Final Project

Text Summarization

Dhruvin Patel
University of Mississippi
Oxford, Mississippi
drpatel@go.olemiss.edu

ABSTRACT

Since the beginning of the internet, the amount of information available from news articles, research papers, blogs, etc. has dramatically increased. Therefore, the use of text summarization can be a great tool process information. Auto text summarization can help find the relevant and important information from text without having human interpretation.

ACM Reference Format:
Dhruvin Patel. 2018. Final Project: Text Summarization. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 MOTIVATIONS

My motivation for making a text summarization comes from voice assistants, surprisingly. Voice assistants have changed how people get information, but one downfall is that most voice assistants require a human to enter latest information. I want to take multiple news sources with information about the University and the town of Oxford to extract an accuracy summary of the yesterday's news. For the scope of the class, I plan to find ways to better and experiment with ideas on how to achieve possible summaries. Using auto text summarization methods to find a way to get a brief few sentences in multiple documents, I can achieve this. The goal is to have a text summarizer be able to know the important and relevant information from documents, have that information in the summary, and create a summary that is readable that is also grammatically and coherent.

Also, I hope to integrate the text summarizer to my Alexa Skill for people to use. I want to utilize the model I create from this project to implement it in summarizing real news articles. Having a voice assistant summarization the news will provide the user a quick and concise update on information.

2 APPLICATIONS

The applications for text summarization can be use in any form of large text. For example, web summarization, scientific articles summarization, and email summarization are just a few examples where auto text summarization can be applied. For my project, I want to fine tune a text summarization for news articles.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3 APPROACHES/RELATED WORKS

Currently text summarization can be broken down into two different approaches: *extraction* and *abstraction* [1]. Extraction summarization methods works by pinpointing relevant phrases from the text and copying sentences to form a summary. In sense, extraction summarization is copying and pasting important sentences from the original text to get a summary. On the other hand, abstraction summarization methods try to interpret the text and generate a summary using different words and phrases not in the original text. Abstraction methods are similar to how humans summarize text by using paraphrasing sections of the text.

Most of the research in text summarization is in extraction summarization methods. This is because abstraction methods tries to fix problems such as semantic representation, inference and natural language generation and easier to implement[1]. However, extractive summaries lack generalization or the incorporation of real-world knowledge. For my project, I want to use a combined approach using extractive and abstraction methods to create a summary.

Extraction methods can be split into three independent tasks: 1) Creates an intermediate representation of the input text. 2) Score the sentences based on representation. 3) Select sentences to make up a summary [1]. Extractions methods for sentence extraction scoring can be split into methods like graph base, feature base, and topic base. Graph based models are influenced by PageRank algorithm where sentences are the vertices and the edges indicated how similar they are the sentences are. Feature based models extract features from a sentences like position in a document, length of sentence, term frequency, etc to score each sentence. Topic based models try to evaluate each sentence by which topic is mentioned in the sentence.

Nowadays, abstraction methods have utilized sequence to sequence models for summarization. A sequence to sequence model is composed of an encoder and decoder, in which the encoder converts an input text into a vector representation, and the decoder generates a summary by using the encoders input. However, the basic encoder decoder model has many issues including how to retain important words or phrases. In Nallapati et al. paper, they proposed using an Attention encoder-decoder Recurrent Neural Network, which was originally proposed for machine translation to use in text summarization [3]. The Attention mechanism keeps track of the inputs to remember, which helps focus on the important keywords from a sentence. This model improved abstractive method but did have some weakness such as the ability to produce incorrect factual detail inaccurately and lend to be repetitive [4].

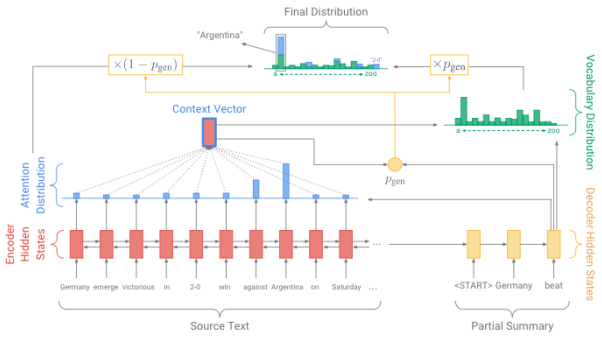

Figure 1: See et al. pointer-generator model [4]

In order to fix these issues, See et al. proposed a mix approach using both abstraction and extraction, where they creating a pointer-generator network that allows both copying words and generating words[4]. Also, See et al. also purpose a coverage network which helps solve the problem of repetition[4].

4 PROPOSES

For my final project, I want to explore way to combine both extractive and abstraction methods or make progressive in See et al. model using pointer-generator and coverage. I plan on making a baseline sequence to sequence model as a baseline to compare for future changes to my project. From there, I want to experiment with adding to my model to improve it.

I plan on implementing See et al. pointer-generator and coverage network [4]. The pointer-generator helps the RNN from producing incorrect factual detail incorrectly and the coverage helps with repetition in the summary. I want to research more into how both mechanism work and try to improve the results. In See et al. model, it tends to copy more than abstract/paraphrase words [4]. After, I want to fine tune these methods and make changes to improve the results. From viewing the sample summarization, I identify that it copies more of the original text. In order to fix these, I want to research more into how the pointer-generator decides to copy from the text.

My project can be broken down into parts. For example, the first step is to create a sequence to sequence baseline model for testing purposes. Then, researching into ideas like pointer-generator and implementing them. Next, would be to fine tune my model to see if I could get better performance. After, I want to evaluate the weakness of the model and last is to try to fix these weaknesses.

4.1 Possible Solutions

I have identify a weakness in See et al. implementation. Their pointer-generator mechanism tends to copy from the original text more frequently than generating a word from the vocabulary distribution. I want to modify the pointer-generator so that it will generate words more after copying from the text. The pointer-generator currently works as a soft switch that determine to copy from the original text by using a generation probability that is calculated from the context vector, the decoder state, and the decoder input. In figure one, it shows how the pointer-generator choose to generate a word from the vocabulary distribution or copy from the attention distribution.

One idea I had was to replace the pointer-generator mechanism. When an unknown word is output from the decoder, use the pretrained vector embedding from either Word2Vec or GloVe and replace the unknown word with the closed word using cosine distance. This will allow replacing an unknown words with out having to copy from the text and copy large part of the text after it. One issue with this is that It will not be able to detect proper nouns that do not have replaceable words.

Another idea is to add an additional parameter to the generation probability that holds the previous states from the decoder. This parameter will help control the copying after copying a word from the original text. After a word is copied, we can increase the parameter to negatively affect the probability. One down side of this approach is that it might give a low probability when there is multiple unknown words in an order within the decoder. In order to learn the parameter, I will have to adjust the probability distribution.

5 DATASET/SETUP

I have chose to use the CNN dataset, which was used in [4] and [3]. The different between [4] and [3] is that they used both CNN and Dailymail articles and summarizes. The CNN contains 92,579 articles with one summary for each. I preprocess all the articles and summaries by removing html tags, titles, and any none ascii characters.

Because of the increase in training time, I chose to train and test my model with 20 percent and 15 percent of the data. With 20 percent of the data, there are 16,182 articles in the training and 6,935 in the testing sets. With 15 percent of the data, there are 9,720 in training and 4,166 in the test set. I will discuss why I chose to limit the dataset in the results discussion.

6 EVALUATION/BASELINE

Evaluation on a summary is difficult assignment because there is no absolute summary for a document. Creating a good summary is subjective. A person might think a summary is good while another person might like the same summary is not so good. But what can be agreed on is that a summary should have important, relevant information in a grammatically correct sentence.

Baseline evaluation will be done with ROUGE, which is a widely used metric for automatic evaluation. ROUGE states for Recall-Oriented Understudy for Gisting Evaluation. There are different variations of ROUGE evaluation. One in particular is ROUGE-N, which is a N-gram measure between the model and gold summary. Generally, Rouge-1 and Rouge-2 are used to for the baseline results.

For my baseline model, I will make a simple sequence to sequence model using my dataset and capture the Rouge-1 and Rouge-2 performance on the dataset. This will help by having baseline comparison to my project. Also, it will help me understand how to build a sequence to sequence model and improve the model.

I will be comparing my results from the baseline to [4], which used the same dataset has I did. I will be comparing the two rouge scores. In their paper, they have a recall at 28.97 percent for rogue-1 and 8.79 for rogue-2. I hope to match these results. The difference between my model and See et al. is that they set the encoder and

decoder to each 256 hidden states, used 128 word embedding, and build their own attention mechanism. On the other hand my model had 256 hidden states on the encoder and 100 hidden states in the decoder. I used 50 for the dimensional of the word embedding, and I used Tensorflow's seq to seq library for Bahdanau's Attention.

In addition to using Rouge metric, the summaries will to be review to check the overall flow of the summary. For example, checking if the sentence is syntactically correct sense.

7 BASELINE TEST

This section will explain how I setup my baseline encoder-decoder model and the results and evaluations from my model. I will explain how I preprocess each article, and describe the network in detail.

7.1 Methodology

The baseline follows most of encoder decoder models. First, I preprocess the articles and gold summaries by removing any characters that are not letters, numbers, or punctuations. Then, I removed the beginning of the article because it contains '(CNN) – '. After cleaning the article, I split each article into sentences and then into tokens. Then I make a lexicon of words in the articles and summaries. To create an embedding for the words I decided to use GloVe's pretained word embedding with 50 dimensions. Using the GloVe embedding and the lexicon of words, I made a dictionary for each token in the article it is assigned a number. If a token is not in GloVe, then we use a UNK token. For the start of a sentence and end of a sentence, I used SOS and EOS token. This will allow converting a sentence into an integer vector. I also created a reverse dictionary that had the integers as the key and the word is the value.

The encoder is a one layer bidirectional LSTM (Long-Short Term Memory) cell with 256 hidden states. The decoder is a one directional LSTM cell with 100 hidden states. The tokens from an article are fed into the encoder one at a time to produce an encoder hidden state. On each time step t, the decoder receives the word embedding for the previous word. When training, the previous word is from the reference summary, and when generating summaries, the previous word is what is generated from the previous decoder time step. At each tilmestep of the decoder, there is a dense layer with the number of hidden units equal to the vocabulary size. This will give the probability of the word at each time step.

My model uses Bahdanau's Attention [2]. Bahdanau's Attention can be seen as the probability distribution over the source tokens. This will tell the decoder where to look to produce the next token.

When training, the batch size for my model is 10 summaries per batch. Each sentence in the batch is fixed on the max sentence length, so the sentences are padded with <PAD> token. Some of the hyper parameters on my network are batch size, RNN size, learning rate, min learning rate, learning rate decay, and keep probability of neutron. The batch size equal 10. The RNN size is 256 units. The learning rate is 0.005. The min learning rate is 0.001. The learning rate decay is 0.95, and the keep probability of a neutron is 0.95.

The loss function of the network is a weighted cross-entropy loss for a sequence of the generated logits. I decided to use Adam optimizer with gradient clipping. Gradient clipping caps the magnitude of the gradient and can make stochastic gradient behave better because it prevents gradients from getting too large.

	Rouge		
	1	2	L
Seq-to-Seq + Attn baseline (See el at.)	30.49	11.17	28.08
My Baseline Seq-to-Seq + Attn	12.97	1.78	10.14

Table 1: ROUGE score on test set. See et al. was trained on the whole CNN/Daliymail dataset with more epochs.

7.2 Results/Evaluation

My results are given in Table 1. I reported the ROUGE-1, ROUGE-2, and ROUGE-L that measures the word overlap, bi-gram overlap, and longest common sequence between the reference summary and the summary to be evaluated. I displayed See et al. baseline seq-to-seq model with attention ROUGE scores with my model. From the table, my scores are dramatically lower See et al. The ROUGE-2 score is almost near zero. I believe this is because my model is under fit to the articles. I was only able to run 10 epochs with my dataset because of time constraints. These results are only for the 25 percent of the dataset. Using 15 percent of the dataset lead to poor performance. In particular, the ROUGE score were near 0.001 percent with training with 33 epochs. Therefore, I chose not to display the result of using only 15 percent of the dataset. In comparison, See et al. baseline model run 33 epochs with the full CNN/Dailymail dataset, which contain 312,084 total articles. Also, in See et al. baseline model they used a beam search to produce multiple generated summaries for one article. I was unable to implement the beam search in my baseline model.

In Figure 2, I show a truncated article with the gold summary and my model's generated summary. In the article, it was talking about Egyptian Prosector sentencing the president and other government member to prison. In the Gold Summary provided, the summary correctly paraphrases the article. On the other the hand, my model baseline summary is not grammatically or semantically correct. The model summary also tend to very repetitive in repeating phrases multiple times. The generate summary also generated the name Nasser, which was one in the article. This is an example of seq-to-seq model generating incorrect factual information.

8 LIMITATION

While working on this project, I ran into multiple problems. One problem was choosing the right dataset to run my model. At first I thought I would be able to run all the CNN/Dailymail dataset, but when I started to run my model, I realized that I do not have enough computing power or time to train for that long. Second problem I ran into was learning tensorflow library for seq to seq models. This was my first program using tensorflow with seq to seq model and I spend a while learning it. Third problem, I ran into was debugging and trouble shooting my model. I had a lot of issue with decoding the model summarizer. Another limitation was computation power, abstractive summarizer require a lot of computation power to train models. I used my computer with a Nvidia 1080 GPU, and the training time still required at least 1 day of time to complete. The last limitation I had was time. A lot of time went into this model and I had to take some time out at the end to focus on other projects. This is the reason I should not get to writing the pointer-generator and coverage network.

Figure 2: The baseline model produces summaries that tend to be repetitive and factual incorrect. The red text shows incorrect factual information the generate summary produces. The green text shows repetitive text.

9 CONCLUSION

All in all, working on this project has been a learning experience about how abstractive text summarizer works. With all the limitation to this project, I have learned how to code seq to seq model with tensorflow. I soon found out that abstractive methods are not great at generating summarization. From my results, I found that extractive methods would be easier to implement and have better results than abstractive methods; however, abstractive methods try to generate human like sentences. I wished I had more time to work on implementing my own version of the pointer-generator and coverage network.

REFERENCES

[1] Mehdi Allahyari, Seyed Amin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text Summarization Techniques: A Brief Survey. *CoRR* abs/1707.02268 (2017). arXiv:1707.02268 http://arxiv.org/abs/1707.02268

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014). arXiv:1409.0473 http://arxiv.org/abs/1409.0473

[3] Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-Sequence RNNs for Text Summarization. *CoRR* abs/1602.06023 (2016). arXiv:1602.06023 http://arxiv.org/abs/1602.06023

[4] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. *CoRR* abs/1704.04368 (2017). arXiv:1704.04368 http://arxiv.org/abs/1704.04368