [1] which uses Attentional Encoder - Decoder Recurrent Neural Networks for text abstraction. It identifies drawbacks in existing literature, and proposes some improvements such as 1) using Large Vocabulary Trick (LVT) in Encoder- Decoder Attention based RNN, 2) Capturing keywords using feature-rich encoder such as Parts of speech tags, TF-IDF, Named-Entity tags, etc. and 3) Capturing Hierarchical Document Structure with Hierarchical Attention 4) Pointer Generation for direct substitution of input word. We use this as our baseline model and enhance it by following proposed approaches: 1) Using TextRank algorithm to obtain scores of all the words which will be concatenated with Glove embedding during Encoding, 2) Using Beam search in decoder to compute top k words to be passed to LSTM decoder cell.**

I. Introduction

Text summarization signifies generating short summary that captures the gist of the article. For this, a simple way would be extracting the most significant sentences from the input document and generating a summary using these sentences called Extractive Text Summarization. A better approach would be Abstractive Text Summarization - paraphrasing the contents into a short summary by ensuring appropriate grammar usage involving use of vocabulary that is not present in the input text. A basic idea to model this would be, mapping an input sequence of words from source to target sequence of words called summary. For this, many sequence to sequence deep learning models have been used successfully in areas such as Machine Translation and Speech Recognition. One such model is the attentional Recurrent Neural Network (RNN) encoder decoder model proposed in Bahdanau et al. (2014), which has produced state-of-the-art performance in Machine Translation. Abstractive text summarization is similar to Machine Translation except the fact that, length of words in decoder is much less than that in encoder (as its lossy conversion), while in Machine Translation, length of words is more or less same in both encoder and decoder (as its lossless conversion). We plan to start with the model described in Nallapati R. et al. (2016) [1] and identify its limitations. Later on, we will try to overcome these limitations using various approaches. Text Rank algorithm is one such algorithm that we plan to use to get **the ranking weights for each word with the hypothesis that it would highlight the most important words from the input that could be used for summarization**.

II. Dataset

Nallapati R. et al. (2016) [1] have tested their techniques over standard Gigaword and DUC datasets, where training each epoch required 10 hours. Training on this data would require days, hence we decided to go with a subsample of dataset from Gigaword and CNN/DM training dataset [2]. The dataset consists of short one and two sentence articles along with the title of the articles. We will use the article as the input document and the title of the article as the target summary.

III. Baseline Model

Nallapati R. et al. (2016) [1] would serve as our baseline model on the dataset chosen by us, and it includes the following key components,

1) Encoder made of bidirectional LSTM while Decoder consist of unidirectional LSTM with the same hidden state size, an attention based mechanism over these hidden states, then applying softmax layer on target vocabulary to generate words.
2) Use large vocab trick (LVT) where the decoder-vocabulary of each batch is restricted to words in the source documents of that batch, the most frequent words in the target dictionary are added

until the vocabulary reaches a fixed size. This technique helps to reduce the size of the softmax layer computation of the decoder and speeds up convergence by focusing the modeling effort only on the words that are essential to a given example.

3) To identify key concepts and key features, we will use parts of speech tags, named-entity tags, TF and IDF tags along with word embeddings.
4) What if we have rare words not trained by Encoder, such words would be generally replaced by UNK token in decoder which would make text summary less legit. Rather, Nallapati R. et al. (2016) [1] comes up with Switching Generator-Pointer technique, where decoder would generate summary for known words from its vocabulary, but if it encounters a rare word, it will switch from generation and point to the word position in the source and copy that directly into summary.
5) In documents having many sentences, it is important to identify important key sentences. In the encoder, bidirectional LSTM aims to capture both important words and important sentences by using attention. Here, word level attention is again re-weighted by corresponding sentence level attention. This rescaled attention goes as input to the hidden state of the decoder. Additional positional embeddings is concatenated to the hidden state of the sentence-level LSTM to model positional importance of sentences in the document

We will be reusing github repo [7] which has implemented components (1) and (3). Hence, these components would serve as our baseline and we propose below additional components to enhance the summary.

IV. Proposed Model

Since the baseline model has no information regarding important words from the document, we propose to use the TextRank algorithm to identify important words and use it as an additional feature to the model input. We implement the TextRank algorithm to precompute scores over all the words in training data and concatenated the normalized scores as a feature along with the input embeddings. We also propose to replace the basic decoder with a Beam Search decoder to improve the summaries. Finally we fine-tune the model parameters to enhance the accuracy.

A. *Text Rank Algorithm*

One of the drawbacks observed in Abstractive summarization, though they possess human paraphrasing capabilities to summarize, still they miss significant check-points (or phrases) in input text. This is expected since whenever we reduce objects in input space into a smaller contextual representation there is some information loss. To overcome this, we suggest to use the Text Rank Algorithm which can be used to compute scores and rank significant words. This additional information can be passed as a feature along with the Glove Word Embeddings.

We implemented TextRank instead of relying on vanilla implementation available in standard libraries like *gensim* because the standard implementation of TextRank prunes words and compute their ranks but the use-case of abstractive summarization needs weights for each of the words represented in the given graph of text.

Below are the steps executed inorder to get scores,

1) Cleaning: In order to perform scoring of text we had to first normalize it by lower-casing all the text. We filter out text characters that are non-printable. For each word we receive in our input text we search for its lemma. **For words that are adjective comparative or adjective superlative, we lemmatize them separately**. We filter words based on standard stop-word list as well based on pos-tags. We consider only those words which are tagged as either: Noun, Adjective and Verb.

2) Graph Construction: Below figure 1 represents the graph developed by our Text Rank algorithm. Its algorithm that we implemented is mentioned on next page. We consider words within a window size and create edges between them if they occur in same window.

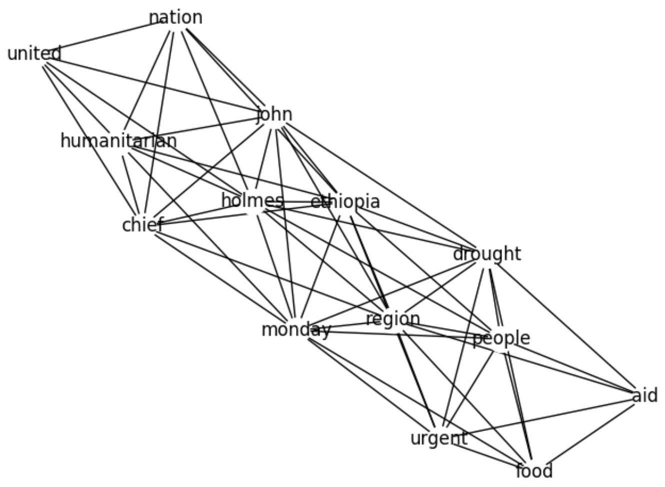


Fig. 1: Graph generated by TextRank algorithm for window size 6

3) Ranking Text: The TextRank algorithm [3] is inspired by Google's PageRank algorithm for web pages. It creates a text graph of a document which is a weighted undirected graph where each word is a vertex in the graph while edge weights denote the distance between the words in a sliding window. Figure 1 shows

Algorithm 1 Graph Construction

```
1:  procedure BUILD-GRAPH(Vocab, Text, Window)
2:      $V \leftarrow \phi$
3:      $Score \leftarrow \phi$
4:      for each $word$ in $Vocab$ do
5:          $u \leftarrow$ Create-Vertex($word$)
6:          $V \leftarrow V \cup u$
7:          $Score[u] = 1$                ▷ Initial Default score
8:      $weights \leftarrow [][]$
9:      for each $u$ in $V$ do
10:         for each $v$ in $V$ do
11:             $weights[u][v] \leftarrow 0$
12:     $n \leftarrow |Vocab|$
13:     for each $(u, v) \leftarrow V * V \mid u! = v$ do
14:         for $start$ in {1 to n - window - 1} do
15:             $end \leftarrow start + Window - 1$
16:             $substring \leftarrow Text(start, end)$
17:             if u ∈ substring & v ∈ substring then $weights[u][v] \leftarrow weights[u][v] + 1/dist(u, v)$
18:     return $weights, V, Score$
```

Algorithm 2 Rank the text based on Graph

```
1:  procedure RANK-GRAPH(Weights, Score, V)
2:      $inoutWeights \leftarrow$
3:      for each $u$ in $V$ do
4:          for each $u$ in $V$ do
5:              inoutWeights[u] = inoutWeights[u] + weights[u][v]
6:      $maxIterations \leftarrow 50$
7:      $damp \leftarrow 0.85$
8:      $threshold \leftarrow 0.0001$
9:      for i in 1, $\ldots maxIterations$ do
10:         $prevScore \leftarrow Score$
11:         for each $u$ in $V$ do $sum \leftarrow 0$
12:             for each $v$ in $V$ do $sum \leftarrow sum + Weights[u][v]/(inoutWeights[u] + Score[v]$
13:             $Score[u]$ = (1-damp) + damp * $sum$
14:             if $\Sigma(prevScore_i - Score_i) <= threshold$ then
15:                 Terminate the algorithm
        return $Score$
```

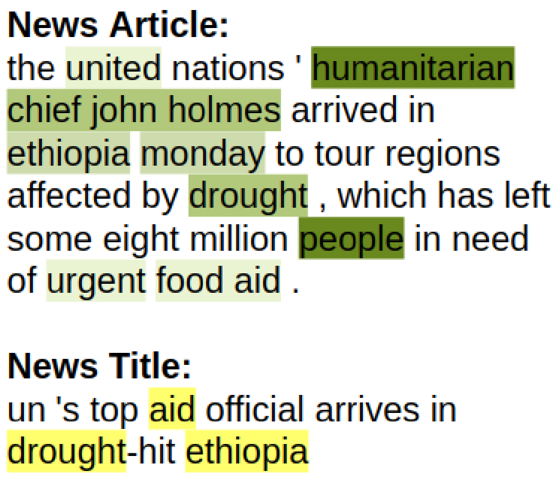
a textrank graph created for a sample news article with a window size of 6. We apply a modified PageRank formula to this graph incorporating weights as shown below,

$$TR(V_i) = (1-d) + d * \sum_{V_j \epsilon In(V_i)} \frac{w_{ji}}{\sum_{V_k \epsilon Out(V_j)} w_{jk}} TR(V_j)$$

The TextRank algorithm works because co-occurring words recommend each other as important, and it is the common context that enables the identification of connections between words in text. Consider a sample news article with the following text "the united nations' humanitarian chief john holmes arrived in ethiopia monday to tour regions affected by drought , which has left some eight million people in need of urgent food aid". Table I shows the TextRank scores along with their normalized scores for this sentence. It can be observed that important words from the sentence are identified by the TextRank algorithm giving them higher scores while non-important words have lower scores.

word	score	normalized
people	1.1300387	1
humanitarian	1.1300366	0.9999981417
drought	1.08964	0.9642501624
chief	1.089638	0.9642483926
john	1.065463	0.9428553199
holmes	1.0523089	0.9312149221
monday	1.0523086	0.9312146566
ethiopia	1.0481833	0.9275640737
urgent	0.99123394	0.8771681359
food	0.6471075	0.5726418927
united	0.64710397	0.5726387689
aid	0.15	0.1327388168

TABLE I: TextRank scores of important words for sample news article


News Article:
the united nations ' humanitarian chief john holmes arrived in ethiopia monday to tour regions affected by drought , which has left some eight million people in need of urgent food aid .

News Title:
un 's top aid official arrives in drought-hit ethiopia

Fig. 2: Important words identified by the TextRank algorithm from the News Article and their occurrence in the News Title

B. *Beam Search in Decoder*

In the basic decoder model, only the word with the highest probability in the previous step is passed on to the next RNN cell in the decoder. The drawback of this approach is that, if a wrong word is selected at the one of the time steps, its effect is passed forward to the following sequences resulting in the generation of incorrect sentences. Hence we propose to use the Beam

Search algorithm in the decoder of the baseline model. In beam search, the "beam width" (B) indicates the top B mostly likely word / sequence to be considered and passed to the next RNN cell. For a beam width of 3, if the top 3 words with the highest probabilities are "in", "Jane" and "September" at time step t then the next word is predicted then the beam search algorithm predicts the next word by taking each to these three words as input, and evaluates the probability of generating the whole sequence and pick the top 3 sequences. Figure 3 shows the top 3 sequences at time step $t+1$ are "in september", " Jane is" and "Jane visits". In this case, the sequences starting with "september" (one of the candidates in the previous step) is eliminated. These three sequences are then used to predict the next word until $< EOS >$ is reached.

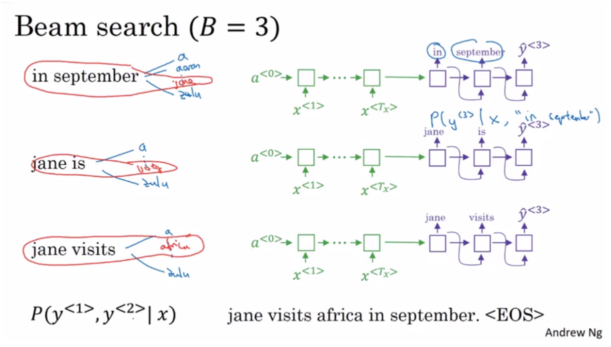


Fig. 3: Beam Search Decoder

C. Architecture

The baseline implementation [1] had basic encoder-decoder structure. The encoder is a bi-directional sequence-to-sequence model (LSTM). The decoder is a simple sequence-to-sequence model. We compute attention over hidden states generated at encoder.

Unlike baseline model, we introduce Text-Rank embedding for each word in the input sentence. We append it to standard-baseline input structure comprised of: Glove, Tf-IDF, POS tag, etc.

As explained previously we rely on Beam Search Decoder to improve our overall prediction of word at output layer at Decoder. The Beam Search helps at each time step to keep our options opened for multiple words that could be paraphrased at that position.

V. Evaluation

There are various standardized evaluation metrics [5] experimented for the task of text summarization. Both intrinsic and extrinsic evaluation methods exist for summary evaluation measures. While the extrinsic methods focus on downstream NLP tasks like document categorization, information retrieval and question answering,

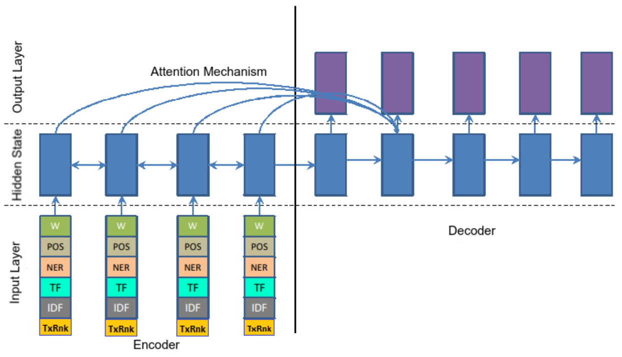


Fig. 4: Architecture of the Proposed Model

assessing the accuracy of these tasks, it will be more time consuming for training these downstream pipelines and then evaluating them. Thus we focus on evaluating our model based on intrinsic evaluation methods only and make a direct comparison with the baseline model. The intrinsic evaluation methods are largely grouped into two categories where one group of methods contains text quality evaluation measures while the other group contains content based evaluation. As the baseline model is evaluated on content based evaluation methods, we evaluate our model on similar evaluation methods, specifically n-gram matching methods called ROUGE.

A. Recall-Oriented Understudy for Gisting Evaluation

ROUGE [6] consists of a family of measures $ROUGE_n$ which tries to evaluate the summary based on the similarities of n-grams between the generated summary and the reference summary. The $ROUGE_n$ score of a candidate summary is computed as -

$$ROUGE_n = \frac{\sum_{C \varepsilon RSS} \sum_{gram_n \varepsilon C} Count_{match}(gram_n)}{\sum_{C \varepsilon RSS} \sum_{gram_n \varepsilon C} Count(gram_n)}$$

where RSS is the Reference Summary Set, $Count_{match}(gram_n)$ is the number of matching n-grams and $Count(gram_n)$ is the total number of n-grams. The $ROUGE_n$ score computes the recall score as the number of matching n-grams between the generated and reference summary divided by the total number of n-grams in the reference summary. There is also another ROUGE score which is based on the longest common subsequence matching called the $ROUGE_L$ score. We compute the $ROUGE_1$, $ROUGE_2$ and $ROUGE_L$ scores measuring unigram, bigram and LCS statistics for the baseline and the textrank model. Figure 5 shows the comparison of the evaluation measures for the baseline model and the textrank model. The evaluation scores are calculated based on the average evaluation score on a set of 50 validation

Example Summaries
A: japanese share prices rose # percent in morning trade thursday to hit the highest level in more than five years as fresh gains on wall street fanned upbeat investor sentiment here, dealers said **RS:** tokyo shares rise # percent in morning trade **BS:** tokyo shares close # percent higher **TS:** tokyo shares inch up # percent
A: us business leaders lashed out wednesday at legislation that would penalize companies for employing illegal immigrants **RS:** us business attacks tough immigration law **BS:** us business leaders lash out against illegal immigrants **TS:** us business leaders denounce penalties for illegal immigrants
A: two egyptian border guards were killed wednesday in clashes with palestinian militants near the rafah crossing on the border with gaza, a medical source said **RS:** two egyptian guards killed on border with gaza **BS:** two egyptian border guards killed in clashes **TS:** two egyptian militants clash in gaza strip

TABLE II: Example Summaries from the validation set. **A:** Source Article, **RS:** Reference Summary, **BS:** Baseline model summary, **TS:** TextRank model summary

samples different from the training set.

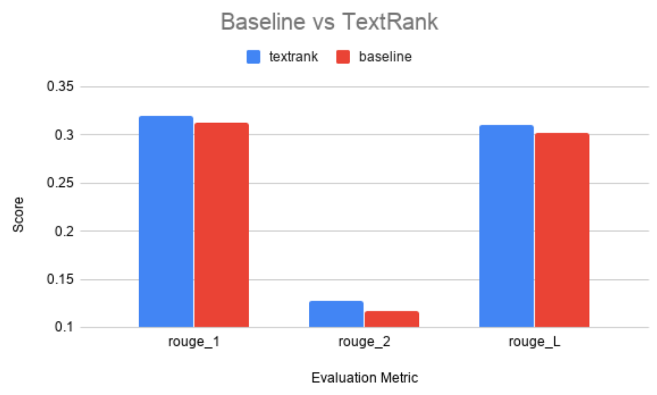


Fig. 5: Comparison of Average Evaluation Metrics for baseline and textrank model

The TextRank model performs slightly better than the baseline model in all of the evaluation metrics with an average $ROUGE_1$ score of 0.32, $ROUGE_2$ score of 0.13 and $ROUGE_L$ score of 0.31. We believe this is because the TextRank model is able to better understand the important words from the text and use that in the summary which matches with the reference summary slightly better than the baseline model which has no information of the important words in the text.

Table III shows a comparison of the summaries from the baseline model and the textrank model along with the reference summary. As we can see for all the basic measures Text-Rank out-performed Base-Line by a small margin. As you observe these values are low, since we were successful to run these only until 10 epochs. If we run for 50 epochs, we propose that the expected range of scores would be much better. Regardless, you can observe the monotonic relationship between scores of Baseline and Text-Rank based model.

Model	Bleu	rogue_1	rogue_2	rogue_L
Text-Rank	8.00595	0.32012	0.12787	0.31012
Base-Line	6.18066	0.31295	0.11727	0.30161

TABLE III: Comparison of baseline vs text-rank based model

VI. Future work

Correcting grammar of sentences is altogether a different domain and tremendous research has been done in this area. Neural machine Translation (NMT) and phrase-based model (PBMT) are used for correcting grammar. One such famous online tool for grammar correction used is Grammarly [8]. We can try to incorporate the idea of a grammar correction model in our system to enhance the grammar of the generated summary.

VII. Conclusion

In this paper, we have discussed some of the proven ways to get great abstractive summarization by using Attention based Bidirectional LSTM for Encoder and Decoder. Also, it was coupled with some of the features such as using Parts of speech tags, TF and IDF tags, large vocab trick (LVT), handling OOV words (Out of Vocabulary words) by using generator-pointer switch and applying hierarchical attention to sentences and words. We enhanced this model by incorporating TextRank embeddings in the input vector along with using a Beam Search decoder in the output layer. Evaluation of the model shows improvement over the baseline model using these techniques.

References

[1] Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond [Online].
Available: https://www.aclweb.org/anthology/K16-1028.pdf
[2] Dataset: https://github.com/harvardnlp/sent-summary
[3] TextRank: Bringing Order into Texts [Online].
Available: https://web.eecs.umich.edu/ mihalcea/papers/mihalcea.emnlp04.pdf
[4] Beam Search Decoder [Online].
Available: https://ctcoding.wordpress.com/2019/05/04/study-notes-sequential-model-week-3/
[5] Evaluation Measures for Text Summarization [Online].
Available: http://www.cai.sk/ojs/index.php/cai/article/viewFile/37/24
[6] Automatic Evaluation of Summaries Using N-gram Co-Occurrence Statistics [Online].
Available: https://www.aclweb.org/anthology/N03-1020.pdf
[7] Github : https://github.com/theamrzaki/text_summurization_abstractive_metho
[8] Grammarly: https://www.grammarly.com/

[9] Textrank Preprocessing: https://github.com/JRC1995/TextRank-Keyword-Extraction
[10] Glove: https://nlp.stanford.edu/pubs/glove.pdf