Sequence-to-sequence (seq2seq) Part II

In Part I, we have explored Encoder-Decoder model, Transformer (from scratch) to tackle translation task. In notebook, we will explore one of the most useful transformer variants on a different task, text summarization task. We will learn T5 model which is a transformer based model and we will also learn how to fine-tune pretrained models.

In [13]:

```
!pip3 install datasets
!pip3 install rich
!pip3 install SentencePiece
!pip3 install rouge_score
!pip3 install transformers

# Importing libraries
import os
import numpy as np
import pandas as pd
import torch
import torch.nn.functional as F
from torch.utils.data import DataLoader
import os

# rich: for a better display on terminal
from rich.table import Column, Table
from rich import box
from rich.console import Console

# Importing the T5 modules from huggingface/transformers
from transformers import PegasusTokenizer, PegasusForConditionalGeneration, T5Tokeni

import nltk

from datasets import load_dataset
from datasets import load_metric

# #uncomment this if you are not using puffer
# import os
# os.environ['http_proxy'] = 'http://192.41.170.23:3128'
# os.environ['https_proxy'] = 'http://192.41.170.23:3128'

# !pip3 install datasets
# !pip3 install rich
# !pip3 install SentencePiece
# !pip3 install rouge_score
```

```
Requirement already satisfied: datasets in /usr/local/lib/python3.7/di
st-packages (1.18.3)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist
-packages (from datasets) (1.3.5)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/pyth
on3.7/dist-packages (from datasets) (2.23.0)
Requirement already satisfied: xxhash in /usr/local/lib/python3.7/dist
-packages (from datasets) (2.0.2)
Requirement already satisfied: fsspec[http]>=2021.05.0 in /usr/local/l
ib/python3.7/dist-packages (from datasets) (2022.2.0)
Requirement already satisfied: importlib-metadata in /usr/local/lib/py
thon3.7/dist-packages (from datasets) (4.11.1)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.
7/dist-packages (from datasets) (0.70.12.2)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.
7/dist-packages (from datasets) (4.62.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/d
ist-packages (from datasets) (21.3)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.
7/dist-packages (from datasets) (1.21.5)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.7/dis
```

```
t-packages (from datasets) (3.8.1)
Requirement already satisfied: dill in /usr/local/lib/python3.7/dist-p
ackages (from datasets) (0.3.4)
Requirement already satisfied: pyarrow!=4.0.0,>=3.0.0 in /usr/local/li
b/python3.7/dist-packages (from datasets) (6.0.1)
Requirement already satisfied: huggingface-hub<1.0.0,>=0.1.0 in /usr/l
ocal/lib/python3.7/dist-packages (from datasets) (0.4.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/loca
l/lib/python3.7/dist-packages (from huggingface-hub<1.0.0,>=0.1.0->dat
asets) (3.10.0.2)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist
-packages (from huggingface-hub<1.0.0,>=0.1.0->datasets) (6.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/di
st-packages (from huggingface-hub<1.0.0,>=0.1.0->datasets) (3.6.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/
lib/python3.7/dist-packages (from packaging->datasets) (3.0.7)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->data
sets) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/py
thon3.7/dist-packages (from requests>=2.19.0->datasets) (2021.10.8)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/pyt
hon3.7/dist-packages (from requests>=2.19.0->datasets) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.
7/dist-packages (from requests>=2.19.0->datasets) (2.10)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/loc
al/lib/python3.7/dist-packages (from aiohttp->datasets) (4.0.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python
3.7/dist-packages (from aiohttp->datasets) (21.4.0)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python
3.7/dist-packages (from aiohttp->datasets) (1.7.2)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/pyth
on3.7/dist-packages (from aiohttp->datasets) (1.2.0)
Requirement already satisfied: asynctest==0.13.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from aiohttp->datasets) (0.13.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/pyt
hon3.7/dist-packages (from aiohttp->datasets) (1.3.0)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/lo
cal/lib/python3.7/dist-packages (from aiohttp->datasets) (2.0.12)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/p
ython3.7/dist-packages (from aiohttp->datasets) (6.0.2)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/d
ist-packages (from importlib-metadata->datasets) (3.7.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.
7/dist-packages (from pandas->datasets) (2018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/li
b/python3.7/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/di
st-packages (from python-dateutil>=2.7.3->pandas->datasets) (1.15.0)
Requirement already satisfied: rich in /usr/local/lib/python3.7/dist-p
ackages (11.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.6.0 in /usr/local/li
b/python3.7/dist-packages (from rich) (2.6.1)
Requirement already satisfied: colorama<0.5.0,>=0.4.0 in /usr/local/li
b/python3.7/dist-packages (from rich) (0.4.4)
Requirement already satisfied: typing-extensions<5.0,>=3.7.4 in /usr/l
ocal/lib/python3.7/dist-packages (from rich) (3.10.0.2)
Requirement already satisfied: commonmark<0.10.0,>=0.9.0 in /usr/loca
l/lib/python3.7/dist-packages (from rich) (0.9.1)
Requirement already satisfied: SentencePiece in /usr/local/lib/python
3.7/dist-packages (0.1.96)
```

```
Requirement already satisfied: rouge_score in /usr/local/lib/python3.
7/dist-packages (0.0.4)
Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python3.
7/dist-packages (from rouge_score) (1.15.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-
packages (from rouge_score) (1.21.5)
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-p
ackages (from rouge_score) (3.2.5)
Requirement already satisfied: absl-py in /usr/local/lib/python3.7/dis
t-packages (from rouge_score) (1.0.0)
Requirement already satisfied: transformers in /usr/local/lib/python3.
7/dist-packages (4.16.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/pyt
hon3.7/dist-packages (from transformers) (2019.12.20)
Requirement already satisfied: tokenizers!=0.11.3,>=0.10.1 in /usr/loc
al/lib/python3.7/dist-packages (from transformers) (0.11.5)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/di
st-packages (from transformers) (3.6.0)
Requirement already satisfied: sacremoses in /usr/local/lib/python3.7/
dist-packages (from transformers) (0.0.47)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.
7/dist-packages (from transformers) (6.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/
dist-packages (from transformers) (4.62.3)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/pytho
n3.7/dist-packages (from transformers) (21.3)
Requirement already satisfied: importlib-metadata in /usr/local/lib/py
thon3.7/dist-packages (from transformers) (4.11.1)
Requirement already satisfied: requests in /usr/local/lib/python3.7/di
st-packages (from transformers) (2.23.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.1.0 in /usr/loc
al/lib/python3.7/dist-packages (from transformers) (0.4.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.
7/dist-packages (from transformers) (1.21.5)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/loca
l/lib/python3.7/dist-packages (from huggingface-hub<1.0,>=0.1.0->trans
formers) (3.10.0.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/
lib/python3.7/dist-packages (from packaging>=20.0->transformers) (3.0.
7)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/d
ist-packages (from importlib-metadata->transformers) (3.7.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests->transformer
s) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.
7/dist-packages (from requests->transformers) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/pyt
hon3.7/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/py
thon3.7/dist-packages (from requests->transformers) (2021.10.8)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-pa
ckages (from sacremoses->transformers) (1.15.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist
-packages (from sacremoses->transformers) (1.1.0)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-
packages (from sacremoses->transformers) (7.1.2)
```

What is a Summary?

A summary is a condensed version of an original text, usually a full article or book. Summaries are usually around a paragraph long, and may even be a few paragraphs long depending on the length of the work being condensed.

Summaries are used in variety of situations. For example, you might want to summarize only the main points of a meeting with a co-worker because you're running late for another meeting. Or, let's say you want to introduce a complex design idea. You could begin by summarizing what your design would accomplish, to give key people an overall sense of your plan without overwhelming them. Students might summarize an article for a class, or when preparing and writing research papers, annotated bibliographies and essays. Abstracts and legal brief are also types of summaries.

The "summary" itself has some varieties and approaches.

Types of summary

- **Indicative summary**
  It looks like a summary of the book. This summary describes what kinds of the story, but not tell all of the stories especially its ends (so indicative summary has only partial information).
- **Informative summary**
  In contrast to the indicative summary, the informative summary includes full information of the document.
- **Keyword summary**
  Not the text, but the words or phrases from the input document.
- **Headline summary**
  Only one line summary.

Basic Approach

There are mainly two ways to make the summary. Extractive and Abstractive. In short, in extractive summarization sentences are chosen from the article(s) given as input, whereas in abstractive summarization sentences may be generated or a new representation of the article(s) may be output.

Extractive

- Select relevant **phrases of the input document** and **concatenate** them to form a summary (like "copy-and-paste"). The network calculates the most important sentences from the article and gets them together to provide the most meaningful information from the article.
  - Pros: They are quite robust since they use existing natural-language phrases that are taken straight from the input.
  - Cons: But they lack in flexibility since they cannot use novel words or connectors. They also cannot paraphrase like people sometimes do (less fluent).

Abstractive

- Generate a summary that keeps original intent. It's just like humans do. The network creates new sentences to encapsulate maximum gist of the article and generates that as output. The sentences in the summary may or may not be contained in the article.
  - Pros: They can use words that were not in the original input. It enables to make more fluent and natural summaries.
  - Cons: But it is also a much harder problem as you now require the model to generate coherent phrases and connectors.

Extractive & Abstractive is not conflicting ways. You can use both to generate the summary. And there are a way collaborate with human.

- Aided Summarization
  - Combines automatic methods with human input.
  - Computer suggests important information from the document, and the human decide to use it or not. It uses information retrieval, and text mining way.

The beginning of the abstractive summarization, Banko et al. (2000) (http://www.anthology.aclweb.org/P/P00/P00-1041.pdf) suggest to use machine translatation model to abstractive summarization model. As like the machine translation model converts a source language text to a target one, the summarization system converts a source document to a target summary.

Nowadays, **encoder-decoder** model that is one of the neural network models is mainly used in machine translation. So this model is also widely used in abstractive summarization model. The summarization model that used encoder-decoder model first (http://www.aclweb.org/anthology/N16-1012) achieved state-of-the-art on the two sentence-level summarization dataset, DUC-2004 and Gigaword.

Encoder-Decoder Model

The encoder-decoder model is composed of encoder and decoder like its name. The encoder converts an input document to a latent representation (vector), and the decoder generates a summary by using it.

One of the well-known Encoder-Decoder models is T5 which we will be using in this lecture. T5 is one of the most recent and novel transformers model. T5 in many ways is one of its kind transformers architecture that not only gives state of the art results in many NLP tasks, but also has a very radical approach to NLP tasks. Text-2-Text - According to the graphic taken from the T5 paper. All NLP tasks are converted to a text-to-text problem. Tasks such as translation, classification, summarization and question answering, all of them are treated as a text-to-text conversion problem, rather than seen as separate unique problem statements. Unified approach for NLP Deep Learning - Since the task is reflected purely in the text input and output, you can use the same model, objective, training procedure, and decoding process to ANY task. Above framework can be used for any task - show Q&A, summarization, etc.

We will be taking inputs from the T5 paper to prepare our dataset prior to fine tuning and training.

Preparing the Dataset for data processing: Class

We will start with creation of Dataset class - This defines how the text is pre-processed before sending it to the neural network. This dataset will be used the the Dataloader method that will feed the data in batches to the neural network for suitable training and processing. The Dataloader and Dataset will be used inside the Trainer().

CustomDataset Dataset Class

- This class is defined to accept the data as input and generate tokenized output that is used by the T5 model for training.
- add_prefix() inside Dataset Class is to add a prefix "summarize: ", this is important when you use T5
- We are using the T5 tokenizer to tokenize the data in the text and ctext column of the dataframe.
- The tokenizer uses the batch_encode_plus method to perform tokenization and generate the necessary outputs, namely: source_id, source_mask from the actual text and target_id and target_mask from the summary text. To read further into the tokenizer, refer to this document

- The CustomDataset class is used to create 1 dataset for each call, hench is called twice in the Trainer() for training and for validation.
- Training Dataset is used to fine tune the model: 80% of the original data
- Validation Dataset is used to evaluate the performance of the model. The model has not seen this data during training.

In [14]:

```
from torch.utils.data import Dataset
import torch
import pandas as pd

class Dataset(Dataset):
    """
    Creating a custom dataset for reading the dataset and
    loading it into the dataloader to pass it to the
    neural network for finetuning the model

    """

    def __init__(
        self, data, tokenizer, model_name, source_len, target_len, source_text, targ
    ):
        """
        Initializes a Dataset class

        Args:yo
            data: Input data
            tokenizer (transformers.tokenizer): Transformers tokenizer
            source_len (int): Max length of source text
            target_len (int): Max length of target text
            source_text (str): column name of source text
            target_text (str): column name of target text
        """
        self.tokenizer = tokenizer
        self.data = data
        self.source_len = source_len
        self.summ_len = target_len

        # Xsum contains about 200000+ samples, so lets just take a small portion
        self.source_text = self.data[source_text][:1000]
        if "t5" in model_name:
            self.source_text = self.add_prefix(self.source_text)
        self.target_text = self.data[target_text][1:1000]

    def __len__(self):
        """returns the length of data"""

        return len(self.target_text)

    def __getitem__(self, index):
        """return the input ids, attention masks and target ids"""

        source_text = str(self.source_text[index])
        target_text = str(self.target_text[index])

        # cleaning data so as to ensure data is in string type
        source_text = " ".join(source_text.split())
        target_text = " ".join(target_text.split())

        source = self.tokenizer.batch_encode_plus(
            [source_text],
            max_length=self.source_len,
            pad_to_max_length=True,
            truncation=True,
            padding="max_length",
            return_tensors="pt",
```

```
        )

        target = self.tokenizer.batch_encode_plus(
            [target_text],
            max_length=self.summ_len,
            pad_to_max_length=True,
            truncation=True,
            padding="max_length",
            return_tensors="pt",
        )

        source_ids = source["input_ids"].squeeze()
        source_mask = source["attention_mask"].squeeze()
        target_ids = target["input_ids"].squeeze()
        target_mask = target["attention_mask"].squeeze()

        return {
            "source_ids": source_ids.to(dtype=torch.long),
            "source_mask": source_mask.to(dtype=torch.long),
            "target_ids": target_ids.to(dtype=torch.long),
            "target_ids_y": target_ids.to(dtype=torch.long),
        }

    def add_prefix(self, examples):
        prefix = "summarize: "
        inputs = [prefix + doc for doc in examples]
        return inputs
```

Fine Tuning the Model: Function

Here we define a training function that trains the model on the training dataset created above, specified number of times (EPOCH), An epoch defines how many times the complete data will be passed through the network.

This function is called in the Trainer()

Following events happen in this function to fine tune the neural network:

The epoch, tokenizer, model, device details, testing_ dataloader optimizer and scheduler are passed to the train () when its called from the Trainer() The dataloader passes data to the model based on the batch size. language_model_labels are calculated from the target_ids also, source_id and attention_mask are extracted. The model outputs first element gives the loss for the forward pass. Loss value is used to optimize the weights of the neurons in the network. After every 10 step the loss value is logged in the wandb service. This log is then used to generate graphs for analysis.

In [15]:

```
def train(epoch, tokenizer, model, device, loader, optimizer, scheduler):

    """
    Function to be called for training with the parameters passed from main functior

    """

    model.train()
    losses = 0
    for _, data in enumerate(loader, 0):
        y = data["target_ids"].to(device, dtype=torch.long)
        y_ids = y[:, :-1].contiguous()
        lm_labels = y[:, 1:].clone().detach()
        lm_labels[y[:, 1:] == tokenizer.pad_token_id] = -100
        ids = data["source_ids"].to(device, dtype=torch.long)
        mask = data["source_mask"].to(device, dtype=torch.long)

        outputs = model(
            input_ids=ids,
            attention_mask=mask,
            decoder_input_ids=y_ids,
            labels=lm_labels,
        )
        loss = outputs[0]
        if _ % 10 == 0:
            print("STEP: ", _,"/",len(loader))
            training_logger.add_row(str(epoch), str(f'{_}/{len(loader)}'), str(loss)
            console.print(training_logger)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        losses += loss.detach()

    scheduler.step()
    losses = losses/len(loader)
    return losses
```

Validating the Model Performance: Function

During the validation stage we pass the unseen data(Testing Dataset), trained model, tokenizer and device details to the function to perform the validation run. This step generates new summary for dataset that it has not seen during the training session.

This function is called in the Trainer()

This unseen data is the 20% of the data which was seperated during the Dataset creation stage. During the validation stage the weights of the model are not updated. We use the generate method for generating new text for the summary.

It depends on the Beam-Search coding method developed for sequence generation for models with LM head.

The generated text and originally summary are decoded from tokens to text and returned to the main()

In [16]:

```
def validate(epoch, tokenizer, model, device, loader):

    """
    Function to evaluate model for predictions

    """
    model.eval()
    predictions = []
    actuals = []
    with torch.no_grad():
        for _, data in enumerate(loader, 0):
            y = data['target_ids'].to(device, dtype = torch.long)
            ids = data['source_ids'].to(device, dtype = torch.long)
            mask = data['source_mask'].to(device, dtype = torch.long)

            generated_ids = model.generate(
                  input_ids = ids,
                  attention_mask = mask,
                  max_length=150,
                  num_beams=2,
                  repetition_penalty=2.5,
                  length_penalty=1.0,
                  early_stopping=True
                  )
            preds = [tokenizer.decode(g, skip_special_tokens=True, clean_up_tokeniza
            target = [tokenizer.decode(t, skip_special_tokens=True, clean_up_tokeniz
            if _ % 10==0:
                console.print(f'Completed {_}')

            predictions.extend(preds)
            actuals.extend(target)

    return predictions, actuals
```

Trainer: Function

In Trainer Function dataset, source text, target text, model parameters, output directory and device are passed into.

Inside this function the tokenizer and the pretrained weights of the defined model are downloaded. Moreover our customdataset class is also called here, loader parameters are defined, loaders are created, optimizer is defined and most importanlty, training loop is called.

After each time the train() is called, validate() is called after. One thing that is still missing is the performance metric.

In the Seq2Seq Part I, we have learnt to implement `BLEU score` , and other evaluation methods were asked to be explored. One of the commonly used metric to evaluate a summarization task is called `ROUGE score` .

ROUGE Score

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). Let's say that we have the following system and reference summaries:

System Summary (what the machine produced):

```
the cat was found under the bed
```

Reference Summary (gold standard — usually by humans):

```
the cat was under the bed
```

If we consider just the individual words, the number of overlapping words between the system summary and reference summary is 6. This, however, does not tell you much as a metric. To get a good quantitative value, we can actually compute the precision and recall using the overlap.

Simply put, recall (in the context of ROUGE) refers to how much of the reference summary the system summary is recovering or capturing. If we are just considering the individual words, it can be computed as:

$$\left(\frac{number\ of\ overlapping\ words'}{total\ words\ in\ the\ reference\ summary}\right)$$

$$\text{Recall} = \left(\frac{6}{6}\right) = 1.0$$

This means that all the words in the reference summary have been captured by the system summary, which indeed is the case for this example.

This looks really good for a text summarization system. But it does not tell you the other side of the story. A machine generated summary (system summary) can be extremely long, capturing all words in the reference summary. But, many of the words in the system summary may be useless, making the summary unnecessarily verbose.

This is where precision comes into play. In terms of precision, what you are essentially measuring is, how much of the system summary was in fact relevant or needed? Precision is measured as:

$$\left(\frac{number\ of\ overlapping\ words'}{total\ words\ in\ the\ system\ summary}\right)$$

$$\text{Precision} = \left(\frac{6}{7}\right) = 0.86$$

Now, let's say System Summary 2:

```
the tiny little cat was found under the big funny bed
```

The Precision now becomes:

$$\text{Precision} = \left(\frac{6}{11}\right) = 0.55$$

Now, this doesn't look so good, does it? That is because we have quite a few unnecessary words in the summary. The precision aspect becomes really crucial when you are trying to generate summaries that are concise in nature. Therefore, it is always best to compute both the precision and recall and then report the F-Measure.

If your summaries are in some way forced to be concise through some constraints, then you could consider using just the recall, since precision is of less concern in this scenario.

ROUGE-N, ROUGE-S, and ROUGE-L can be thought of as the granularity of texts being compared between the system summaries and reference summaries.

- ROUGE-N — measures unigram, bigram, trigram and higher order n-gram overlap

- ROUGE-L — measures longest matching sequence of words using LCS. An advantage of using LCS is that it does not require consecutive matches but in-sequence matches that reflect sentence level word order. Since it automatically includes longest in-sequence common n-grams, you don't need a predefined n-gram length.
- ROUGE-S — Is any pair of words in a sentence in order, allowing for arbitrary gaps. This can also be called skip-gram concurrence. For example, skip-bigram measures the overlap of word pairs that can have a maximum of two gaps in between words. As an example, for the phrase "cat in the hat" the skip-bigrams would be "cat in, cat the, cat hat, in the, in hat, the hat".

Q1. Please implement Rouge Score as our evaluation metric as a function that is called the for-loop (3 pt)

Hint: similar to BLEU

In [17]:

```
#!pip install ignite
```

In [18]:

```
#!pip3 install pytorch-ignite
```

In [19]:

```
from rouge_score import rouge_scorer

def compute_metrics(predictions, actuals, tokenizer):
    scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=Tr
    rouge = scorer.score(" ". join(predictions), " ". join(actuals))

    # <your code here>
    return rouge
```

In [20]:

```
Trainer(
dataset, source_text, target_text, model_params, output_dir="./outputs/", device = "


"""
trainer

"""

losses = []

# Set random seeds and deterministic pytorch for reproducibility
torch.manual_seed(model_params["SEED"])  # pytorch random seed
np.random.seed(model_params["SEED"])  # numpy random seed
torch.backends.cudnn.deterministic = True

# logging
console.log(f"""[Model]: Loading {model_params["MODEL"]}...\n""")

# Defining the model. We are using t5-base model and added a Language model layer on
# tokenzier for encoding the text
# Further this model is sent to device (GPU/TPU) for using the hardware.
if "bart" in model_params["MODEL"]:
    tokenizer = BartTokenizer.from_pretrained(f'facebook/{model_params["MODEL"]}')
    model = BartForConditionalGeneration.from_pretrained(f'facebook/{model_params["M
elif "t5" in model_params["MODEL"]:
    tokenizer = T5Tokenizer.from_pretrained(model_params["MODEL"])
    model = T5ForConditionalGeneration.from_pretrained(model_params["MODEL"])
elif "pegasus" in model_params["MODEL"]:
    tokenizer = PegasusTokenizer.from_pretrained(f'google/{model_params["MODEL"]}')
    model = PegasusForConditionalGeneration.from_pretrained(f'google/{model_params["
else:
    raise ValueError("Undefined model")

model = model.to(device)

# logging
console.log(f"[Data]: Reading data...\n")

# Creation of Dataset and Dataloader
train_dataset = dataset["train"]
val_dataset = dataset["validation"]

console.print(f"FULL Dataset: {dataset.shape}")
console.print(f"TRAIN Dataset: {train_dataset.shape}")
console.print(f"TEST Dataset: {val_dataset.shape}\n")

del dataset

# Creating the Training and Validation dataset for further creation of Dataloader
training_set = Dataset(
    train_dataset,
    tokenizer,
    model_params["MODEL"],
    model_params["MAX_SOURCE_TEXT_LENGTH"],
    model_params["MAX_TARGET_TEXT_LENGTH"],
    source_text,
    target_text,
      train = True,
```

```
)
val_set = Dataset(
    val_dataset,
    tokenizer,
    model_params["MODEL"],
    model_params["MAX_SOURCE_TEXT_LENGTH"],
    model_params["MAX_TARGET_TEXT_LENGTH"],
    source_text,
    target_text,
      train = False,
)

del train_dataset, val_dataset

# Defining the parameters for creation of dataloaders
train_params = {
    "batch_size": model_params["TRAIN_BATCH_SIZE"],
    "shuffle": True,
    "num_workers": 0,
}

val_params = {
    "batch_size": model_params["VALID_BATCH_SIZE"],
    "shuffle": False,
    "num_workers": 0,
}

# Creation of Dataloaders for testing and validation. This will be used down for tra
training_loader = DataLoader(training_set, **train_params)
val_loader = DataLoader(val_set, **val_params)

print("TRAIN LOADER: ", len(training_loader))
print("VAL LOADER: ", len(val_loader))

del train_params, val_params

# Defining the optimizer that will be used to tune the weights of the network in the
optimizer = torch.optim.Adam(
    params=model.parameters(), lr=model_params["LEARNING_RATE"]
)

if model_params["SCHEDULER"] == "linear":
    scheduler = torch.optim.lr_scheduler.LinearLR(optimizer)

# Training loop
console.log(f"[Initiating Fine Tuning]...\n")

for epoch in range(model_params["TRAIN_EPOCHS"]):

    loss = train(epoch, tokenizer, model, device, training_loader, optimizer, schedu
    losses.append(loss.cpu().numpy())

    # evaluating test dataset
    predictions, actuals = validate(epoch, tokenizer, model, device, val_loader)

    final_df = pd.DataFrame({"Generated Text": predictions, "Actual Text": actuals})
    final_df.to_csv(os.path.join(output_dir, f""predictions_{model_params['MODEL']}
    print("SAVE TO CSV FINISHED")


    rouge = compute_metrics(predictions, actuals, tokenizer)
```

```
    rouge_df = pd.DataFrame.from_dict(rouge, orient='index')
    rouge_df.to_csv(os.path.join(output_dir, f"""rouge_{model_params['MODEL']}_epoch
    print("SAVE ROUGE TO CSV FINISHED")

console.log(f"[Saving Model]...\n")
# Saving the model after training
path = os.path.join(output_dir, "model_files")
model.save_pretrained(path)
tokenizer.save_pretrained(path)

# converting list to array
arr = np.array(losses)
np.save(os.path.join(output_dir, f"""losses_{model_params['MODEL']}_epoch{model_para

console.save_text(os.path.join(output_dir, "logs.txt"))

console.log(f"[Validation Completed.]\n")
console.print(
    f"""[Model] Model saved @ {os.path.join(output_dir, "model_files")}\n"""
)
console.print(
    f"""[Validation] Generation on Validation data saved @ {os.path.join(output_dir,
)
console.print(f"""[Logs] Logs saved @ {os.path.join(output_dir,'logs.txt')}\n""")
```

Load data

In [21]:

```
from torch import cuda
device = 'cuda' if cuda.is_available() else 'cpu'
print("configured device: ", device)
```

```
configured device:  cuda
```

In [22]:

```
data = 'xsum'

if data == 'cnn_dailymail':
    dataset = load_dataset(data, '3.0.0')
    source_text = "article"
    target_text = "highlights"
elif data == "xsum":
    dataset = load_dataset(data)
    source_text = "document"
    target_text = "summary"
else:
    raise ValueError("Undefined dataset")
```

```
Using custom data configuration default
Reusing dataset xsum (/root/.cache/huggingface/datasets/xsum/default/
1.2.0/32c23220eadddb1149b16ed2e9430a05293768cfffbdfd151058697d4c11f93
4)

  0%|          | 0/3 [00:00<?, ?it/s]
```

Let's define model parameters specific to T5

In [23]:

```
# let's define model parameters specific to BART
model_params = {
    "MODEL": "t5-small",  # model_type: t5-base/t5-large
    "TRAIN_BATCH_SIZE": 16,  # training batch size
    "VALID_BATCH_SIZE": 16,  # validation batch size
    "TRAIN_EPOCHS": 3,  # number of training epochs
    "VAL_EPOCHS": 1,  # number of validation epochs
    "LEARNING_RATE": 2e-05,  # learning rate default betas=(0.9, 0.999), eps=1e-08
    "SCHEDULER": "linear",
    "MAX_SOURCE_TEXT_LENGTH": 512,  # max length of source text
    "MAX_TARGET_TEXT_LENGTH": 36,  # max length of target text
    "SEED": 42,  # set seed for reproducibility
}
```

In [24]:

```
console = Console(record=True)

training_logger = Table(
    Column("Epoch", justify="center"),
    Column("Steps", justify="center"),
    Column("Loss", justify="center"),
    title="Training Status",
    pad_edge=False,
    box=box.ASCII,
)

Trainer(
    dataset=dataset,
    source_text=source_text,
    target_text=target_text,
    model_params=model_params,
    output_dir=f"""./outputs/""",
    device = device,
)
```

```
[08:15:57] [Model]: Loading t5-small...                    <ipython-in
put-20-8830e8dcc794>:18
```

```
[08:16:00] [Data]: Reading data...                         <ipython-in
put-20-8830e8dcc794>:38
```

```
FULL Dataset: {'train': (204045, 3), 'validation': (11332, 3), 'test':
(11334, 3)}

TRAIN Dataset: (204045, 3)

TEST Dataset: (11332, 3)


TRAIN LOADER:  63
VAL LOADER:  63

[08:16:01] [Initiating Fine Tuning]...                     <ipython-inp
ut-20-8830e8dcc794>:105
```

```
STEP:  0 / 63

                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
```

```
+------------------------------------------------------------------
-----+

STEP:  10 / 63

                              Training Status
+------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-------------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+------------------------------------------------------------------
-----+

STEP:  20 / 63

                              Training Status
+------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-------------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+------------------------------------------------------------------
-----+

STEP:  30 / 63

                              Training Status
+------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-------------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
```

```
+-------------------------------------------------------------------
-----+
STEP:  40 / 63

                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                                Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-----+

STEP:  50 / 63

                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                                Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-----+

STEP:  60 / 63

                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                                Loss
```

```
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-----------------------------------------------------------------------
-----+


Completed 0

Completed 10

Completed 20

Completed 30

Completed 40

Completed 50

Completed 60

SAVE TO CSV FINISHED
SAVE ROUGE TO CSV FINISHED
STEP:  0 / 63

                                Training Status
+-----------------------------------------------------------------------
------+
|Epoch | Steps |                                Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
```

```
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-----+
```

```
STEP:  10 / 63

                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-----+
```

```
STEP:  20 / 63

                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
```

```
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-+-----+
```

```
STEP:  30 / 63
```

```
                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                                Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-+-----+
```

```
STEP:  40 / 63
```

```
                                Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                                Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 40/63 | tensor(5.8279, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-----+

STEP:  50 / 63

                                Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                                Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
```

```
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 40/63 | tensor(5.8279, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 50/63 | tensor(5.1422, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-----+

STEP:  60 / 63

                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 40/63 | tensor(5.8279, device='cuda:0', grad_fn=<NllLossBackw
```

```
ard0>)|
|  1   | 50/63 | tensor(5.1422, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 60/63 | tensor(5.3188, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+------------------------------------------------------------------
------+
Completed 0

Completed 10

Completed 20

Completed 30

Completed 40

Completed 50

Completed 60

SAVE TO CSV FINISHED
SAVE ROUGE TO CSV FINISHED
STEP:  0 / 63

                              Training Status
+------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 40/63 | tensor(5.8279, device='cuda:0', grad_fn=<NllLossBackw
```

```
ard0>)|
|  1   | 50/63 | tensor(5.1422, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 60/63 | tensor(5.3188, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 0/63  | tensor(5.4344, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-----+

STEP:  10 / 63

                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                                Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 40/63 | tensor(5.8279, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 50/63 | tensor(5.1422, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 60/63 | tensor(5.3188, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 0/63  | tensor(5.4344, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 10/63 | tensor(5.3791, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
```

```
STEP:  20 / 63

                              Training Status
+--------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 40/63 | tensor(5.8279, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 50/63 | tensor(5.1422, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 60/63 | tensor(5.3188, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 0/63  | tensor(5.4344, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 10/63 | tensor(5.3791, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 20/63 | tensor(5.5759, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+--------------------------------------------------------------------
-------+

STEP:  30 / 63

                              Training Status
+--------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
```

```
|
|------+-------+-------------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 40/63 | tensor(5.8279, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 50/63 | tensor(5.1422, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 60/63 | tensor(5.3188, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 0/63  | tensor(5.4344, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 10/63 | tensor(5.3791, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 20/63 | tensor(5.5759, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 30/63 | tensor(5.4420, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-----+
STEP:  40 / 63

                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-------------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
```

```
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 40/63 | tensor(5.8279, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 50/63 | tensor(5.1422, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 60/63 | tensor(5.3188, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 0/63  | tensor(5.4344, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 10/63 | tensor(5.3791, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 20/63 | tensor(5.5759, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 30/63 | tensor(5.4420, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 40/63 | tensor(5.4352, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+------------------------------------------------------------------
-----+
STEP:  50 / 63

                              Training Status
+------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-------------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
```

```
ard0>)|
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 40/63 | tensor(5.8279, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 50/63 | tensor(5.1422, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 60/63 | tensor(5.3188, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 0/63  | tensor(5.4344, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 10/63 | tensor(5.3791, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 20/63 | tensor(5.5759, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 30/63 | tensor(5.4420, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 40/63 | tensor(5.4352, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 50/63 | tensor(5.2034, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+-------------------------------------------------------------------
-----+
STEP:  60 / 63

                              Training Status
+-------------------------------------------------------------------
------+
|Epoch | Steps |                              Loss
|
|------+-------+-----------------------------------------------------
------|
|  0   | 0/63  | tensor(6.3002, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 10/63 | tensor(6.3517, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 20/63 | tensor(6.2645, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
```

```
|  0   | 30/63 | tensor(5.9552, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 40/63 | tensor(5.9280, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 50/63 | tensor(5.6532, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  0   | 60/63 | tensor(5.8273, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 0/63  | tensor(5.6962, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 10/63 | tensor(5.7922, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 20/63 | tensor(6.0882, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 30/63 | tensor(5.5599, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 40/63 | tensor(5.8279, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 50/63 | tensor(5.1422, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  1   | 60/63 | tensor(5.3188, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 0/63  | tensor(5.4344, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 10/63 | tensor(5.3791, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 20/63 | tensor(5.5759, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 30/63 | tensor(5.4420, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 40/63 | tensor(5.4352, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 50/63 | tensor(5.2034, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
|  2   | 60/63 | tensor(5.2346, device='cuda:0', grad_fn=<NllLossBackw
ard0>)|
+------------------------------------------------------------------
-+-----+
Completed 0

Completed 10

Completed 20

Completed 30

Completed 40

Completed 50

Completed 60

SAVE TO CSV FINISHED
SAVE ROUGE TO CSV FINISHED
```

```
[08:54:36] [Saving Model]...                                    <ipython-inp
ut-20-8830e8dcc794>:126
```

```
[08:54:37] [Validation Completed.]                              <ipython-inp
ut-20-8830e8dcc794>:138
```

```
[Model] Model saved @ ./outputs/model_files


[Validation] Generation on Validation data saved @ ./outputs/predictio
ns.csv


[Logs] Logs saved @ ./outputs/logs.txt
```

But is ROUGE score the best metric to evaluatate a summary?

Here are examples of the generated text. Let's take a closer look at the results.

Example 1:

At the highlighted row, you will notice that in the Generated Text, it exists `Graeme Roy` , while the Actual Summary it is `Jason Roy`

Example 2:

	Generated Text	Actual Text
71	yvette Cooper says it is not "appropriate" or "statesmanlike" for him to use such language. shadow minister said it was "offensiv…	Nico Rosberg and Mercedes have been penalised 10 seconds for breaking radio transmission rules during the British Grand Prix.

	Generated Text	Actual Text
8	Rory Burns and Tim Murtagh put on a 118-run fourth wicket stand as Surrey closed on 384-8. Roy's century was a fine retort a…	Labour leadership hopefuls Liz Kendall and Yvette Cooper have said their supporters should back anyone other than Jeremy C…

You can see that the name `Yvette Cooper` is contained in the sample **8th** of the Actual Summary, but it appeared in the sample **71st** of the Generated Text.

This problem is called **Name Entity Halluciation** which is one of comon problems found in Text Summarization task. Since summarization models are optimized to generate summaries that highly overlap with human references, the faithfulness of the summary is not guaranteed. ROUGE Score is insensitive to semantic errors.

These n-gram based approaches weight all portions of the text equally, even when only a small fraction of the n-grams carry most of the semantic content. As a result, factual inconsistency caused by small changes may be drowned out by otherwise high n-gram overlap.

Another type of problem is called **Factual Consistency**
For example:
the sentence: `My name is Beau.` and
the sentence: `My name is not Beau.`
share nearly all unigrams and bigrams despite having the opposite meaning.

Q2. Please spend some time explore 2 useful metrices that can be used to evaluate summary. Explain what they are briefly and explain how they could be

useful and why could they not. (1 pt)

Write your answer here.

Answer :

- METEOR (Lavie and Agarwal, 2007) computes an alignment between candidate and reference sentences by mapping unigrams in the generated summary to 0 or 1 unigrams in the reference, based on stemming, synonyms, and paraphrastic matches. Precision and recall are computed and reported as a harmonic mean.
- BertScore (Zhang et al., 2020) computes similarity scores by aligning generated and reference summaries on a token-level. Token alignments are computed greedily to maximize the cosine similarity between contextualized token embeddings from BERT.

https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00373/100686/SummEval-Re-evaluating-Summarization-Evaluation (https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00373/100686/SummEval-Re-evaluating-Summarization-Evaluation)

Reference:

https://paperswithcode.com/method/t5#:~:text=T5%2C%20or%20Text%2Dto%2D,to%20generate%20some%
(https://paperswithcode.com/method/t5#:~:text=T5%2C%20or%20Text%2Dto%2D,to%20generate%20some%

In [24]: