Transformer-based Text Summarization and Comparative Analysis

Group Member: Changyu Liu, Jinda Zhang, Juan Yu, Ye Zhang

Problem statement

- Our research delves into the capabilities of transformer models in natural language processing

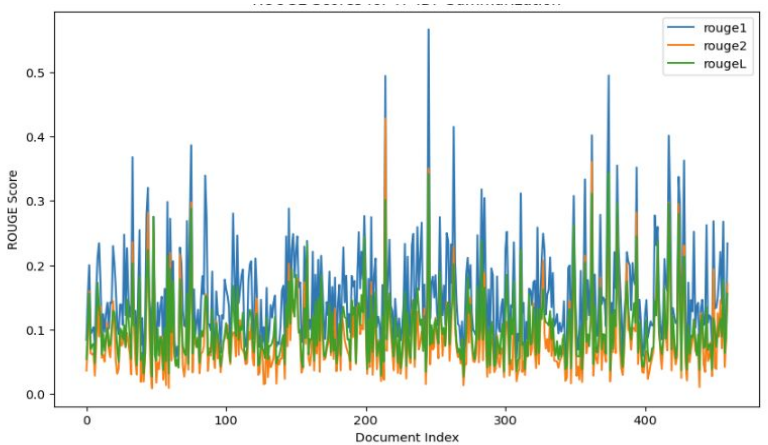
- We focus on the fine-tuning and comparative analysis of three prominent Large Language Models (LLMs) — Pegasus, T5, and Bart — tailored for automated text summarization.

Baseline Model

Random Summarization

- A simple baseline approach is random text summarization. This method randomly selects sentences from the document to create the summary

- The random summarization process involves splitting the document into sentences and randomly selecting a subset of sentences to form the summary.

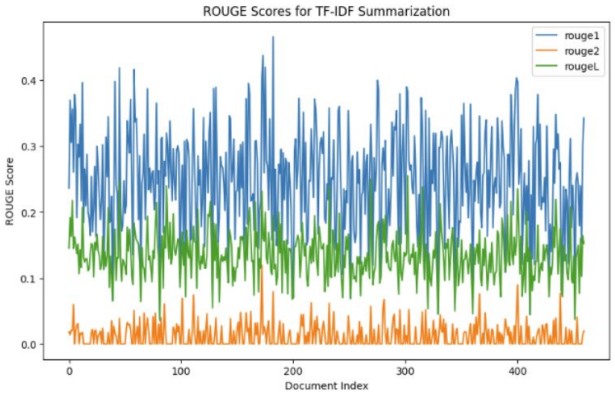


TFIDF Baseline

- TF-IDF is a widely used technique to represent the importance of words in a document corpus. We use this approach to generate extractive summaries for the given documents.

- computing the TF-IDF matrix using the scikit-learn library.

- The TF-IDF vectorizer is applied to the training set of documents, creating a matrix that represents the importance of each word in the corpus.

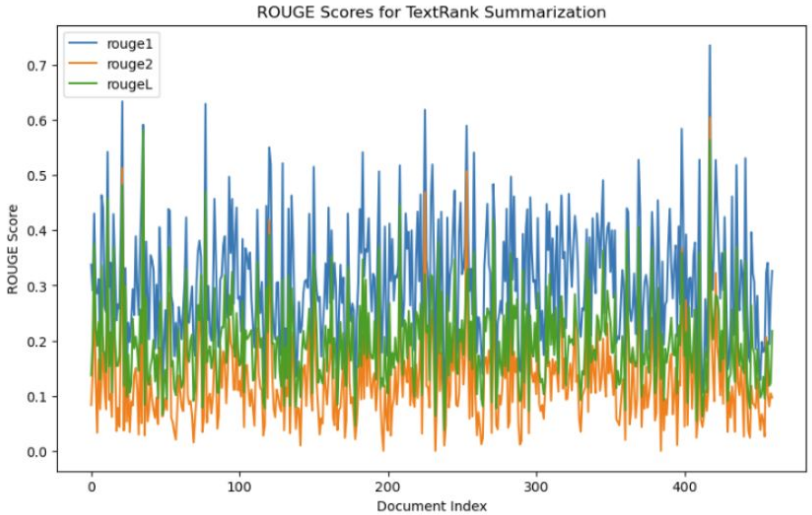


TextRank Algorithm

- TextRank is an unsupervised extractive summarization technique that assigns importance scores to sentences based on their similarity within the document.

- TextRank assigns importance scores to sentences and selects the top sentences to form the summary. The ratio parameter, set to 0.2 in our implementation, controls the length of the generated summary.

- We analyze the ROUGE scores obtained from TextRank summarization to evaluate the baseline performance.



Pre-trained LLMs

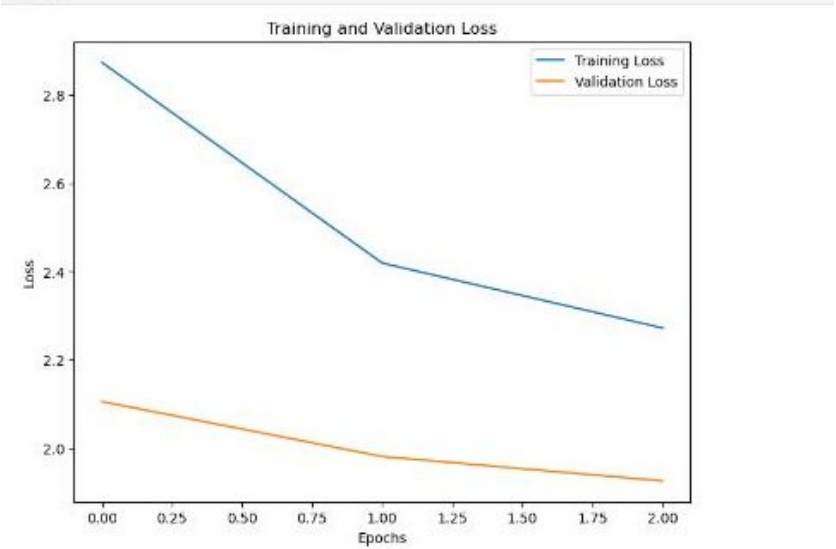
Pegasus

- state-of-the-art NLP model developed by Google Research
- abstractive summarization
- applications in various NLP tasks such as text summarization, content generation, and document understanding

Pegasus Training

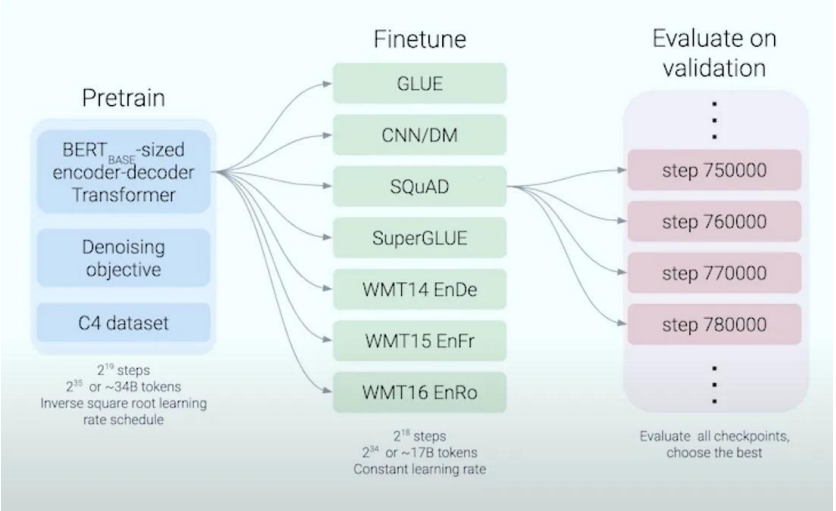
- There are 568M parameters in the model. We fine tune it on CNN / DailyMail Dataset

- 3 epochs,Some important fine tuning parameters are: learning_rate=2e-5, weight_decay=0.01,per_device_train_batch_size=16, and num_train_epochs=3



T5

- Text-to-Text Transfer Transformer
- Unsupervised pre-training + supervised fine-turning
- Easy to fine-tune according to downstream tasks, e.g. summarization



Source: Colin Raffel: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, https://www.youtube.com/watch?v=eKqWC577WlI&list=UUEqgmyWChwvt6MFGGlmUQCQ&index=5

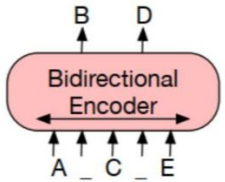
T5

- There are 60.5M parameters in the model. We fine tune it on CNN / DailyMail Dataset
- 2 epochs; Some important fine tuning parameters are: learning_rate=2e-5, per_device_train_batch_size=8, and num_train_epochs=2.

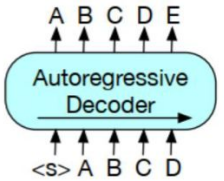
```
Epoch 1/2
35889/35889 [==============================] - 10181s 283ms/step - loss: 1.8782 - val_loss: 1.6667
Epoch 2/2
35889/35889 [==============================] - 10111s 282ms/step - loss: 1.8230 - val_loss: 1.6437
<keras.src.callbacks.History at 0x7a1fab2ba8f0>
```

Bart

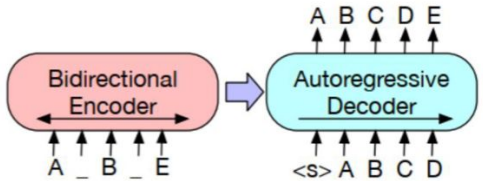
- Bidirectional and Auto-Regressive Transformers
- constructed from a bi-directional encoder like in BERT
- and an autoregressive decoder like GPT



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.

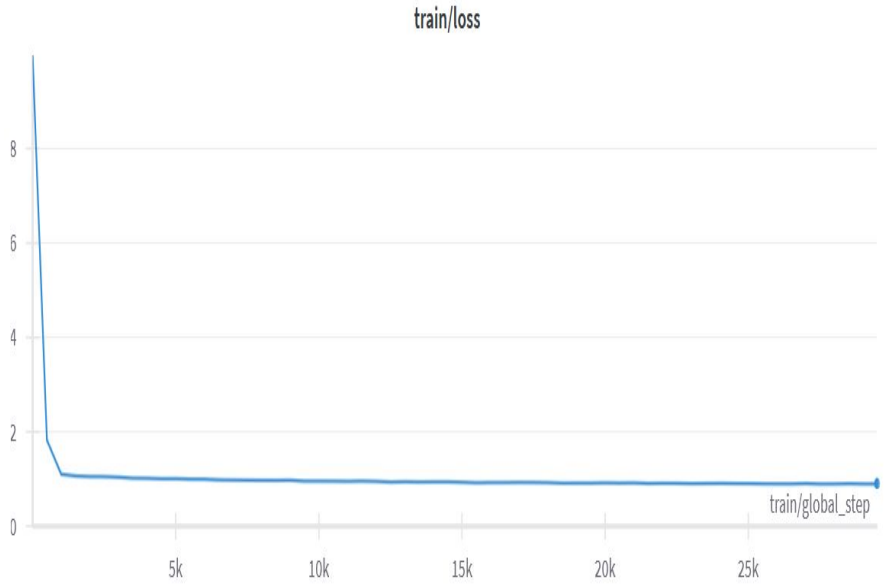


(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Bart

- There are 130M parameters in the model. We fine tune it on CNN / DailyMail Dataset

- 10 epochs,Some important fine tuning parameters are: learning_rate=2e-5, weight_decay=0.01,per_device_train_batch_size=8, and num_train_epochs=10



Data and Evaluation

Dataset: News Articles from CNN and the DailyMail

Training dataset: 287,113 samples

Validation: 13,368 samples

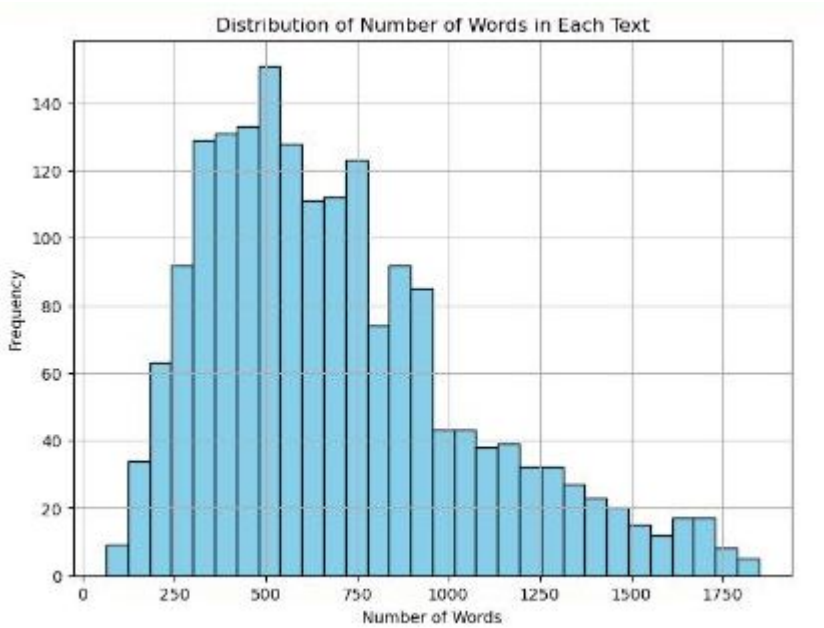
Test: 1,000 samples

Example:

{'id': '0054d6d30dbcad772e20b22771153a2a9cbeaf62',

'article': '(CNN) -- An American woman died aboard a cruise ship that
which 86 passengers previously fell ill, according to the state-run Braz
died aboard the MS Veendam, owned by cruise operator Holland Ame
doctors were investigating her death. The ship's doctors told police tha
hypertension, according the agency. The other passengers came down
the trip, the ship's doctors said. The Veendam left New York 36 days ag

'highlights': 'The elderly woman suffered from diabetes and hypertens
fallen ill on the ship, Agencia Brasil says .'}



Metric : Rouge

A commonly used evaluation metric for text summarization

ROUGE-N: overlap of n-gram between the system and reference summaries.

- ROUGE-1 based on the overlap of ***unigrams*** *(each word)* between generated summaries and ground truth
-
- ROUGE-2 based on the overlap of ***bigrams*** between generated summaries and ground truth.

ROUGE-L: Longest Common Subsequence (LCS) based statistics. Longest common subsequence problem takes into account sentence-level structure similarity naturally and identifies longest co-occurring in sequence n-grams automatically.

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

Results

Table 1: Results from different models

Model	Rouge 1	Rouge 2	Rouge L
Random	0.15	0.08	0.10
TF-IDF	0.24	0.01	0.13
TextRank	0.31	0.13	0.20
Pegasus	0.33	0.15	0.18
T5	0.38	0.12	0.23
Bart	0.25	0.12	0.21

Conclusion