News Article summarization using Multi-Document Summarization approach.

Natural Language Processing

Final Project Report

Submitted By Akshay Kumar Gupta, Garv Jasuja

Contents

Overview 3
Problem Statement 3
  Novelty 3
Dataset 3
  Data transformation 4
Solution 4
Implementation 4
Experiment 4
Result 5
Conclusion 5
Future Work 6

Overview

Multi-document summarization (MDS) is a challenging task in natural language processing (NLP) that involves generating a concise summary of multiple documents, typically news articles, that cover the same topic. The goal of our project is to determine the most talked sector of industry using MDS on real time news articles. We want to build our solution through which we can suggest people sectors in which they should invest with the use of MDS and sentimental analysis. We trained our model on a dataset containing 10,200 unique clusters comprising of 2.39 million articles.

Problem Statement

Multi-document summarization (MDS) is a task in natural language processing (NLP) that involves generating a summary of multiple documents that discuss the same topic. The goal of MDS is to provide a concise representation of the key information in the documents, which can be used to quickly understand the main ideas and themes. MDS is a challenging task in NLP because it involves processing large amounts of text and identifying the most important information across multiple sources.

News articles are often used as a source of information for predicting stock prices in the share market. This is because news articles can provide valuable insights into the financial and economic factors that may affect the performance of a company or an industry.

Novelty

Through our approach we aim to train models using a database which contains multiple articles with the same summary. In normal scenario whenever some news starts trending, there is a high probability that users share same news content but through different articles, henceforth all will have same summary. Through our model, we try to find the connection between multiple articles to detect the most trending sector. This approach hasn’t been tried before, in which MDS is used on real time news articles to predict most talked sector.

In case of any new happening in the financial world it's possible that some news articles are reported through multiple reporters. But for analysts all they need is a summary of all the articles. So, what we are trying to do is train a model on sources of multiple news articles which report the same incident and generate a summary. This way when fed news about an event from various sectors it will be able to understand and produce a single summary making it an invaluable tool.

We trained our data on news sets, and we wish to apply to financial news. However, our dataset is a mixture of all and not just news.

Dataset

We are using WCEP dataset for multi-document summarization (MDS), which contains short, human-written summaries of news events that are obtained from the Wikipedia Current Events Portal (WCEP), along with a cluster of news articles associated with each event. The dataset consists of 10,200 clusters split in train, test and validation set. It contains 2.39 million news articles published between 2016 – 2019 year.

The news articles in each cluster are the sources cited by editors on WCEP and are extended with additional articles automatically obtained from the Common Crawl News dataset. The dataset is designed to provide a large-scale resource for training and evaluating MDS algorithms and can be used to test the ability of such algorithms to identify the most important information across multiple sources and generate a concise summary of the key points. By training the model on a diverse set of news articles, it can learn to recognize patterns in language use and structure, which can lead to more accurate and coherent summaries.

Data transformation

We have done some pre-processing on data before feeding it into the model for training. We have used google/mt5-small pre-trained transform mainly because it can handle input from multiple languages. This helped us to train our model in bilingual mode, which will improve the accuracy of the model. Taking input size of 512 and output size of 30 we used the tokenizer. We have used this to extract features from the dataset.

Solution

We are using TFMT5ForConditionalGeneration pre-train model which is TensorFlow implementation of the MT5 (Multilingual Translation Transformer) model for conditional generation tasks, such as text summarization, machine translation, and text generation. This model is based on the transformer architecture and is pre-trained on a large multilingual corpus. The TFMT5ForConditionalGeneration model can be fine-tuned on a specific task by providing it with a dataset of input/output pairs and adjusting the model's weights to minimize the difference between its generated output and the desired output.

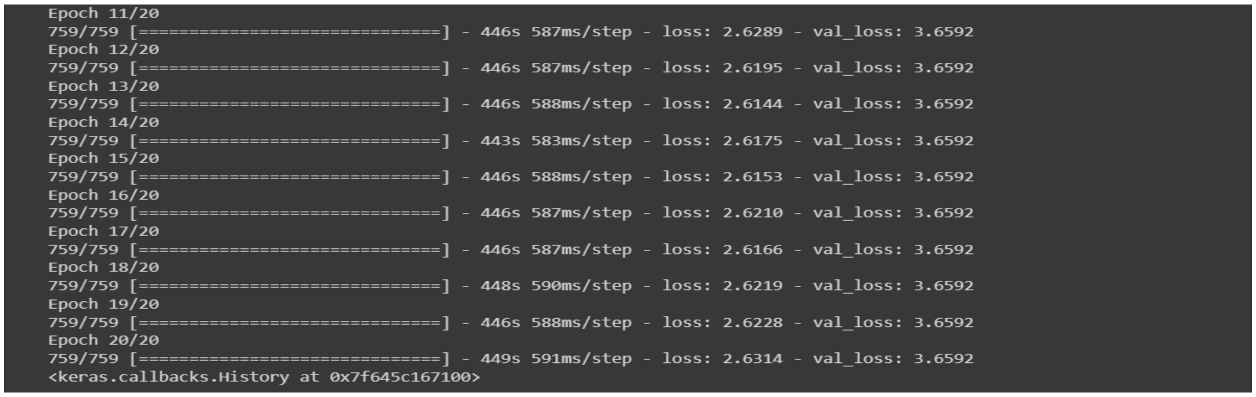
Main motivation of using this model was multilingual support and capability of fine-tuning using custom dataset. The model is trained on a large multilingual corpus, which allows it to generate summaries in different languages. This is particularly useful for summarizing news articles that are published in different languages, allowing for more comprehensive news coverage. The model can be fine-tuned on a specific dataset for a specific summarization task. This allows the model to be tailored to specific domains and improves the quality of the generated summaries.

Implementation

We have trained our dataset on pre-existing TFMT5ForConditionalGeneration model on 20 epochs. We have split the dataset in 80-10-10 ratio of training, testing and validation. The dataset consisted of 650k articles, but because of the large size of dataset and limited resources we have done training on limited dataset of approximately 200k random articles. This was the maximum size we could use due to the constraint on RAM and GPU available. We have used the input size of 512 in our model, this was because of the limitation of RAM available. To evaluate the performance, we have used ROGUE score. ROGUE evaluates the overlap between the generated text and one or more reference texts, measuring the degree to which the generated text captures the most important information contained in the reference texts.

Experiment

We have trained our model on 20 epochs using a dataset of approx. 200k random articles with their summary.

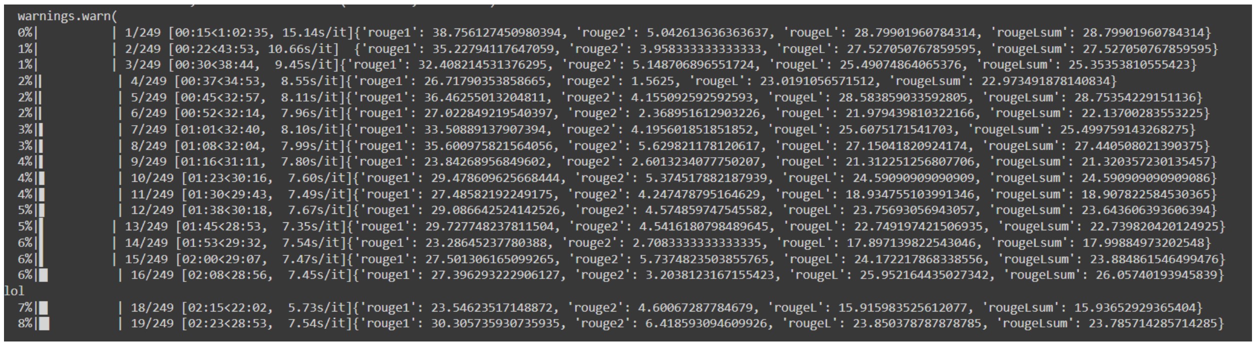


```
Epoch 11/20
759/759 [==============================] - 446s 587ms/step - loss: 2.6289 - val_loss: 3.6592
Epoch 12/20
759/759 [==============================] - 446s 587ms/step - loss: 2.6195 - val_loss: 3.6592
Epoch 13/20
759/759 [==============================] - 446s 588ms/step - loss: 2.6144 - val_loss: 3.6592
Epoch 14/20
759/759 [==============================] - 443s 583ms/step - loss: 2.6175 - val_loss: 3.6592
Epoch 15/20
759/759 [==============================] - 446s 588ms/step - loss: 2.6153 - val_loss: 3.6592
Epoch 16/20
759/759 [==============================] - 446s 587ms/step - loss: 2.6210 - val_loss: 3.6592
Epoch 17/20
759/759 [==============================] - 446s 587ms/step - loss: 2.6166 - val_loss: 3.6592
Epoch 18/20
759/759 [==============================] - 448s 590ms/step - loss: 2.6219 - val_loss: 3.6592
Epoch 19/20
759/759 [==============================] - 446s 588ms/step - loss: 2.6228 - val_loss: 3.6592
Epoch 20/20
759/759 [==============================] - 449s 591ms/step - loss: 2.6314 - val_loss: 3.6592
<keras.callbacks.History at 0x7f645c167100>
```

Figure 1 Training Epoch Progress

Result

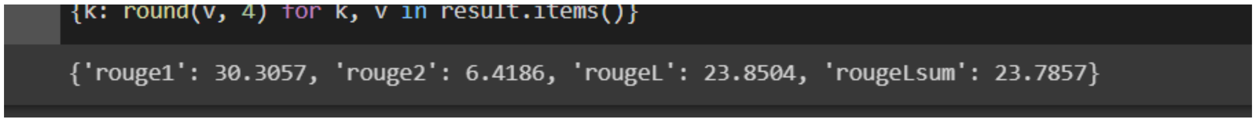
We have used ROGUE score to evaluate the accuracy of the model. Below is the ROGUE score of some articles on our model. As we can notice the score varies between 25 and 35. This is considered an average accuracy score.



```
warnings.warn(
 0%|          | 1/249 [00:15<1:02:35, 15.14s/it]{'rouge1': 38.756127450980394, 'rouge2': 5.042613636363637, 'rougeL': 28.79901960784314, 'rougeLsum': 28.79901960784314}
 1%|          | 2/249 [00:22<43:53, 10.66s/it]  {'rouge1': 35.22794117647059, 'rouge2': 3.958333333333333, 'rougeL': 27.527050767859595, 'rougeLsum': 27.527050767859595}
 1%|          | 3/249 [00:30<38:44,  9.45s/it]{'rouge1': 32.408214531376295, 'rouge2': 5.148706896551724, 'rougeL': 25.49074864065376, 'rougeLsum': 25.35353810555423}
 2%|▏         | 4/249 [00:37<34:53,  8.55s/it]{'rouge1': 26.71790353858665, 'rouge2': 1.5625, 'rougeL': 23.0191056571512, 'rougeLsum': 22.973491878140834}
 2%|▏         | 5/249 [00:45<32:57,  8.11s/it]{'rouge1': 36.46255013204811, 'rouge2': 4.155092592592593, 'rougeL': 28.583859033592805, 'rougeLsum': 28.75354229151136}
 2%|▏         | 6/249 [00:52<32:14,  7.96s/it]{'rouge1': 27.022849219540397, 'rouge2': 2.368951612903226, 'rougeL': 21.979439810322166, 'rougeLsum': 22.13700283553225}
 3%|▎         | 7/249 [01:01<32:40,  8.10s/it]{'rouge1': 33.50889137907394, 'rouge2': 4.195601851851852, 'rougeL': 25.6075171541703, 'rougeLsum': 25.499759143268275}
 3%|▎         | 8/249 [01:08<32:04,  7.99s/it]{'rouge1': 35.600975821564056, 'rouge2': 5.629821178120617, 'rougeL': 27.15041820924174, 'rougeLsum': 27.440508021390375}
 4%|▎         | 9/249 [01:16<31:11,  7.80s/it]{'rouge1': 23.84268956849602, 'rouge2': 2.6013234077750207, 'rougeL': 21.312251256807706, 'rougeLsum': 21.320357230135457}
 4%|▍         | 10/249 [01:23<30:16,  7.60s/it]{'rouge1': 29.478609625668444, 'rouge2': 5.374517882187939, 'rougeL': 24.59090909090909, 'rougeLsum': 24.590909090909086}
 4%|▍         | 11/249 [01:30<29:43,  7.49s/it]{'rouge1': 27.48582192249175, 'rouge2': 4.247478795164629, 'rougeL': 18.934755103991346, 'rougeLsum': 18.90782258453O365}
 5%|▍         | 12/249 [01:38<30:18,  7.67s/it]{'rouge1': 29.086642524142526, 'rouge2': 4.574859747545582, 'rougeL': 23.75693056943057, 'rougeLsum': 23.643606393606394}
 5%|▌         | 13/249 [01:45<28:53,  7.35s/it]{'rouge1': 29.727748237811504, 'rouge2': 4.5416180798489645, 'rougeL': 22.749197421506935, 'rougeLsum': 22.739820420124925}
 6%|▌         | 14/249 [01:53<29:32,  7.54s/it]{'rouge1': 23.28645237780388, 'rouge2': 2.7083333333333335, 'rougeL': 17.897139822543046, 'rougeLsum': 17.99884973202548}
 6%|▌         | 15/249 [02:00<29:07,  7.47s/it]{'rouge1': 27.501306165099265, 'rouge2': 5.7374823503855765, 'rougeL': 24.172217868338556, 'rougeLsum': 23.884861546499476}
 6%|▋         | 16/249 [02:08<28:56,  7.45s/it]{'rouge1': 27.396293222906127, 'rouge2': 3.2038123167155423, 'rougeL': 25.952164435027342, 'rougeLsum': 26.05740193945839}
lol
 7%|▋         | 18/249 [02:15<22:02,  5.73s/it]{'rouge1': 23.54623517148872, 'rouge2': 4.60067287784679, 'rougeL': 15.915983525612077, 'rougeLsum': 15.93652929365404}
 8%|▊         | 19/249 [02:23<28:53,  7.54s/it]{'rouge1': 30.305735930735935, 'rouge2': 6.418593094609926, 'rougeL': 23.850378787878785, 'rougeLsum': 23.785714285714285}
```

Figure 2 ROGUE score per article result

Overall ROGUE score of all testing set in 30.3057, which is considered average score.



```
{k: round(v, 4) for k, v in result.items()}

{'rouge1': 30.3057, 'rouge2': 6.4186, 'rougeL': 23.8504, 'rougeLsum': 23.7857}
```

Figure 3 Overall ROGUE result

Conclusion

Through our project we have trained our model using a novel approach where multiple articles are printed with the same summary. This approach will help us to correctly predict the similarity of similar news articles through our model and hence predict the most talked about after industry sector in real time. This information can help people in share trading. Our model performed fairly in summarizing the text using multiple document summarization approach.

Future Work

We were not able to train our model on a complete dataset due to resource limitations. We believe if the model is trained using full dataset, it will perform much better. We aim to build this model where we can generate the sentiment of people on specific sector using sentiment analysis on summarized news article.