

VNUHCM – University of Science
FACULTY OF INFORMATION TECHNOLOGY



FINAL PROJECT REPORT

COURSE: STATISTICAL LEARNING

CLASS: 21_22

Project Instructor:

Ngô Minh Nhật

Lê Long Quốc

Group members:

20120076 – Mai Vinh Hiển

21120070 – Nhan Hữu Hiếu

21120201 – Bùi Đình Bảo

WORK DISTRIBUTION TABLE

STUDENT ID	FULL NAME	TASKS
20120076	Mai Vinh Hiên	Do research on Transformer and relevant materials Pre-process dataset Train Optimize Write report
21120070	Nhan Hữu Hiếu	Do research on Transformer and relevant materials Find dataset
21120201	Bùi Đình Bảo	Do research on Transformer and relevant materials Pre-process dataset Set up model, config, tokenizer, metrics Train Fine-tune Set up web application

Contents

I. NLP Problem Chosen: Text Summarization	4
II. Model Used: Google's T5-Small	4
Text-to-Text Transfer Transformer (T5)	4
Training.....	4
Models in the series.....	4
The Transformer Architecture	5
Self-Attention Mechanism	5
Position-Wise Feed-Forward Networks.....	6
Positional Encoding	7
Encoder and Decoder.....	7
III. Dataset Used: CNN/Daily Mail Dataset.....	7
IV. Training.....	8
V. Web Application	10
References	10

I. NLP Problem Chosen: Text Summarization

Here, we chose text summarization because it's often the most common NLP task we encountered in our day-to-day lives. We college students, read a lot, and often from verbal sources. When we read, we often need to filter out ideas, which means we have to ignore unnecessary words. In order to save time and energy, this process should be automated by AI models. Hence, our project.

II. Model Used: Google's T5-Small

Text-to-Text Transfer Transformer (T5)

T5, or Text-to-Text Transfer Transformer, is a series of large language models developed by Google AI. When introduced in 2019, they're able to handle various text-based tasks, like: question answering, machine translation, text summarization, code generation.

T5 models use the Transformer architecture, using both encoder and decoder of Transformer, where the encoder processes the input text, and the decoder generates the output text.

What makes them special is their text-to-text framework. Unlike traditional models trained for specific tasks (translation, summarization), T5 treats all tasks as text-in, text-out problems. This allows for greater flexibility and adaptation.

Training

T5 models are pre-trained on the Colossal Clean Crawled Corpus (C4), containing text and code scraped from the internet. This pre-training process enables the models to learn general language understanding and generation abilities. T5 models can then be fine-tuned on specific downstream tasks, adapting their knowledge to perform well in various applications.

The T5 models were pretrained on many tasks, all in the format of [input text] -> [output text].

Models in the series

The T5 series encompasses several models with varying sizes and capabilities. These models are often distinguished by their parameter count, which indicates the complexity and potential capacity of the model. The original [paper](#)^[1] reported the following 5 models:

Model	Parameter	# Layers	d_{model}	d_{ff}	d_{kv}	# Heads
T5-Small	60M	6	512	2048	64	8
T5-Base	220M	12	768	3072	64	12
T5-Large	770M	24	1024	4096	64	16
T5-3B (XL)	3B	24	1024	16384	128	32
T5-11B (XXL)	11B	24	1024	65536	128	128

- **# Layers:** Number of layers in the encoder; also, number of layers in the decoder. They always have the same number of layers.
- **# Heads:** Number of attention heads in each attention block.
- **d_{model} :** Dimension of the embedding vectors.
- **d_{ff} :** Dimension of the feedforward network within each encoder and decoder layer.
- **d_{kv} :** Dimension of the key and value vectors used in the self-attention mechanism.

Here, we chose **T5-Small** for its quick training time.

The Transformer Architecture

Originally introduced in the paper [“Attention Is All You Need”](#)^[2] in 2017, Transformer is a groundbreaking architecture in the field of natural language processing (NLP) that relies entirely on attention mechanisms, dispensing with recurrence and convolution entirely. This model has since become the foundation for state-of-the-art approaches in a variety of NLP tasks.

The Transformer architecture is designed to handle sequential data with an innovative approach that emphasizes parallelization and long-range dependencies through self-attention mechanisms. Traditional models, such as Recurrent Neural Networks (RNNs) and their variants like Long Short-Term Memory (LSTM) networks, process sequences in a step-by-step manner, which often leads to inefficiencies in capturing dependencies over long sequences and limits parallelization. In contrast, the Transformer processes the entire sequence at once, enabling it to capture complex dependencies and significantly improve computational efficiency.

Self-Attention Mechanism

Central to the Transformer model is the self-attention mechanism, which allows the model to weigh the importance of different tokens in a sequence relative to each other. The self-attention

mechanism computes a representation of a sequence by relating each word to every other word in the sequence. This is achieved through the following steps:

1. **Input Representation:** Given an input sequence $x = (x_1, x_2, \dots, x_n)$, each token x_i is embedded into a continuous representation.
2. **Linear Transformations:** The input embeddings are linearly transformed to produce three distinct vectors for each token: the Query (Q), Key (K), and Value (V) vectors. These are obtained using learned weight matrices W^Q, W^K, W^V :

$$Q = XW^Q, K = XW^K, V = XW^V$$

Where X is the matrix of input embeddings.

3. **Scaled Dot-Product Attention:** The attention scores are calculated using the scaled dot-product of the Query and Key vectors. These scores are then normalized using a softmax function to obtain attention weights:

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

where d_k is the dimensionality of the Key vectors. The division by $\sqrt{d_k}$ helps to mitigate the effect of large dot-product values.

4. **Multi-Head Attention:** Instead of performing a single attention function, the Transformer employs multiple attention heads. Each head independently applies the attention mechanism to different linear projections of the input, and the results are concatenated and linearly transformed:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \dots, head_h)W^O$$

Where each attention head $head_i$ is computed as:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

And W^O is the output projection matrix.

Position-Wise Feed-Forward Networks

After the multi-head attention mechanism, the Transformer applies a position-wise feed-forward network to each position separately and identically. This network consists of two linear transformations with a ReLU activation in between:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Where W_1 and W_2 are weight matrices, b_1 and b_2 are biases.

Positional Encoding

Since the Transformer does not inherently model the sequential nature of the input data, it introduces positional encodings to inject information about the relative position of tokens in the sequence. These encodings are added to the input embeddings and are defined as:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

Where pos is the position and i is the dimension index.

Encoder and Decoder

The Transformer model consists of an encoder and a decoder, each composed of multiple layers. Each encoder layer contains two sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward network. Each decoder layer additionally includes a third sub-layer that performs multi-head attention over the encoder's output.

Each **encoder** layer can be described as:

$$EncoderLayer(x) = FFN(MultiHead(Q, K, V) + PositionalEncoding(x))$$

Where residual connections and layer normalization are applied around each sub-layer.

Each **decoder** follows a similar structure but includes an additional cross-attention mechanism to attend to the encoder's output:

$$\begin{aligned} DecoderLayer(x, enc_output) \\ = FFN(MultiHead(Q, K, V) + MultiHead(Q, K', V') + PositionalEncoding(x)) \end{aligned}$$

Where K' and V' come from the encoder output.

III. Dataset Used: CNN/Daily Mail Dataset

We chose [CNN/Daily Mail dataset](#)^[3] because of its popularity for training text summarization models, particularly those focused on abstractive summarization.

The CNN/Daily Mail dataset consists of news articles and their corresponding summaries. These summaries are in the form of bullet points that highlight the main points of each article. The dataset was created by scraping articles from the CNN and Daily Mail websites, making it a large and rich resource for training and evaluating text summarization models.

The dataset is structured into pairs of full-text articles and multi-sentence summaries. Each article-summary pair includes **Article text** and **Highlights** – a set of bullet points summarizing the key points of the article.

The dataset downloaded from HuggingFace has 286,817 training pairs, 13,368 validation pairs, and 11,487 test pairs. Its whole size is about 838 MB.

Here, we first tokenize the dataset, and then use only 10,000 training pairs, 1,000 validation pairs, and 1,000 test pairs for the sake of quick training.

First, we will **fetch the tokenizer**:

```
tokenizer = T5Tokenizer.from_pretrained("t5-small")
```

In order to handle data processing required for our model, we will **initialize a data collator**:

```
data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)
```

- DataCollatorForSeq2Seq: This utility class is responsible for dynamically padding the inputs and labels to the maximum length of the batch, ensuring that all sequences are the same length. It also handles the creation of decoder inputs and other necessary transformations for training Seq2Seq models.

Preprocessing function:

We will prepare inputs, tokenize, prepare labels with the function.

- We will iterate over each article (here, `example['article']`) and prepends “summarize: ” to each article, for the purpose of specifying the task for the model.
- Tokenization: the tokenizer is used to convert the input texts into token IDs, with a max length of 512 tokens, texts that are longer than 512 tokens are truncated.
- The summaries are tokenized with a max length of 150 tokens, truncation is applied when necessary.
- The tokenized summaries are assigned to the `model_input['labels']` field, which will be used as target labels during training.

IV. Training

We will use HuggingFace’s Transformers library to fine-tune our model, Sacrebleu to measure training metrics.

Pre-trained Configuration: We will use the default configuration provided by HuggingFace: <https://huggingface.co/google-t5/t5-small/blob/main/config.json>

And then **initialize the model**:

```
model = T5ForConditionalGeneration.from_pretrained("t5-small",
config=config)
```

Training arguments:

- We will set the learning rate to $3e-5$
- Set the evaluation only happens and the end of each epoch
- Batch size for training per device (e.g., per GPU) to 4, same goes for evaluation per device
- The number of epochs to only 5 (since we had limited time)
- Weight decay to 0.01 to prevent overfitting by applying a penalty to large weights
- The number of steps between model checkpoints to 10,000

We will also set the **compute metrics**:

- The predictions (preds) are decoded from token IDs back into text using the tokenizer, with special tokens (e.g., <pad>, <s>, </s>) skipped
- The labels are similarly decoded, but first, any padding tokens (-100, often used to mask out padding in the labels) are replaced with the tokenizer's pad token ID
- The SacreBLEU metric is computed using the cleaned predictions and references. The result is stored in a dictionary with the key **bleu**
- The length of each prediction (excluding padding tokens) is calculated, and the average length is added to the results
- All results are rounded to four decimal places

And finally we create our trainer and train our model:

```
trainer = Seq2SeqTrainer(model=model, args=training_args,
train_dataset=sub_train, eval_dataset=sub_test,
data_collator=data_collator, tokenizer=tokenizer,
compute_metrics=compute_metrics)

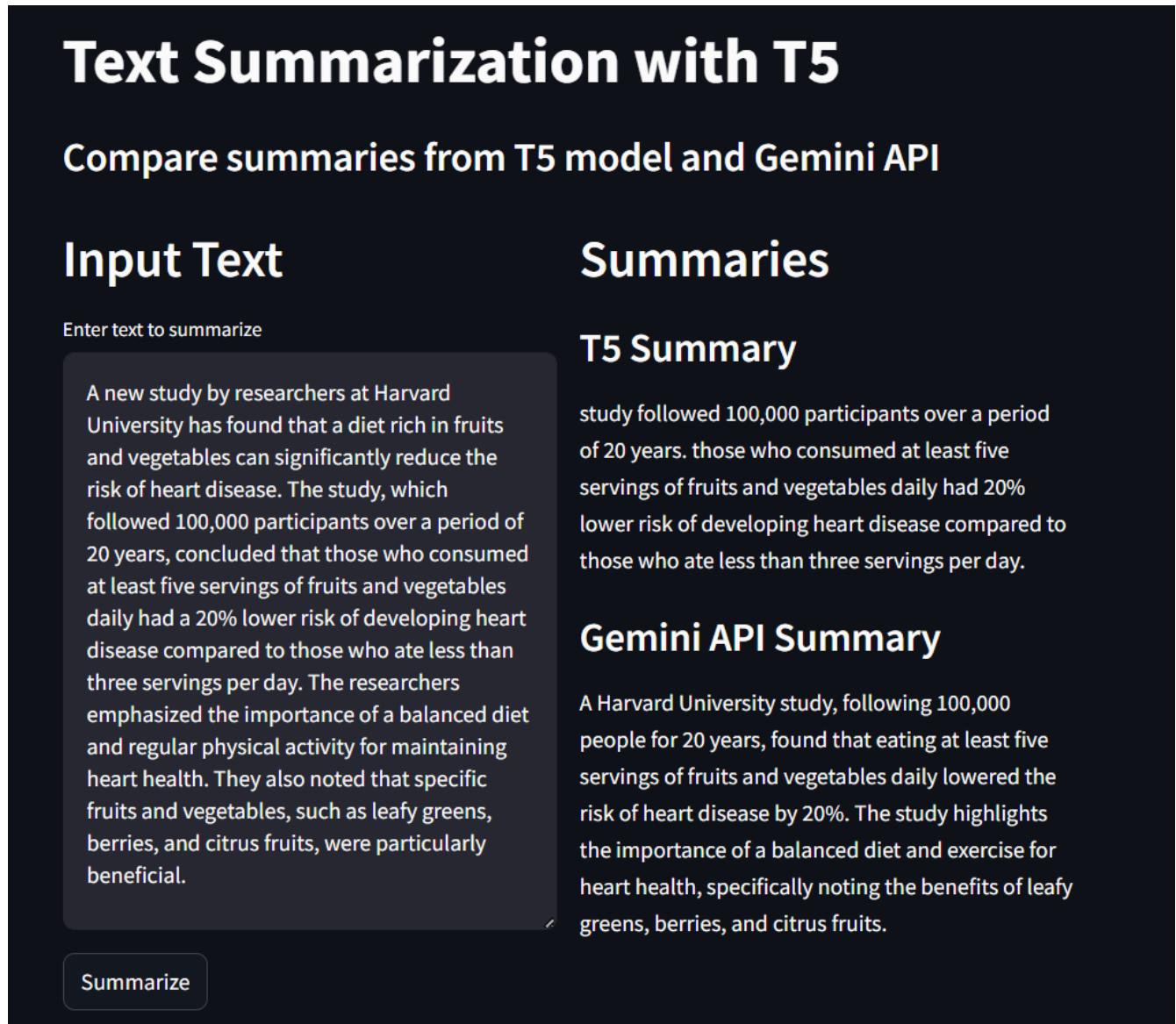
trainer.train()

metrics = trainer.evaluate()
print(metrics)
```

V. Web Application

We use Streamlit library for an easy interface. Code can be found in the notebook.

Inference result:



Text Summarization with T5

Compare summaries from T5 model and Gemini API

Input Text

Enter text to summarize

A new study by researchers at Harvard University has found that a diet rich in fruits and vegetables can significantly reduce the risk of heart disease. The study, which followed 100,000 participants over a period of 20 years, concluded that those who consumed at least five servings of fruits and vegetables daily had a 20% lower risk of developing heart disease compared to those who ate less than three servings per day. The researchers emphasized the importance of a balanced diet and regular physical activity for maintaining heart health. They also noted that specific fruits and vegetables, such as leafy greens, berries, and citrus fruits, were particularly beneficial.

Summarize

Summaries

T5 Summary

study followed 100,000 participants over a period of 20 years. those who consumed at least five servings of fruits and vegetables daily had 20% lower risk of developing heart disease compared to those who ate less than three servings per day.

Gemini API Summary

A Harvard University study, following 100,000 people for 20 years, found that eating at least five servings of fruits and vegetables daily lowered the risk of heart disease by 20%. The study highlights the importance of a balanced diet and exercise for heart health, specifically noting the benefits of leafy greens, berries, and citrus fruits.

References

Hugging Face documentation: <https://huggingface.co/docs/hub/transformers>

[1] T5 paper - Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer: <https://jmlr.org/papers/v21/20-074.html> and

<https://research.google/blog/exploring-transfer-learning-with-t5-the-text-to-text-transfer-transformer/>

[2] Transformer original paper - Attention Is All You Need: <https://arxiv.org/abs/1706.03762>

[3] CNN/Daily Mail dataset:

https://huggingface.co/datasets/abisee/cnn_dailymail/tree/main/3.0.0 and

<https://paperswithcode.com/dataset/cnn-daily-mail-1>