```
!pip install transformers[sentencepiece] datasets sacrebleu rouge_score py7zr -q
```

```
from transformers import pipeline, set_seed

import matplotlib.pyplot as plt

import pandas as pd
from datasets import load_dataset, load_metric
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

import pandas as pd
import numpy as np

import nltk
from nltk.tokenize import sent_tokenize

nltk.download("punkt")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

The CNN/DailyMail Dataset

An important aspect of the dataset is that the summaries are abstractive and not extractive, which means that they consist of new sentences instead of simple excerpts.

Extractive Summarization: the extractive approach selects the most important phrases and lines from the documents. It then combines all the important lines to create the summary. So, in this case, every line and word of the summary actually belongs to the original document which is summarized.

Abstractive Summarization: The abstractive approach uses new phrases and terms that are different from the original document, keeping the meaning the same, just like how humans do in summarization. So, it is much harder than the extractive approach.

```
from datasets import load_dataset

dataset = load_dataset("cnn_dailymail", version="3.0.0")

print(f"Features in cnn_dailymail : {dataset['train'].column_names}")
```

```
WARNING:datasets.builder:Using custom data configuration default
WARNING:datasets.builder:Reusing dataset cnn_dailymail (/root/.cache/huggingface/data
100%                                                3/3 [00:00<00:00, 2.75it/s]
```

```
sample = dataset["train"][1]
print(f"""
Article (excerpt of 500 characters, total length: {len(sample["article"])}):
""")
print(sample["article"][:500])
print(f'\nSummary (length: {len(sample["highlights"])}):')
print(sample["highlights"])
```

```
Article (excerpt of 500 characters, total length: 4051):

Editor's note: In our Behind the Scenes series, CNN correspondents share their experiences in covering news and analyze the stories

Summary (length: 281):
Mentally ill inmates in Miami are housed on the "forgotten floor"
Judge Steven Leifman says most are there as a result of "avoidable felonies"
While CNN tours facility, patient shouts: "I am the son of the president"
Leifman says the system is unjust and he's fighting for change .
```

Text Summarization Pipelines

```
sample_text = dataset["train"][1]["article"][:1000]

# We'll collect the generated summaries of each model in a dictionary
summaries = {}
```

Summarization Baseline

```
def baseline_summary_three_sent(text):
    return "\n".join(sent_tokenize(text)[:3])


summaries['baseline'] = baseline_summary_three_sent(sample_text)

summaries['baseline']
```

```
'Editor\'s note: In our Behind the Scenes series, CNN correspondents share their experiences in covering news and analyze the stori
es behind the events.\nHere, Soledad O\'Brien takes users inside a jail where many of the inmates are mentally ill. An inmate house
d on the "forgotten floor," where many mentally ill inmates are housed in Miami before trial.\nMIAMI, Florida (CNN) -- The ninth fl
oor of the Miami-Dade pretrial detention facility is dubbed the "forgotten floor."'
```

huggingface pipeline

The pipelines are a great and easy way to use models for inference.

Stating ”summarization”: will return a `SummarizationPipeline`

”text-generation”: will return a TextGenerationPipeline

GPT-2

We can use GPT-2 it to generate summaries by simply appending “TL;DR” at the end of the input text.

The expression “TL;DR” (too long; didn’t read) is often used on platforms like Reddit to indicate a short version of a long post. We will start our summarization experiment by re-creating the procedure of the original paper with the pipeline() function from Transformers

We create a text generation pipeline and load the GPT-2 model:

```
from transformers import pipeline, set_seed

set_seed(42)

pipe = pipeline('text-generation', model = 'gpt2-medium' )

gpt2_query = sample_text + "\nTL;DR:\n"

pipe_out = pipe(gpt2_query, max_length = 512, clean_up_tokenization_spaces = True)
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
```

```
pipe_out
```

```
[{'generated_text': 'Editor\'s note: In our Behind the Scenes series, CNN correspondents share their experiences in covering news
and analyze the stories behind the events. Here, Soledad O\'Brien takes users inside a jail where many of the inmates are mentally
ill. An inmate housed on the "forgotten floor," where many mentally ill inmates are housed in Miami before trial. MIAMI, Florida
(CNN) -- The ninth floor of the Miami-Dade pretrial detention facility is dubbed the "forgotten floor." Here, inmates with the most
severe mental illnesses are incarcerated until they\'re ready to appear in court. Most often, they face drug charges or charges of
assaulting an officer --charges that Judge Steven Leifman says are usually "avoidable felonies." He says the arrests often result
from confrontations with police. Mentally ill people often won\'t do what they\'re told when police arrive on the scene --
confrontation seems to exacerbate their illness and they become more paranoid, delusional, and less likely to follow
dir\nTL;DR:\nMIAMI-DADE, Florida | April 13, 2012 -- Some inmates are locked up in solitary confinement, where they must be
isolated from the world for six months before they\'re released onto the general jail population to participate in other inmates.
Others are housed in "theforgotten floor," the top floor of the pretrial facility that is also used by criminal offenders. What
makes these inmates separate is their medical needs. They\'re given special medications to treat their mental illness, but it\'s
only in those cases where necessary to take certain medications themselves for the very dangerous condition. These medications can
lead to anaphylaxis or dangerous reactions if taken by people who are on them alone. A mentally ill person like John Brown, a
convicted felon with severe mental illness. He spends four months in the "forgotten floor" and is charged with being a felon,
disorderly conduct, assault with a deadly weapon (a gun), possession of child pornography, resisting arrest, and possession of
marijuana. Brown was in the second floor when he fell asleep in bed. On April 11, his bed fell over and he was arrested and held
until jail officials could check on him. After his detention, Brown refused to return to court the next morning to answer the
charges. "I won\'t be back at all," Brown said. "There will be no new hearings or there will be no court." This episode will air on
January 18, 2013 on CNN\'s Inside Story.'}]
```

```
pipe_out[0]["generated_text"][len(gpt2_query) :]
```

```
'MIAMI-DADE, Florida | April 13, 2012 -- Some inmates are locked up in solitary conf
inement, where they must be isolated from the world for six months before they\'re r
eleased onto the general jail population to participate in other inmates. Others are
housed in "theforgotten floor," the top floor of the pretrial facility that is also
used by criminal offenders. What makes these inmates separate is their medical need
s. They\'re given special medications to treat their mental illness, but it\'s only
in those cases where necessary to take certain medications themselves for the very d
```

```
summaries['gpt2'] = "\n".join(sent_tokenize(pipe_out[0]["generated_text"][len(gpt2_query) :]))
```

T5

T5 (Text-To-Text Transfer Transformer) is a transformer model that is trained in an end-to-end manner with text as input and modified text as output, in contrast to BERT-style models that can only output either a class label or a span of the input. This text-to-text formatting makes the T5 model fit for multiple NLP tasks like Summarization, Question-Answering, Machine Translation, and Classification problems.

How T5 is different from BERT? Both T5 and BERT are trained with MLM (Masked Language Model) approach.

What is MLM?

The MLM is a fill-in-the-blank task, where the model masks part of the input text and tries to predict what that masked word should be.

Example:

“I like to eat peanut butter and sandwiches,” “I like to eat peanut butter and jelly sandwiches,”

The only difference is that T5 replaces multiple consecutive tokens with the single Mask Keyword, unlike, BERT which uses Mask token for each word. This illustration is shown below.

T5 expects a prefix before the input text to understand the task given by the user. For example,

- “summarize:” for the summarization,
- “cola sentence:” for the classification,
- “translate English to Spanish:” for the machine translation, etc.,

---

But here in this case, I can directly load T5 for summarization with the pipeline() function, which also takes care of formatting the inputs in the text-to-text format so we don’t need to prepend them with "summarize":

```
pipe = pipeline('summarization', model = 't5-small' )
```

```
pipe_out = pipe(sample_text)
```

```
/usr/local/lib/python3.7/dist-packages/transformers/models/t5/tokenization_t5_fast.py:166: FutureWarning: This tokenizer was incorre
For now, this behavior is kept to avoid breaking backwards compatibility when padding/encoding with `truncation is True`.
- Be aware that you SHOULD NOT rely on t5-small automatically truncating your input to 512 when padding/encoding.
- If you want to encode/pad to sequences longer than 512 you can either instantiate this tokenizer with `model_max_length` or pass `
- To avoid this warning, please instantiate this tokenizer with `model_max_length` set to your preferred value.
  FutureWarning,
```

```
pipe_out
```

```
[{'summary_text': "inmates with the most severe mental illnesses are incarcerated until they're ready to appear in court . most
often, they face drug charges or charges of assaulting an officer . mentally ill people become more paranoid, delusional, and less
likely to follow dir ."}]
```

```
summaries['t5'] = 'n'.join(sent_tokenize(pipe_out[0]['summary_text']))
```

BART

BART is a denoising autoencoder for pretraining sequence-to-sequence models. It is trained by (1) corrupting text with an arbitrary noising function, and (2) learning a model to reconstruct the original text. It uses a standard Transformer-based neural machine translation architecture.

That means, It uses a standard seq2seq/NMT architecture with a bidirectional encoder (like BERT) and a left-to-right decoder (like GPT). This means the encoder's attention mask is fully visible, like BERT, and the decoder's attention mask is causal, like GPT2.

This means that a fine-tuned BART model can take a text sequence (for example, English) as input and produce a different text sequence at the output (for example, French).

This type of model is relevant for machine translation, question-answering , text summarization, or sequence classification (categorizing input text sentences or tokens).

Another task is sentence entailment which, given two or more sentences, evaluates whether the sentences are logical extensions or are logically related to a given statement.

```
pipe = pipeline("summarization", model="facebook/bart-large-cnn")
pipe_out = pipe(sample_text)
```

```
pipe_out
```

```
[{'summary_text': 'Miami-Dade pretrial detention facility is dubbed the "forgotten floor" Here, inmates with the most severe mental
illnesses are incarcerated. Most often, they face drug charges or charges of assaulting an officer. Judge Steven Leifman says the
arrests often result from confrontations with police.'}]
```

```
summaries["bart"] = "\n".join(sent_tokenize(pipe_out[0]["summary_text"]))
```

```
summaries["bart"]
```

```
'Miami-Dade pretrial detention facility is dubbed the "forgotten floor" Here, inmates with the most severe mental illnesses are inc
arcerated.\nMost often, they face drug charges or charges of assaulting an officer.\nJudge Steven Leifman says the arrests often re
sult from confrontations with police.'
```

PEGASUS

The PEGASUS model's pre-training task is very similar to summarization, i.e. important sentences are removed and masked from an input document and are later generated together as one output sequence from the remaining sentences, which is fairly similar to a summary. In PEGASUS, several whole sentences are removed from documents during pre-training, and the model is tasked with recovering them. The Input for such pre-training is a document with missing sentences, while the output consists of the missing sentences being concatenated together. The advantage of this self-supervision is that you can create as many examples as there are documents without any human intervention, which often becomes a bottleneck problem in purely supervised systems.

```
pipe = pipeline('summarization', model="google/pegasus-cnn_dailymail"  )

pipe_out = pipe(sample_text)
```

```
pipe_out
```

```
[{'summary_text': 'Mentally ill inmates are housed on the "forgotten floor" of a Miami jail .<n>Judge Steven Leifman says the
charges are usually "avoidable felonies"<n>He says the arrests often result from confrontations with police .<n>Mentally ill people
often won\'t do what they\'re told when police arrive on the scene .'}]
```

```
summaries["pegasus"] = pipe_out[0]["summary_text"].replace(" .<n>", ".\n")
```

```
## Comparing Different Summaries
```

```
print("GROUND TRUTH")

print(dataset['train'][1]['highlights'])
```

```
for model_name in summaries:
    print(model_name.upper())
    print(summaries[model_name])
```

```
GROUND TRUTH
Mentally ill inmates in Miami are housed on the "forgotten floor"
Judge Steven Leifman says most are there as a result of "avoidable felonies"
While CNN tours facility, patient shouts: "I am the son of the president"
Leifman says the system is unjust and he's fighting for change .
BASELINE
Editor's note: In our Behind the Scenes series, CNN correspondents share their experiences in covering news and analyze the stories
Here, Soledad O'Brien takes users inside a jail where many of the inmates are mentally ill. An inmate housed on the "forgotten floor
MIAMI, Florida (CNN) -- The ninth floor of the Miami-Dade pretrial detention facility is dubbed the "forgotten floor."
GPT2
MIAMI-DADE, Florida | April 13, 2012 -- Some inmates are locked up in solitary confinement, where they must be isolated from the wor
Others are housed in "theforgotten floor," the top floor of the pretrial facility that is also used by criminal offenders.
What makes these inmates separate is their medical needs.
They're given special medications to treat their mental illness, but it's only in those cases where necessary to take certain medica
These medications can lead to anaphylaxis or dangerous reactions if taken by people who are on them alone.
A mentally ill person like John Brown, a convicted felon with severe mental illness.
He spends four months in the "forgotten floor" and is charged with being a felon, disorderly conduct, assault with a deadly weapon
Brown was in the second floor when he fell asleep in bed.
On April 11, his bed fell over and he was arrested and held until jail officials could check on him.
After his detention, Brown refused to return to court the next morning to answer the charges.
"I won't be back at all," Brown said.
"There will be no new hearings or there will be no court."
This episode will air on January 18, 2013 on CNN's Inside Story.
T5
```

```
inmates with the most severe mental illnesses are incarcerated until they're ready to appear in court .nmost often, they face drug c
BART
Miami-Dade pretrial detention facility is dubbed the "forgotten floor" Here, inmates with the most severe mental illnesses are incar
Most often, they face drug charges or charges of assaulting an officer.
Judge Steven Leifman says the arrests often result from confrontations with police.
PEGASUS
Mentally ill inmates are housed on the "forgotten floor" of a Miami jail.
Judge Steven Leifman says the charges are usually "avoidable felonies"<n>He says the arrests often result from confrontations with p
Mentally ill people often won't do what they're told when police arrive on the scene .
```

SacreBLEU

The bleu_metric object is an instance of the Metric class, and works like an aggregator: you can add single instances with add() or whole batches via add_batch(). Once you have added all the samples you need to evaluate, you then call compute() and the metric is calculated. This returns a dictionary with several values, such as the precision for each n-gram, the length penalty, as well as the final BLEU score. Let's look at the example from before:

```
from datasets import load_metric

bleu_metric = load_metric("sacrebleu")

bleu_metric.add(prediction = [summaries["pegasus"]], reference = [dataset['train'][1]['highlights'] ])

results = bleu_metric.compute(smooth_method = 'floor', smooth_value = 0 )

results['precision'] = [np.round(p , 2) for p in results['precisions'] ]

pd.DataFrame.from_dict(results, orient = 'index', columns = ['Value'] )
```

	Value
score	18.73841
counts	[27, 14, 10, 6]
totals	[67, 66, 65, 64]
precisions	[40.298507462686565, 21.21212121212121, 15.384...
bp	1.0
sys_len	67
ref_len	57
precision	[40.3, 21.21, 15.38, 9.38]

ROUGE

ROUGE vs BLEU

Bleu measures precision: how much the words (and/or n-grams) in the machine generated summaries appeared in the human reference summaries.

Rouge measures recall: how much the words (and/or n-grams) in the human reference summaries appeared in the machine generated summaries.

Interpretation of Rouge Score

ROUGE-n recall=40% means that 40% of the n-grams in the reference summary are also present in the generated summary.

The ROUGE score was specifically developed for applications like summarization where high recall is more important than just precision.5

The approach is very similar to the BLEU score in that we look at different n-grams and compare their occurrences in the generated text and the reference texts.

The difference is that with ROUGE we check how many n-grams in the reference text also occur in the generated text. For BLEU we looked at how many n-grams in the generated text appear in the reference

```
rouge_metric = load_metric('rouge')
```

ROUGE-N

With ROUGE-N, the N represents the n-gram that we are using. For ROUGE-1 we would be measuring the match-rate of unigrams between our model output and reference.

ROUGE-2 and ROUGE-3 would use bigrams and trigrams respectively.

ROUGE-L

ROUGE-L measures the longest common subsequence (LCS) between our model output and reference. All this means is that we count the longest sequence of tokens that is shared between both:

In the HF Datasets implementation, two variations of ROUGE are calculated: one calculates the score per sentence and averages it for the summaries (ROUGE-L), and the other calculates it directly over the whole summary (ROUGE-Lsum).

```
rouge_names = ["rouge1", "rouge2", "rougeL", "rougeLsum"]

reference = dataset['train'][1]['highlights']

records = []

for model_name in summaries:
    rouge_metric.add(prediction = summaries[model_name], reference = reference )
    score = rouge_metric.compute()
    rouge_dict = dict((rn, score[rn].mid.fmeasure ) for rn in rouge_names )
    print('rouge_dict ', rouge_dict )
    records.append(rouge_dict)

pd.DataFrame.from_records(records, index = summaries.keys() )
```

```
rouge_dict  {'rouge1': 0.365079365079365, 'rouge2': 0.14516129032258066, 'rougeL': 0.20634920634920634, 'rougeLsum': 0.285714285714
rouge_dict  {'rouge1': 0.1836734693877551, 'rouge2': 0.0410958904109589, 'rougeL': 0.10204081632653061, 'rougeLsum': 0.170068027210
rouge_dict  {'rouge1': 0.1758241758241758, 'rouge2': 0.0, 'rougeL': 0.13186813186813187, 'rougeLsum': 0.15384615384615383}
rouge_dict  {'rouge1': 0.3655913978494624, 'rouge2': 0.13186813186813184, 'rougeL': 0.2150537634408602, 'rougeLsum': 0.322580645161
rouge_dict  {'rouge1': 0.5, 'rouge2': 0.24489795918367346, 'rougeL': 0.36000000000000004, 'rougeLsum': 0.46}
```

	rouge1	rouge2	rougeL	rougeLsum
baseline	0.365079	0.145161	0.206349	0.285714
gpt2	0.183673	0.041096	0.102041	0.170068
t5	0.175824	0.000000	0.131868	0.153846
bart	0.365591	0.131868	0.215054	0.322581
pegasus	0.500000	0.244898	0.360000	0.460000

Evaluationg on the TEST set of the CNN/DailyMail Dataset

```
def calculate_metric_on_baseline_test_ds(dataset, metric, column_text = 'article', column_summary = 'highlights' ):
    """
    This function calculates a specified metric on a baseline test dataset for a Natural Language Processing (NLP) task.
    It assumes the task is a text summarization task, where the goal is to generate a summary (e.g., highlights) from a text (e.g., arti

    Parameters:
    dataset (pandas.DataFrame): The test dataset. It should contain a column for the text and a column for the true summary.
    metric (datasets.Metric): The metric to calculate. This should be a metric object from the Hugging Face datasets library.
    column_text (str, optional): The name of the column in the dataset that contains the text. Defaults to 'article'.
    column_summary (str, optional): The name of the column in the dataset that contains the true summary. Defaults to 'highlights'.

    Returns:
    score (float): The calculated score of the metric on the test dataset.
    """
    summaries = [baseline_summary_three_sent(text) for text in dataset[column_text] ]

    metric.add_batch(predictions = summaries, references = dataset[column_summary] )

    score = metric.compute()
    return score


test_sampled = dataset['train'].shuffle(seed = 42).select(range(1000))

score = calculate_metric_on_baseline_test_ds(test_sampled, rouge_metric )

rouge_dict = dict((rn, score[rn].mid.fmeasure ) for rn in rouge_names )

pd.DataFrame.from_dict(rouge_dict, orient = 'index' , columns = ['baseline'] ).T
```

	rouge1	rouge2	rougeL	rougeLsum
baseline	0.253995	0.100642	0.165754	0.231571

Strategy to calculate the ROUGE Metric on test dataset for the other Models

```
from tqdm import tqdm
import torch

device = "cuda" if torch.cuda.is_available() else "cpu"

def generate_batch_sized_chunks(list_of_elements, batch_size):
    """split the dataset into smaller batches that we can process simultaneously
    Yield successive batch-sized chunks from list_of_elements.

    Generator function to yield successive batch-sized chunks from list_of_elements.

    Parameters:
    list_of_elements (list): List of elements to be divided into chunks.
    batch_size (int): The size of each chunk.

    Yields:
    list: Batch-sized chunk from list_of_elements.

    """
    for i in range(0, len(list_of_elements), batch_size):
        yield list_of_elements[i : i + batch_size]

def calculate_metric_on_test_ds(dataset, metric, model, tokenizer,
                                batch_size=16, device=device,
                                column_text="article",
                                column_summary="highlights"):
    """
    Function to calculate a specified metric on a test dataset for a Natural Language Processing (NLP) task.
    It assumes the task is a text summarization task, where the goal is to generate a summary from a text.

    Parameters:
    dataset (pandas.DataFrame): The test dataset. It should contain a column for the text and a column for the true summary.
    metric (datasets.Metric): The metric to calculate. This should be a metric object from the Hugging Face datasets library.
    model (transformers.PreTrainedModel): The transformer model to use for text generation.
    tokenizer (transformers.PreTrainedTokenizer): The tokenizer corresponding to the model.
    batch_size (int, optional): The size of the batches to use for processing. Defaults to 16.
    device (str, optional): The device to run the model on. Defaults to the output of torch.cuda.is_available().
    column_text (str, optional): The name of the column in the dataset that contains the text. Defaults to 'article'.
    column_summary (str, optional): The name of the column in the dataset that contains the true summary. Defaults to 'highlights'.

    Returns:
    score (float): The calculated score of the metric on the test dataset.
    """
    article_batches = list(generate_batch_sized_chunks(dataset[column_text], batch_size))
    target_batches = list(generate_batch_sized_chunks(dataset[column_summary], batch_size))

    for article_batch, target_batch in tqdm(
        zip(article_batches, target_batches), total=len(article_batches)):

        inputs = tokenizer(article_batch, max_length=1024,  truncation=True,
                        padding="max_length", return_tensors="pt")

        summaries = model.generate(input_ids=inputs["input_ids"].to(device),
                         attention_mask=inputs["attention_mask"].to(device),
                         length_penalty=0.8, num_beams=8, max_length=128)
        ''' parameter for length penalty ensures that the model does not generate sequences that are too long. '''

        # Finally, we decode the generated texts,
        # replace the <n> token, and add the decoded texts with the references to the metric.
        decoded_summaries = [tokenizer.decode(s, skip_special_tokens=True,
                                clean_up_tokenization_spaces=True)
               for s in summaries]

        decoded_summaries = [d.replace("<n>", " ") for d in decoded_summaries]


        metric.add_batch(predictions=decoded_summaries, references=target_batch)

    #  Finally compute and return the ROUGE scores.
    score = metric.compute()
    return score
```

```
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

model_ckpt = "google/pegasus-cnn_dailymail"

tokenizer = AutoTokenizer.from_pretrained(model_ckpt)

model_pegasus = AutoModelForSeq2SeqLM.from_pretrained(model_ckpt).to(device)

score = calculate_metric_on_test_ds(test_sampled, rouge_metric,
                                    model_pegasus, tokenizer, batch_size=8)

rouge_dict = dict((rn, score[rn].mid.fmeasure) for rn in rouge_names)

# At the end, we compute and return the ROUGE scores.
pd.DataFrame(rouge_dict, index=["pegasus"])
```