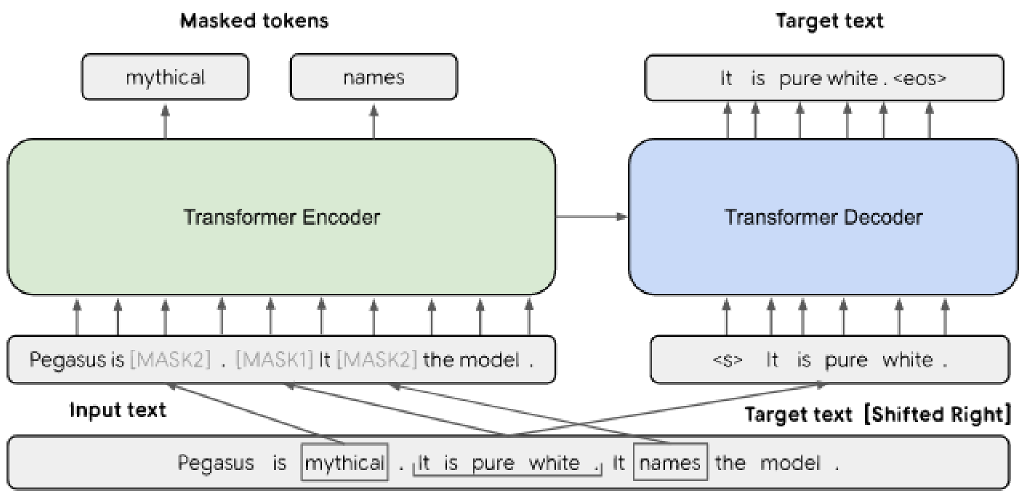
# Report

## Tools and Dependencies

- pytorch (*pip install torch*)
- huggingface Transformer library (*pip install transformers*)
- sentencepiece tokenizer (*pip install sentencepiece*)
- pandas and numpy for preprocessing and analysis of data
- Tesla K80 GPU with CUDA version 11.2 on google-colab.

## PEGASUS

PEGASUS is a transformer Encoder-Decoder model designed to do the task of abstractive text summarization. It was proposed in the paper: PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. PEGASUS is pre-trained on C4 and HugeNews dataset. C4 contains 750GB of textual data from 350 millions web pages and HugeNews contains 3.8TB of textual data collected from news and news-like articles. Following image shows high-level working of PEGASUS:


Masked tokens
mythical
names
Target text
It is pure white . <eos>
Transformer Encoder
Transformer Decoder
Pegasus is [MASK2] . [MASK1] It [MASK2] the model .
<s> It is pure white .
Input text
Target text [Shifted Right]
Pegasus is mythical . It is pure white . It names the model .

## Pegasus Checkpoint

The checkpoint of the PEGASUS transformer I used is pegasus-xsum. This checkpoint has been fine-tuned on the Extreme Summarization (xsum) dataset. There is no specific reason to use this particular checkpoint. There are other checkpoints published on github by the authors as well.

## Reason for using PEGASUS

PEGASUS model is very good on summarization tasks which essentially is a Sequence-to-Sequence or text-to-text task. There are other transformer models that are good at summarization tasks as well like BART and T5 but Pegasus outperformed both.

## Fundamental Idea: De-summarization

Right now, what happens is that you give a long text to PEGASUS, or any other transformer with a similar function, and it outputs a much smaller text (summary). My idea for joke generation is to use the same transformer architecture but see what happens if you flip the input and output. Give the input as a small text (title or category of a joke) and the model outputs a long/lengthy text (the joke).

(The idea sounds ingenious I know!)

## Hypothesis

PEGASUS will work better than GPT or GPT-2 because it is an encoder-decoder architecture and will better encode or represent the prompt (title or category of joke) and as a result will produce much more coherent outputs. GPT-2 on the other hand is only a decoder.

## Experiments

- Recommended parameters were used for fine-tuning the model. It only used 100 epochs for the categories experiment and we will see why.

```
training_args = TrainingArguments(
    output_dir='./results',          # output directory
    num_train_epochs=100,              # total number of training epochs
    per_device_train_batch_size=64,  # batch size per device during training
    per_device_eval_batch_size=64,   # batch size for evaluation
    logging_dir='./logs',            # directory for storing logs
    logging_steps=5,
    eval_accumulation_steps=1,
    learning_rate=1e-4,
    adafactor = True                #use adafactor instead of adam
)
```

- Joke generation is essentially done using beam search with a few added parameters. The top_k parameter sets the value of k-most next likely words. Default value of top_k is 50. The top-p parameter is used for sampling from the smallest possible set of words whose cumulative probability exceeds the probability p. More on decoding/generating methods here. The minimum length of the joke is set to 32. You can change it if you want longer text but we will see why you should not.

```
batch = tokenizer('Medical', truncation=True, padding='longest', return_tensors="pt").to(torch_device)
generated = model.generate(**batch, min_length=32, do_sample=True, top_p=0.92, top_k=0, num_beams=8, no_repeat_ngram_size=2)
tgt_text = tokenizer.batch_decode(generated, skip_special_tokens=True)
```

## Jokes based on categories

- Number of training examples is 13654.
- Parameters for the tokenizer were selected by analyzing the data. The average length of the category text is 8 and average length of the joke based on category is 600. So the parameters for input and output text are set as follows:

```
train_encodings = tokenizer(train_texts, max_length=16, truncation=True, padding='longest')
```

```
train_labels = tokenizer(train_decode, max_length=512, truncation=True, padding='longest')
```

- When we train the model for 5 or 10 epochs. The output makes sense but it looks nothing like a joke. It looks like text from a news article because the model we are using has previously been pre-trained and fine-tuned on datasets which are entirely or mainly news articles.Following are the examples of output of 5 and 10 epochs:
  - **Input:** Blonde Joke
  - **5-epoch output:** Comedians and stand-up comedians from around the world have been sharing their best-of-the-week jokes in a variety of different formats on social media.
  - **10-epoch output:** Blonde Blonde Scenarios - What do you do if you're not happy with the colour of the skin on your face? Here are some of my favourites:"
- The  outputs for 20-30-50 epochs are similar for all the joke categories. Let's see if the output looks more like a joke after 100 epochs for the same input.
  - As the sun sets over the hills, a woman in a negligee is seen sitting by the fire in the back of a backwoods house, not far from the window of the manufactured home she was trying to escape the heat. When the woman looks at the outside shot, the smoke rises high into the

  This looks like a news article but it could be for a (incomplete) joke as well because jokes can be lengthy. There are examples of jokes which have a sequence length of 5000. If we change the input from blonde joke to **medical**. We will have the following output:
  - A woman in her 30s has been in and out of hospital for more than a year. Her doctor has told her that if she ever had any problems with her heart, her first words to him would be "I\'m sorry! But if your heart is as soft as yours, you\'ll

  Increase the minimum length to 64. The output looks like following:
  - A woman was taken to the hospital after being bitten by a dog in a McDonald's restaurant in the early morning hours of Tuesday morning. The woman, who was not identified, regained consciousness and the dog ran back into the kitchen. When the back pain got the better of the woman and she went back to

  This has started to look like a joke because the text is a bit surprising.

- Let’s do a final run where we train the model for a 100 more epochs and see the difference. The input is the same as the previous experiment. The output length is set to minimum 64. We get the following output:
    - A certain lawyer was quite wealthy and had a summer house in a country town in the country, to which he retreated for several weeks of the year. Each summer, the lawyer would invite a different friend of his (no, that's not the punch line) to spend a week or two up at this

  Okay, this most certainly is a joke but an incomplete one.

So after 200 epochs, we can see that the structure and wording has started to look more like a joke. The reason it is incomplete is because PEGASUS, and the rest of the current generation of transformers based on encoder-decoder architecture, are NOT designed for lengthy outputs and hence are not able to produce lengthy outputs. If asked to produce a lengthy output, let’s say a sequence of length 512, it will still give us an incomplete output. For example, we get the following output if we set minimum length to 256 for the input medical:
“A woman was standing in the middle of a crowded lift of the hotel she was staying in when a man grabbed her by the scruff of his neck. The man with him lunged back and forth and tried to get back to his feet. But the man wasn't so lucky and the woman fell back
”

## Jokes based on title

- Number of training examples: 202526
- Parameters for the tokenizer were selected by analyzing the data. The average length of title text is 44 and average length of the joke is 224. So the parameters for input and output text are set as follows:

  ```
  train_encodings = tokenizer(train_texts, max_length=64, truncation=True, padding='longest')
  ```

  ```
  train_labels = tokenizer(train_decode, max_length=256, truncation=True, padding='longest')
  ```
- The approach to this is similar to the category text. Since there are a lot more training examples, we can expect the model to be fine-tuned in fewer examples.
- Output in 5 and 10 epochs. Minimum length of joke is set to 32.
    - **Input:** I just dropped my phone in a load of mayo.
    - **5-epoch output:** I'm not the only one who's had a bad experience with this thing, but I have to admit it was a pretty bad thing to do.
    - **10-epoch output:** I just dropped my phone in a load of mayo, and now I have a tub of mayonnaise to go with it, I'm not so sure.

  The above examples do look like we are on the correct path. Let us generate more jokes on different inputs and see the output.
- I randomly selected 50 examples from the test set and generated jokes. Most of them look like text from news or news-like articles. One of the examples I found funny but is a bit dark is follows:
    - **Input:** Today holds a lot of meaning to me. Today is the 2 year anniversary that I lost my wife and children

    - **Output:** Today is a very important day for me, it is the day I was born and I am very grateful for this day, I can't wait to see it through to the end of the world, but I will do my best to make sure it comes to a successful end.

  Another example I thought could be a joke when you read the first few lines but as the sequence grows, it looks like the model is just blabbering on. Also the output is incomplete:

    - **Input:** A lawyer, A priest and a scout Leader with  his troupe are on a plane.
    - **Output:** A priest and a scout leader are on their way to a meeting in London when they are stopped by a group of young men who do not want to talk to them about the meeting and ask them to leave the room and are told not to return until they have finished with the other members of their group and they

- Since some of the outputs to very few prompts look close to jokes, we can try and see what happens if we train a model for 15 epochs. We get the following jokes:
    - **Input:** A lawyer, A priest and a scout Leader with  his troupe are on a plane.
    - **Output:** A priest and a scout Leader with his troupe have been on a plane to visit the victims of the Paris attacks, but are not expected to be reunited with their families until next week.

  Okay. That seems a bit funny because of the sudden twist.

    - **Input:** Today holds a lot of meaning to me. Today is the 2 year anniversary that I lost my wife and children.
    - **Output:** Today is the anniversary of the death of my mother-in-law, who died after a long battle with cancer, she died in a car crash.

  I don't know if it is funny or if the model completely lost it. Some people may find it funny.

We can observe similarities between title-based and category-based joke generation. The reason we fine-tune the model here for a lot less epochs is that there are a lot more examples here. But the model has similar problems like unfinished sentences and problems with generating coherent longer sequences. The model behaves the same way as the category-based model if we increase the minimum length of the joke.

## Discussion

My idea of de-summarization has not been successful when it comes to generating jokes. There are two very important points I would like to make on why de-summarization is very difficult to achieve:

1. The task of de-summarization is basically a creative writing process. It is like you are given a topic and you are asked to write an essay on it. Similarly, the task of generating jokes is also a creative writing process. So it is very difficult to measure such a process automatically. Until or unless we find a way to create an automatic creative writing metric (which sounds ridiculous) the task of generating something big from something small is going to be a hit and a miss.
2. Transformers, like Neural Networks, in the most simple terminology are language models. When asked to generate text, all they do is predict the next token or set of

tokens. So, it is overly optimistic of us to expect that transformers will generate a good joke (or anything else) everytime.

The second thing that failed is my hypothesis. GPT-2 was able to produce much much better jokes on more consistent bases, according to Anna's report, than PEGASUS.