News Headline Generation

MSiA 490 Text Analytics Project

Github: https://github.com/jiedali/msia490-nlp-news-summarization

Jieda Li

Abstract

This goal of this project is to build a news headline generator by using different text summarization techniques. I have experimented with both abstractive and extractive type of method for text summarization. My main focus is on abstractive method, for which I used a bi-directional LSTM-Attention based encoder decoder model. For extractive method, I used a K-means clustering based approach using pre-trained BERT generated sentence embeddings, finally I tried the classic extractive method of TextRank.

1 Introduction

The purpose of this project is to build a News Headline Generator using text summarization technique. I have experimented both abstractive and extractive methods and have compared the results on the same news article and observed interesting results from each one of the models. For abstractive model, I used an bi-directional LSTM-attention based encoder-decoder model and performed training using Gigaword dataset on Google Colab, I evaluated the model performance using ROUGE score on the test set from the Gigaword data. For extractive model, I use a K-means clustering based method using pre-trained BERT generated sentence embedding and another well-known extractive method TextRank.

2 Related Work

Text summarization in general falls into two categories: extractive or abstractive. **Extractive** is to pick sentences from original text document, the existing work on the extractive generally follows a methodology by first converting the sentence into sentence embeddings, the embeddings can use various approach, such as TF-IDF score, or Word2Vec embeddings, with such sentence embeddings, we then score the sentences, there are different ways to score them, one way could be to compute the cosine similarity between target sentence and the rest of the sentences. Lastly based on the score, we can rank all the sentences and pick the top k sentences that forms the summary. Some work in this category are in reference [1] and [2]. A famous extractive summarization method is called TextRank [2], which is inspired by PageRank, it is a graph-based ranking model, the TextRank algorithm treats each sentence as a node and compute a weighted link between each sentence and the rest of the sentences, the weights of the link can be decided by a similarity score. Once we have the weighted links, we can rank the sentences similar to how PageRank works. **Abstractive** method, on the other hand, reproduces a new shorter text that conveys the most critical information from the original text in a new way, after interpretation and examination of the input text using neural network model. Abstractive summarization is more advanced and closer to human-like interpretation, but it is also more difficult because it involves a comprehensive understanding of the original text as well to produce a new sentence that is cohesive and captures the key meaning. It usually requires training of deep neural network models. A typical abstractive summarization approach deploys a seq2seq architecture, which involve an encoder and a decoder, the encoder is converting the input text into some form of representation, and the decoder takes the transformed representation and outputs the predicted tokens sequentially [3] [4].

The encoder approach recently evolved from using RNN (LSTM) based architecture to transformer-based architecture, transformer-based architecture uses attention mechanism to "bake" the understanding and because it takes all the token all at once (compares to LSTM-base which takes the token one at a time) it was able to significantly reduce training time. One notable implementation of abstractive method is Pointer-Generator Network [3], this architecture has improved upon previous work by allowing copying from original text and avoid repetition of words that often is an issue with LSTM-based model.

3 Dataset

I am using the Harvard NLP **Gigaword** dataset (https://www.tensorflow.org/datasets/catalog/gigaword). It is for headline-generation on a corpus of article pairs from Gigaword consisting of around 4 million articles. There are two features in the dataset: **Article** and **Title** (which serves as the summary). An example data in this dataset looks like following:

Title
australian current account deficit narrows sharply
Article
australia's current account deficit shrunk by a record #. ## billion dollars -lrb- #. ## billion us -rrb- in the june quarter due to soaring commodity prices, figures released monday showed.

There are 3,803,957 records in the training set, 189,651 records in the validation set, with another 1951 in the test set.

4 Method

I have experimented with following 3 methods to perform the summarization. Both abstractive and extractive, detailed below.

4.1 Method I: Abstractive Method

The abstractive method I used is referenced in [4], it is using an Encoder-Decoder Model with attention mechanism. For the encoder, it uses bi-directional LSTM architecture with Bahdanau attention mechanism, for the decoder, it uses a basic LSTM decoder for training and a Beam Search Decoder for inference. The code is referenced from repo [5] and I have made modifications to the code to port it over to Jupyter notebook and performed training on Google Colab. The training curve and model performance is detailed in "Result" section. Following diagram shows a basic architecture of this model.

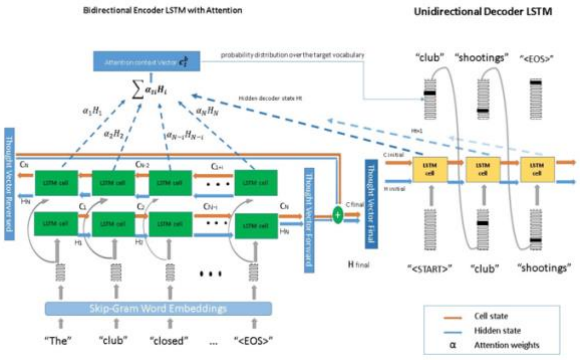

Figure 1: Architecture of the Encoder-Decoder Model used for Method 1 Abstractive model, this figure is cited from [7].

4.2 Method II: Extractive Method

The extractive method I used is referenced from paper [1]. It follows such an idea: first we create sentence embeddings with pre-trained BERT model, with the sentence embeddings, we then perform K-means clustering based on how many sentences that the user would like to be in the summary. Finally, we select the sentence that is closest to the cluster centroid as the summary sentence. Below is an illustration of the model.

The model uses a pre-trained BERT model to create sentence embeddings for the superior performance of BERT and because it uses K-means that means we are measuring distance by Euclidian.

Figure 2: Method II Extractive Model Illustration

4.3 Method III: TextRank

Lastly, I tried the classic method TextRank. TextRank is built based on graph theory similar to PageRank. It treats each sentence in the text as a node and compute a weighted link between one sentence and the rest and ranks the sentence similar to PageRank. TextRank has existing Python

Library built for it called PyTextRank and I used the library implementation.

5 Results

5.1 Comparison of the results

I searched for one piece of the latest news article on the topic of the U.S. election outcome. It is published on ***Chicago Tribune*** on ***November 22, 2020*** and is about the latest information on Trump is requesting to recount Georgia's ballot after it has been called for Joe Biden. Table 1 first shows the original news article and then the 3 summaries produced by the three methods.

Chicago Tribune, News article on 11/22/2020

Trump team requests another recount of Georgia's day after results certified in Joe Biden's victory
President Donald Trump's campaign requested a recount of votes in the Georgia presidential race on Saturday, a day after state officials certified results showing Democrat Joe Biden won the state, as his legal team presses forward with attacks alleging widespread fraud without proof. Georgia's results showed Biden beating Trump by 12,670 votes out of about 5 million cast, or 0.25%.State law allows a candidate to request a recount if the margin is less than 0.5%.Republican Gov. Brian Kemp formalized the state's slate of 16 presidential electors. The Trump campaign sent a hand-delivered letter to the secretary of state's office requesting the recount in an election that has been fraught with unfounded accusations of fraud by Trump and his supporters. A Trump legal team statement said: "Today, the Trump campaign filed a petition for recount in Georgia. We are focused on ensuring that every aspect of Georgia State Law and the U.S. Constitution are followed so that every legal vote is counted. President Trump and his campaign continue to insist on an honest recount in Georgia, which has to include signature matching and other vital safeguards."
Summary from Model 1 (Abstractive Model)
uk hopes of certainty for ballot recount in voter race
Summary from Model 2 (Extractive Model)
President Donald Trump's campaign requested a recount of votes in the Georgia presidential race on Saturday, a day after state officials certified results showing Democrat Joe Biden won the state, as his legal team presses forward with attacks alleging widespread fraud without proof.
Summary from Model 3 (TextRank)
President Donald Trump's campaign requested a recount of votes in the Georgia presidential race on Saturday, a day after state officials certified results showing Democrat Joe Biden won the state, as his legal team presses forward with attacks alleging widespread fraud without proof.

Table 1: Comparison of Summary with 3 Methods

The result from abstractive model is so interesting! It gives "**uk hopes for certainty for ballot recount in voter race**" I was really amazed the moment I see this output. First, for some reason, it mistakenly thinks the country is uk. And it is stated in previous literature [3] that this type of model is liable to reproduce factual details inaccurately. (The Pointer-Generator Network was able to improve upon it but it is outside the scope of my project) Second, it used the word **hopes for certainty** and I found it amazing since the word certainty never appears in the news article, but yet the phrase **hopes for certainty** captures the overall tension and sentiment in this article and the model comes up with the phrase on its own. Third, the word **ballot recount** also never appears in original text but the model precisely captures this key action that is being discussed in this article. And overall, the sentence it produces is perfectly correct grammar wise.

However, there are drawbacks of the abstractive model, not only it appears it got some facts wrong, thinking the country is ***uk***, it also didn't pick up some important fact, for example, in this case, the news is specifically about ***Georgia***'s result and the model doesn't catch that. The Pointer-Generator Network tried to alleviate this by allowing direct copy of certain words from original text [3].

The results from the extractive methods seem easy to understand, both extractive models pick the first sentence in the news article. Which, me, as a human, has checked and think it is also the best representation of the overall content of this article.

5.2 ROUGE score of the abstractive model

In this section I will show the training curve of the abstractive encoder-decoder model and the evaluation metric results. The training is performed on Google Colab and it takes about 10 hours to get the model I have at this point. Performance is evaluated using the ROUGE metric, specifically ROUGE-1, ROUGE-2 and ROUGE-L. ROUGE scores are a recall-based measure that it counts the overlap of n-grams between the prediction and reference summaries.

ROUGE scores are computed for each article in the test set and the result reported here is by averaging

the score across all articles in the test set. There are 1951 articles in the test set.

Rouge-1	***Rouge-2***	***Rouge-L***
35.5	14.3	33.4

Table 2: ROUGE scores on the validation set, average across the ROUGE scores for each article in the validation set

Note that I am reporting the ROUGE score in a format that is consistent with other paper are reporting 35.5 meaning 35.5% based on the formula to calculate the ROUGE scores. I found one reference that also reported ROUGE score on the same data set and the results are 34.2, 16.2 and 31.9, see reference [7]. It appears the result we obtained, although not matching up with the state-of-the-art model performance, but still achieved a ROUGE score that falls into elative same range.

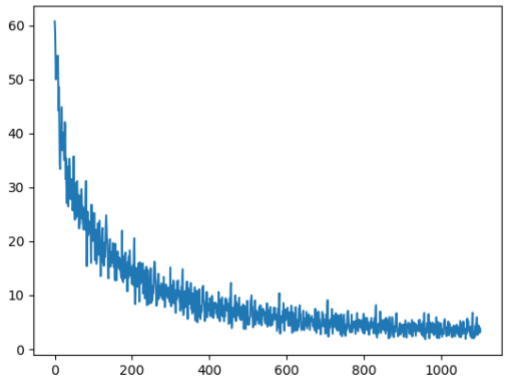

Figure 3: Training loss curve Bi-LSTM with Attention Encoder-Decoder Model, the curve shows the training for approximately 10 hours. (X-axis is every 100 optimization steps)

Lastly, I listed a few summary/reference pairs that are generated by the abstractive model on the test data set for the reader to get a sense of how does the model generated summary look like. Label "ref" is the reference headline/summary, while label "sum" is model generated summary.

ref	**canada recommends avoiding travel to nepal**
sum	canada advises to avoid nepal travel
ref	**ford executive sees weaker us auto sales in #**
sum	us auto sales to lower in #
ref	**two test positive for bird flu virus in turkey**
sum	two suspected of bird flu vaccines in turkey

Table 3: Examples of generated summaries

5.3 Model Productization

I created a REST app using python flask package. The user can input a raw text document and the app will return model generated summary. The REST service is built using the model in Method II, the extractive method. (The abstractive model was too large to make it work conveniently with the REST app).

6 Discussion

Here I summarize pros and cons with each type of method. For **extractive** model, since no training is needed, computation is very fast. Also, because of the fact that we are selecting sentences from original text, it is guaranteed to be cohesive. And there is advantage with simply just copying sentences because you are almost guaranteed to have the factual details correct.

For the **abstractive** model, since it involves training a deep neural network model, it involves more computational resource to train. But the abstractive model was able to perform a much more complicated and efficient summarization of the original article. I read through ~20 summaries generated by the model and most of them are cohesive and match the meaning in the reference summary. Although the drawbacks of neural abstractive model are also obvious, that it can easily get the factual details wrong, or missing facts, at least for the network architecture that I am using here.

References

Derek Miller 2019. *Leverageing BERT for Extractive Text Summarization on Lectures, Arxiv*

Rada Mihalcea and Paul Tarau 2004. *TextRank: Bringing Order into Text, Proc. Of the 2004 conf. on empirical methods in natural language processing*

Abigail See, Peter J Liu and Christopher D. Mannning 2017. *Get to the point: Summarization with Pointer-Generator Networks*

Sumit Chopra, Michael Auli and Alexander M. Rush 2016. *Abstractive Sentence Summarization with Attentive Recurrent Neural Networks*

https://github.com/dongjun-Lee/text-summarization-tensorflow

https://github.com/DeepsMoseli/Bidirectiona-LSTM-for-text-summarization-