NLP Mandate-4

ABSTRACTIVE SUMMARIZATION

Hardeep Singh Arora (MT2022047)

Problem Statement:

Given a set tweet of similar topic, generate a abstractive summarisation of the tweets. Do not just string the tweets together to form the summary. The summary will need to paraphrase and/or say more than what is directly said in the tweets. Propose a rubric to evaluate the accuracy of your summarization.

Data Acquisition:

I have used **snscrape**, for scraping the data from the twitter. Snscrape is a scraper for social networking services (SNS). This python library scrapes things like user, profiles, hashtags, or searches and returns the discovered items, example, the relevant posts.

snscrape Python library that can be used to scrape tweets through Twitter's API without any restrictions or request limits. Moreover, you don't even need a Twitter developer account to scrape tweets when you use snscrape.

Currently the it supports following services for twitter:

users, user profiles, hashtags, searches (live tweets, top tweets, and users), tweets (single or surrounding thread), list posts, communities, and trends.

Text Preprocessing:

Given below are the data cleaning steps I followed to clean my scraped data from the internet.

1. **Removing punctuations:**
   punctuations = "!”$%&’()*+-/:;<=>[]^_`{|}~•@#"
   x.translate(str.maketrans('','',punctuations))

   The above regex (regular expression) was used to remove all the punctuatuions from the text.

2. **Removing URLs:**
   re.sub('http://\S+|https://\S+', '', x)

   The above regex was used to remove all the urls from the text starting with “http” or “https”.
3. **Removing hashtags:**
   re.sub('#[A-Za-z0-9]','',x)

   All the text starting with “#” tags were removed with the help of this regex.

4. **Removing emojis:**
   emoji.demojize(x['text'][i])
   To remove the emojis from the text I have used the emoji python library.
   The main purpose of this package is converting Unicode emoji to emoji names and vice versa with emojize() and demojize().

5. **Removing escape sequences:**
   All the escape sequences like “\n, “\t” etc are also required to be removed as part of data cleaning step.

6. **Replaced short-hands:**
   x.replace("&", "and")
   Replaced short-hands like “&” ampersand into “and” etc.
   After applying all the above preprocessing steps , I have divided the tweets into groups of ten, which I later fed into the model for generating summary.

Some other pre-processing steps which are very important in natural language processing are tokenisation, stemming and lemmatisation.

Tokenisation:
Tokenization is the first step in any NLP pipeline. It has an important effect on the rest of your pipeline. A **tokenizer** breaks unstructured data and natural language text into chunks of information that can be considered as discrete elements. The token occurrences in a document can be used directly as a vector representing that document.
This immediately turns an unstructured string (text document) into a numerical data structure suitable for machine learning.

Stemming:
Stemming is the process of reducing a word to its stem that affixes to suffixes and prefixes or to the roots of words known as "lemmas". **Stemming** is important in natural language understanding (NLU) and natural language processing (NLP).

Lemmatization:
Lemmatization is one of the most common text pre-processing techniques used in Natural Language Processing (NLP) and machine learning in general. Both in stemming and in lemmatization, we try to reduce a given word to its root word. The root word is called a stem in the stemming process, and it is called a lemma in the lemmatization process.

The “**main_df”** dataframe consists of 1041 rows of summaries.

Generating Summaries:-

I have used Chat-GPT API for generating summaries.

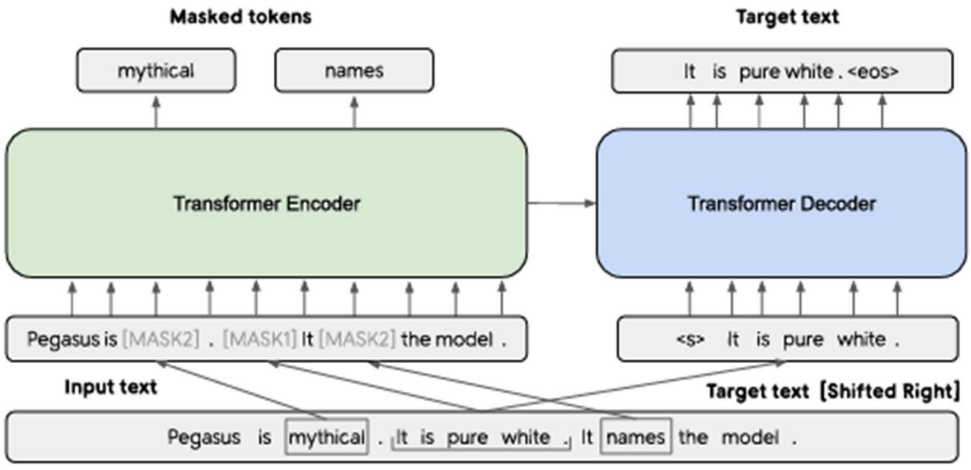
```
r = []
for ind in tqdm(range(len(df)),desc="Loading..."):
    prompt = "Summarize this set of tweets:"+df['content'][ind+546]
    completion = openai.Completion.create(
    engine = "text-davinci-003",
    prompt=prompt,
    max_tokens=300,
    n=1,
    stop=None,
    temperature=0.5)
    r.append(completion.choices[0].text)
```

Model Used:

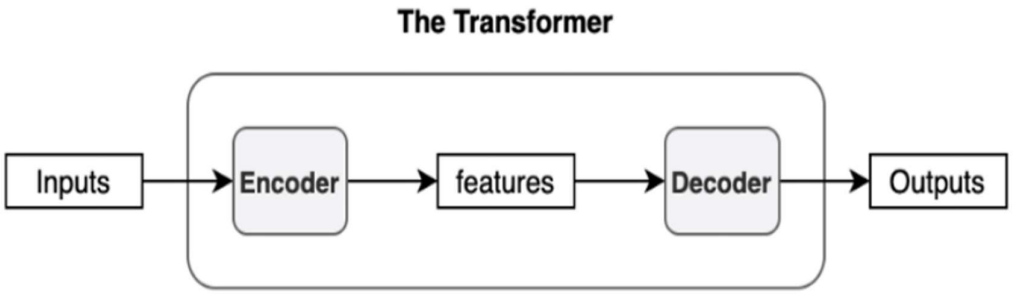
The Pegasus model was proposed by Jingqing Zhang, Yao Zhao, Mohammad Saleh and Peter J. Liu on Dec 18, 2019.

PEGASUS stands for “Pre-training with Extracted Gap-sentences for Abstractive Summarization”

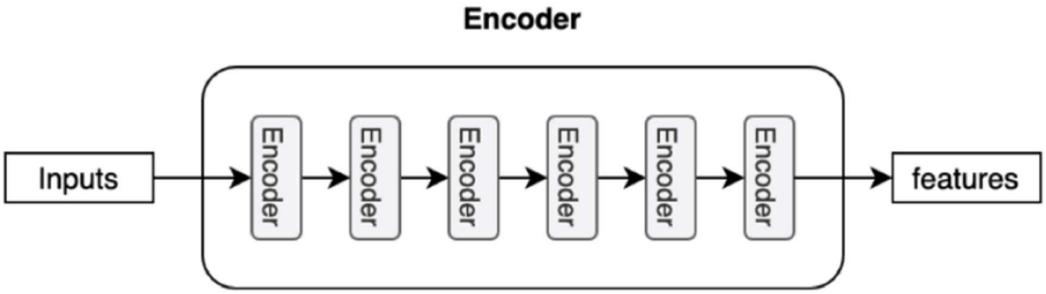
It is the latest state-of-the-art model for abstractive summarization.



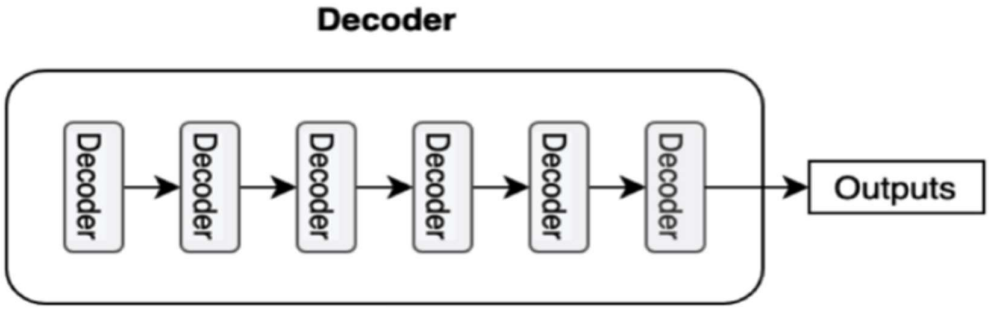
The base architecture of **PEGASUS** is a standard **Transformer** encoder-decoder. The transformer uses an encoder-decoder architecture. The encoder extracts features from an input sentence, and the decoder uses the features to produce an output sentence (translation).



NLP's Transformer is a new architecture that aims to solve tasks sequence-to-sequence while easily handling long-distance dependencies. Computing the input and output representations without using sequence-aligned RNNs or convolutions and it relies entirely on self-attention.



Encoder: The encoder is responsible for stepping through the input time steps and encoding the entire sequence into a fixed-length vector called a context vector.

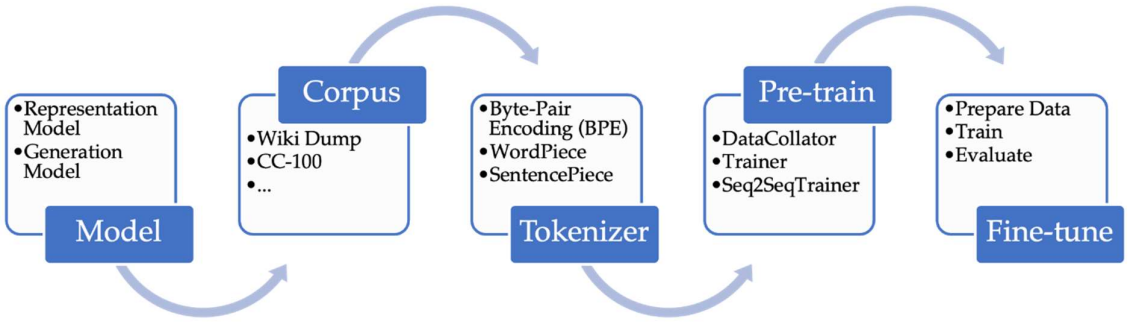


Decoder: The decoder is responsible for stepping through the output time steps while reading from the context vector.

Both **Encoder** and **Decoder** consists of multiple blocks.

Fine-Tuning models:

In NLP fine-tuning a model refers to the procedure of re-training a pre-trained language model using our own collected data. As a result of the fine-tuning procedure, the weights of the original model are updated to account for the characteristics of the domain data and the task you are interested in.



The main benefits of using a pre-trained models are that it reduces the computation costs, carbon footprint and also allows us to use state-of-the-art models without having to train a model like that from scratch. Hugging Face Transformers provides us access to many such pre-trained models for a wide range of tasks.

Evaluation metric used:-

Rouge :-

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is essentially a set of metrics for evaluating automatic summarization of texts as well as machine translations.

It works by comparing an **automatically produced summary** or **translation** against a set of **reference summaries** (typically human-produced).

Model-1:- google/pegasus-cnn_dailymail

```
from transformers import TrainingArguments, Trainer

trainer_args = TrainingArguments(
    output_dir='pegasus/cnn-daily', num_train_epochs=20, warmup_steps=500,
    per_device_train_batch_size=1, per_device_eval_batch_size=1,
    weight_decay=0.01, logging_steps=10,
    evaluation_strategy='steps', eval_steps=500, save_steps=1e6,
    gradient_accumulation_steps=16
)
```

```
trainer = Trainer(model=model_pegasus, args=trainer_args,
                  tokenizer=tokenizer, data_collator=seq2seq_data_collator,
                  train_dataset=train_pt,
                  eval_dataset=test_pt)
```

+ Code + Markdown

```
trainer.train()
```

Rouge Score for Pegasus Model:-

	rouge1	rouge2	rougeL	rougeLsum
pegasus	0.364958	0.162032	0.253622	0.253341

Model 2:- Simple T5

```
model.train(train_df=train_df,
            eval_df=test_df,
            source_max_token_len=1024,
            target_max_token_len=150,
            batch_size=4, max_epochs=5, use_gpu=True)
```

Rouge Score for Simple T5 model:-

-

rouge1	rouge2	rougeL	rougeLsum
0.423908	0.204665	0.308492	0.308488

References:

- PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization.(Authors:- Jingqing Zhang, Yao Zhao, Mohammad Saleh, Peter J. Liu) https://doi.org/10.48550/arXiv.1912.08777
- https://blog.knoldus.com/what-are-transformers-in-nlp-and-its-advantages/
- https://towardsdatascience.com/fine-tuning-for-domain-adaptation-in-nlp-c47def356fd6
- https://huggingface.co/docs/transformers/model_doc/pegasus