IST 664/ CIS 668: Natural Language Processing

Final Project Report

Abstractive Text Summarization using Multi-head Attention Transformers

By:
Prabin Raj Shrestha
Neha Reddy Gaddam
Shivani Agarwal

Table of Contents

Introduction:

Text summarization is the process of creating a summary of a certain document that incorporates the original's most important features. Its objective is to compile a list of the key ideas in the document. Today's massive volume of data makes text summarization more crucial than ever. In contrast to classic text summary techniques, which merely choose and combine existing sentences, more modern approaches to text summarization have focused on abstractive summarization, which requires producing new sentences that capture the crucial information from the original text. Abstractive summarization is still a challenging undertaking since it must balance the correctness of the summary with its concision and readability.

Abstractive summarizing involves writing a fresh, streamlined summary from the source text's most crucial information. It requires understanding the content of the text and creating fresh, concise sentences that express the same meaning as the original text. A technique known as abstractive summarizing aims to condense the text's key concepts while maintaining its readability.

By utilizing the capabilities of neural networks to produce high-quality summaries that precisely capture the salient points from the original text, attention transformers represent a viable method for abstractive text summarization. A specific sort of neural network architecture called an attention transformer enables the model to selectively attend to various parts of the input text at each stage of the summary generation process, focusing on the information that is most pertinent to the current summary phrase. Numerous NLP tasks, including text generation and machine translation, have proven to be very successful when using this strategy.

In this project, we'll use attention transformers to create an abstractive text summarizer. We would use a sizable dataset of news items to develop and train a model. Building a model and assessing its performance are our goals.

Objective:

The purpose of this study is to provide an overview of our methodology, the precise steps we took to develop the attention transformer model, the dataset we used to train and test the model, and more. The study will give a general overview of text summarization, discuss how important it is becoming due to the availability of vast amounts of data, and describe how to do it specifically using abstractive summarization.

Literature Review:

Abstractive text summarizing is a challenging problem in natural language processing (NLP), necessitating the development of a summary that draws out the crucial information from a text while maintaining its style and meaning. Text summaries have traditionally been created using extractive algorithms that merely choose and combine existent phrases, however these methods typically result in repetitious and disjointed summaries. However, abstractive summarizing has been shown to be a more effective technique for producing high-quality summaries. It entails creating new phrases that are not present in the original text.

As a result of recent advances in NLP, attention-based neural network topologies are now practical for a range of tasks, including text generation and machine translation. The model can use attention methods to selectively focus on different parts of the input text at each stage of the summary creation process, allowing it to identify the information that is most important for the current summary phrase. Due to the promising outcomes of this approach, attention-based models have been created for the task of abstractive text summarization.

The attention-based strategy was first described in the well-known paper "All you need is attention," written by Ashish Vaswani, and published in 2017. One of the most popular attention-based techniques for abstractive text summarizing is the transformer model, which uses an encoder and decoder architecture and analyzes input text while producing the summary. The transformer model employs self-attention strategies at each stage of the summary production process to allow the model to focus on different regions of the input text, resulting in summaries that are more accurate and coherent.

In general, transformer-based attention-based models, in particular, have demonstrated a great deal of promise for abstractive text summarization, and it is anticipated that they will continue to be a hot focus of research in the NLP community.

Attention

Attention is a deep learning strategy that enables the model to focus just on portions of the input data that are most pertinent to the job at hand. It helps the model to use information about the relative weights of various input factors to make predictions that are more accurate.

Attention is widely used in natural language processing (NLP) sequence-to-sequence models, where an input sequence (such as a sentence) is encoded into a fixed-length representation and then decoded into an output sequence (such as a summary or translation). The model uses attention at each decoding phase to concentrate on different parts of the input sequence depending on where the decoding process is in the process.

Different attention techniques include additive, multiplicative, and dot-product attention, to name a few. The Transformer model uses dot-product attention, which is widely popular in NLP and has gained state-of-the-art performance in many NLP tasks.

Prior to matching each key with the query to produce a score, the query vector, key vectors, and value vectors are all linearly transformed in dot-product attention. The scores are normalized using the SoftMax function, and the weighted value vectors that result are then joined together to get the final output.

When training a text summarization model, creating a mask is a crucial step. A mask is employed in the context of attention-based models to prevent paying attention to specific input sequence segments during training. By doing this, the model is prevented from "peeking ahead" and using knowledge from the future to create predictions, which is not possible during inference.

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)$$

d_k stands for the dimension of queries and keys.
Q stands for Query.
K stands for Key.
V stands for Value.
M stands for Mask.

Query, Key, and Value are three vectors used in attention mechanisms to compute the attention weights, which are used to determine which parts of the input to focus on.

- *Query:*
  The model's current state, or the "query" that the attention mechanism is attempting to respond to, is represented by the Query vector. The query vector in natural language processing could reflect the word that is currently being processed or the decoder's current hidden state in a machine translation system.

- *Key:*
  The input being processed is represented by the Key vector. The hidden states of the encoder, which encode the input sequence in the source language, could be represented by the key vector, for instance.

- *Value:*
  The associated output for each key vector is shown by the Value vector. The hidden decoder states that produce the output sequence in the target language in the language translation example may be represented by the value vector.

- *Mask:*
  The model should focus only on the article when generating the summary, and not be influenced by the summary itself. The mask blocks any attention to the summary tokens while training the model. This ensures that the model learns to summarize the article based only on the information available in the input sequence. Masking is done using very negative values that will yield a similar effect to using $-\infty$.

To establish how much attention should be paid to each key vector, the attention mechanism calculates a score between the query and key vectors. A dot product between the query and key vectors is used to get the scores, which are then normalized using the softmax function. The context vector reflecting the most pertinent portions of the input is created by computing a weighted sum of the value vectors using the attention weights that are generated.

The Query, Key, and Value vectors are fundamental elements of the attention processes as they enable the model to selectively focus on different inputs at different moments. The model can learn to focus on the most crucial elements of the input and improve performance on a variety of tasks by predicting attention weights based on the similarity between the query and key vectors.

Data Collection and Preprocessing

Data collection

For abstractive text summarization models to be successful, the amount and quality of the training dataset are essential. It has been demonstrated that large datasets with a variety of high-quality text considerably enhance model performance. But gathering and preprocessing such datasets can be a laborious and difficult task.

The CNN/Daily Mail dataset, the New York Times dataset, and the WikiHow dataset are a few well-known datasets that have been used for training abstractive text summarization algorithms. The CNN/Daily Mail dataset, which has been widely used as a benchmark dataset for text summarization, consists of news stories from CNN and the Daily Mail and their related highlights (i.e., summary sentences). The New York Times dataset is a sizable dataset that contains articles and summaries from The New York Times and has been used to train models for a variety of NLP applications. The WikiHow dataset is made up of articles from the well-known how-to website WikiHow and the how-to summaries that go with them.

Data preprocessing

Data preprocessing is a crucial component of abstractive text summarization since it has a significant impact on the dataset's quality and the effectiveness of the models built using it. Sentence tokenization, word tokenization, and data cleaning (such as eliminating HTML tags and special characters) are common preprocessing techniques. Additionally, some researchers have enhanced the quality and diversity of the training dataset using more sophisticated methods like data augmentation and sentence splitting.

One important technique for data preprocessing is the use of language models that have already been trained, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) for abstractive text summarization. These models are trained on a large amount of text data and can be customized for a particular task, like abstractive text summarization. An already trained language model can be improved to perform better while using less training data.

Overall, the effectiveness of abstractive text summarization models depends on the collection and preprocessing of high-quality and diverse datasets. The training dataset has been produced using a variety of methods, including the use of pre-trained language models for fine-tuning.

Our approach

We have proceeded with CNN/Daily Mail dataset as It is a popular benchmark dataset for abstractive text summarization.

Below are some of its advantages:

- **Large-scale:** With more than 300,000 articles and their related highlights, the dataset is rather sizable. The dataset's size makes it possible to create reliable and efficient summarization models.
- **High-quality:** The dataset was obtained from trustworthy news sources (CNN and Daily Mail), and editors with experience in writing for newspapers wrote the highlights. As a result, the dataset has a high level of quality, and the summaries are well-written and educational.
- **Diverse:** Politics, science, entertainment, and other subjects are only a few of the many themes covered in the articles in the collection. This diversity of themes guarantees that the summarization models developed using the dataset can produce summaries for a variety of text inputs.
- **Publicly available:** Since the dataset is openly accessible, researchers can use it to create and assess their summarization models. As a result, there is now a wealth of knowledge about abstractive text summarization, making the dataset a valuable tool for the NLP community.

The input text is prepared for the summarization model using our text processing technology in a number of phases. The incoming text is first divided into individual words or tokens using the Treebank tokenizer. Following lemmatization and lowercase tokenization, the inflectional forms of the words are changed to their dictionary-based or base forms. In turn, this leads to a smaller vocabulary and greater efficacy of the summary model. We ultimately interconnect the tokens in order to produce a processed text.

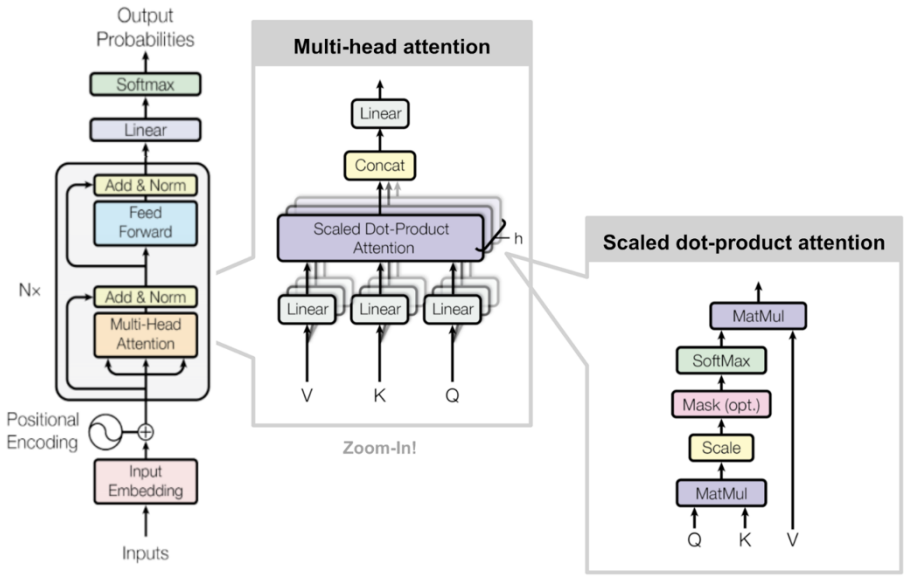
Lemmatization is a benefit of our strategy since it allows us to cut down on the number of vocabulary terms that are unique, which enhances the effectiveness of the summarization model. Furthermore, by leaving stop words in, we are able to keep the input text's syntactic structure, which is crucial for producing high-quality summaries.

We incorporate the Trax tokenizer into our pipeline after text processing is complete. The output of the Trax tokenizer is another generator that either combines the training stream or translates tuples into other tuples. It turns the text of a tuple's first element into a NumPy integer array through tokenization so that it may be passed directly to the summarization model. In order to prevent the model from becoming overfit to the training data, we also use Trax's shuffle function to randomly organize the samples.

Overall, the input language is made simpler by our way of text processing, allowing for an effective and fast summary. By combining tokenization, lemmatization, and Trax tokenization, we may improve the performance of the summarization model while also simplifying the input data.

Methodology

For abstractive text summarization, we have incorporated a Transformer-based language model. A layer that predicts the probability distribution over the vocabulary follows a layer that contains a multi-layer Transformer decoder block in the model. The dot product attention and causal attention techniques are used by the Transformer decoder block to concentrate on different components of the input sequence and generate the summary. Layer normalization, a dense layer, a ReLU activation function, a dropout layer, another dense layer, and a final dropout layer are all layers of the feedforward neural network used in the Transformer decoder block.

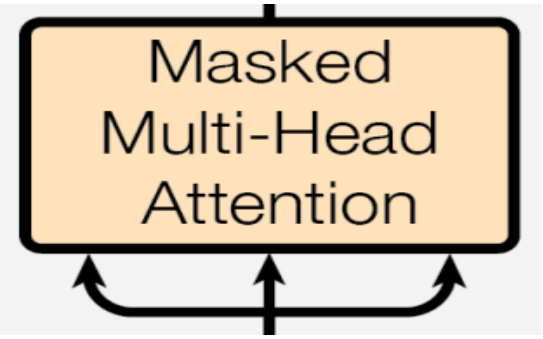


Source: 'Attention is all you need' Ashish Vaswani (2017)

Dot product attention:

The dot product attention mechanism is an essential component of the Transformer architecture that allows the model to manage different input sequence segments while creating the summary. In our technique, we first use dot product attention to compute the similarity between the query and key vectors, and then we use the similarity score to weigh the pertinent value vectors. As a result, the model may focus on the crucial parts of the input while producing the summary.

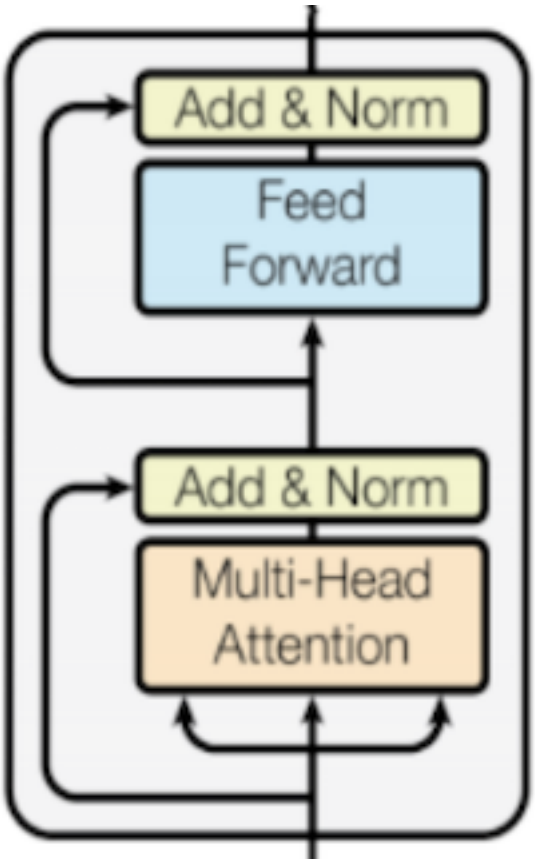
Casual attention:



Causal attention is yet another crucial tool we have incorporated into our technique. Similar to dot product attention, causal attention only pays attention to words that happened before the current time step. This is helpful when text summarization tasks are involved because we only want the model to construct the summary using data that has already been presented in the input text.

We initially define three functions: compute attention heads closure, dot product self-attention, and compute attention output in order to implement causal attention. The function that computes the attention scores for each head of the multi-head attention mechanism is known as the compute attention heads closure function. The dot product self-attention function computes the dot product attention scores from a query, key, value, and mask. To create the final output, the compute attention output function combines the value vectors and attention scores.

Transformer decoder:

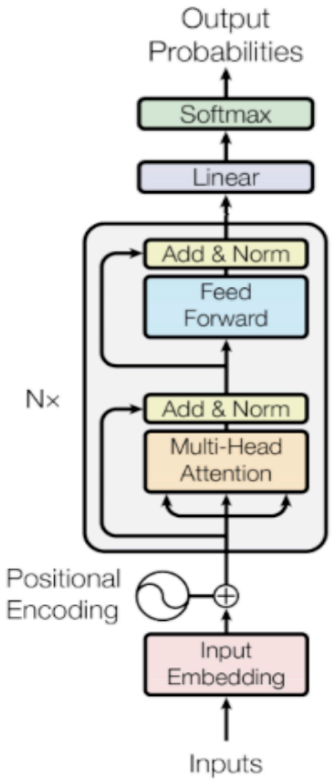


Another essential component of our procedure is the Transformer decoder block. Some of the layers in this system are layer normalization, dense layer, ReLU activation function, dropout layer, subsequent dense layer, and subsequent dropout layer. These layers help the model comprehend complex relationships between input and output sequences and generate accurate summaries.

Transformer Language Model:

Finally, our methodology incorporates a language model based on Transformers to produce the summary. A multi-layer Transformer decoder block and a linear layer that forecasts the probability distribution over the vocabulary make up the language model. In order to improve the model during training, we employ cross-entropy loss as the objective function and the Adam optimizer to update the model's parameters. To get the model to function at its best, we train it over a number of epochs.

In order to obtain cutting-edge outcomes in abstractive text summarization, our methodology uses advanced deep learning techniques, such as attention mechanisms and Transformer-based language models.



Training:

The cost function, optimizer, and training loop must all be defined prior to the model being trained. The Cross Entropy Loss function is utilized here as the cost function, while the Adam optimizer is utilized to optimize the model's parameters. The learning rate schedule that is submitted controls the learning rate.

The 'Training.Loop' function in Trax is used to build the training loop. The output directory where the model weights will be saved, any additional training parameters like the number of epochs and batch size, the model to be trained, a train task specifying the labeled training data, loss function, and optimizer, an eval task specifying the labeled evaluation data and metrics to evaluate the model's performance.

Evaluation & Testing:

Evaluation and testing are important steps in the machine learning workflow to assess the performance of a trained model. The widely used metric in the field of natural language processing to evaluate text summarization systems is the ROUGE metrics. They provide a quantitative measure of how well the generated summary captures the important information in the input text, and they allow researchers to compare different summarization systems and evaluate their performance.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used to evaluate text summarization systems. The metrics are based on comparing the generated summary against a set of reference summaries, and they measure how well the generated summary captures the important information in the input text.

ROUGE metrics are based on the concept of n-grams, which are sequences of words of length n. The most commonly used n-grams are unigrams (n=1), bigrams (n=2), and trigrams (n=3). The ROUGE metrics compare the n-grams in the generated summary against the n-grams in the reference summaries, and compute the recall (the proportion of reference n-grams that appear in the generated summary) and the precision (the proportion of generated n-grams that appear in the reference summaries).

ROUGE-1, ROUGE-2, and ROUGE-L are the three most popular ROUGE measures. The recall and precision of unigrams are measured by ROUGE-1, bigrams are measured by ROUGE-2, and the longest common subsequence between the generated summary and the reference summaries is measured by ROUGE-L.

Results

To create text given a starting prompt, we are using a greedy decoding method. With greedy decoding, the token with the highest probability is selected at each stage and used as the input for the following stage. By determining the argmax of the model's output, the first step of the greedy decoding algorithm determines the next symbol. The following input to the model will be the index of the token with the highest probability. Once the subsequent symbol has been acquired, it is added to the output sequence and the process is repeated until we reach a predetermined stopping criterion, such as a maximum length or an end-of-sequence token.

```
# Test it out with a whole article!
article = "It's the posing craze sweeping the U.S. after being brought to fame by skier Lindsey Vonn, soccer star Omar Cummi
print(wrapper.fill(article), '\n')
print(greedy_decode(test_sentence, model, 50, show_progress=True))
```

```
It's the posing craze sweeping the U.S. after being brought to fame by
skier Lindsey Vonn, soccer star Omar Cummings, baseball player Albert
Pujols - and even Republican politician Rick Perry. But now four
students at Riverhead High School on Long Island, New York, have been
suspended for dropping to a knee and taking up a prayer pose to mimic
Denver Broncos quarterback Tim Tebow. Jordan Fulcoly, Wayne Drexel,
Tyler Carroll and Connor Carroll were all suspended for one day
because the 'Tebowing' craze was blocking the hallway and presenting a
safety hazard to students. Scroll down for video. Banned: Jordan
Fulcoly, Wayne Drexel, Tyler Carroll and Connor Carroll (all pictured
left) were all suspended for one day by Riverhead High School on Long
Island, New York, for their tribute to Broncos quarterback Tim Tebow.
Issue: Four of the pupils were suspended for one day because they
allegedly did not heed to warnings that the 'Tebowing' craze at the
school was blocking the hallway and presenting a safety hazard to
students.

GENERATING SUMMARY:
The U.S. The U.S. The U.S. says the same-year-year-year-year-year-
year-old was found to be a-year-year-year-year-year-year-....(49)
 --------------

The U.S. The U.S. The U.S. says the same-year-year-year-year-year-
year-old was found to be a-year-year-year-year-year-year-
```

Conclusion:

In conclusion, we have implemented and trained a Transformer-based language model for text generation using Trax library.. We have successfully trained the model with limited resources on Google Colab and have shown that with more training steps, the accuracy of the model improves. However, we also recognize that the full potential of the model could not be obtained due to limited resources. Given more powerful resources, such as a GPU, we can increase the training speed and potentially achieve higher accuracy by training the model with more steps.

Future work:

To improve the performance of the model, we can take several steps. Firstly, we can use a more powerful machine, such as a GPU, to increase the training speed and enable us to train for a larger number of steps. This would allow the model to capture more of the nuances in the language and potentially improve performance.

In addition, we can explore other methods to improve the model's performance, such as fine-tuning the hyperparameters or experimenting with different architectures. We can also consider using a larger dataset or incorporating additional sources of data to improve the model's accuracy.Furthermore, we can use different evaluation metrics, such as the ROUGE score, to better measure the model's performance. The ROUGE score evaluates the similarity between the generated text and the reference text and can provide a more comprehensive measure of the model's performance.

References

Papers:

- Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond https://arxiv.org/pdf/1602.06023v5.pdf
- Attention is all you need: https://arxiv.org/abs/1706.03762
- Transformer Based Implementation for Automatic Book Summarization: https://ijisae.org/index.php/IJISAE/article/view/2421
- Review of automatic text summarization techniques & methods: https://tinyurl.com/S1319157820303712
- Neural Abstractive Text Summarization with Sequence-to-Sequence Models https://dl.acm.org/doi/abs/10.1145/3419106
- Text Summarization: A Brief Review https://tinyurl.com/978-3-030-34614
- Abstractive and Extractive Text Summarization using Document Context Vector and Recurrent Neural Networks https://arxiv.org/pdf/1807.08000.pdf
- Bootstrap Your Text Summarization Solution with the Latest Release from NLP-Recipes https://tinyurl.com/ba-p1268809

Articles:

- Abstractive Text Summarization Using Transformers: https://medium.com/swlh/abstractive-text-summarization-using-transformers-3e774cc42453
- Get started with Google Trax for NLP: https://towardsdatascience.com/get-started-with-google-trax-for-nlp-ff8dcd3119cf

Online Tutorial/Courses:

- Coursera Deep Learning Specialization by Andrew Ng https://www.coursera.org/specializations/deep-learning