**Digital Assignment – 1**

Course: Programming for DataScience (CSE3046)

Faculty: Alkha Mohan

Slot: C2 + TC2

Student 1 name: Mugundhan P

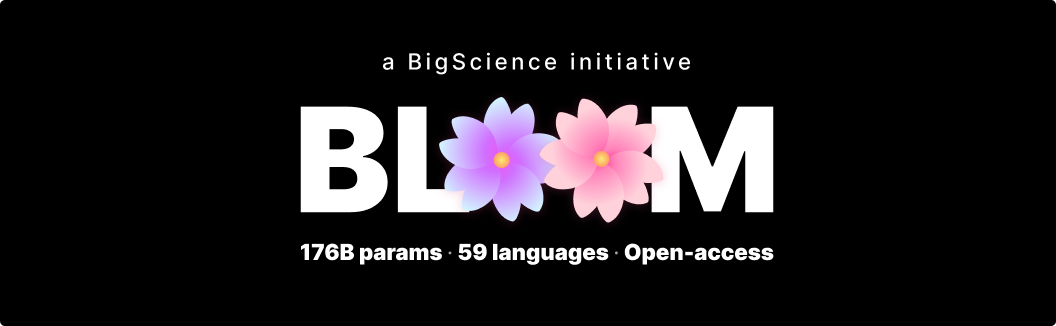
Reg No: 20BDS0372

Student 2 name: Atulya Prabhanjan M

Reg No: 20BDS0305

Assignment Topic: Case study – BLOOM

**Introduction:**



There was major news in the field of large language models (LLMs) on july 12,2020 that affected the artificial intelligence and data science (more especially, NLP) industries. One of the largest research workshops in the field of NLP, The BigScience, an open collaboration of Hugging Face, GENCI, and IDRIS, has introduced complete transparency and open sourced multilingual large language model BLOOM clipped form of BigScience Large Open-science Open-access Multilingual Language Model. With the help of the following hints, let's chat a little more in depth.

BigScience, a Hugging Face-hosted open cooperation with hundreds of academics and institutions worldwide, has developed Bloom, a new 176B parameter multilingual LLM (Large Language Model) trained on 1.6 terabytes of data, including natural language and software source code.

BLOOM is able to generate text in 46 natural languages and 13 programming languages. For almost all of them, such as Spanish, French and Arabic, BLOOM will be the first language model with over 100B parameters ever created.

The fact that Bloom is entirely open source and that Huggingface has made their complete (as well as some smaller) pre-trained models accessible to the public via its transformers API is what makes it so unique, in addition to the diversity of participants. Other research-focused firms, like OpenAI, Meta, and Google, have chosen to keep their LLMs mostly internal or have restricted access to small, carefully curated groups of closed beta testers

**Evaluation of large language models:**

When [BERT](https://en.wikipedia.org/wiki/BERT_(language_model)) came out it was pretty clear which was the path the industry had chosen for the future of the [natural language processing](https://towardsai.net/p/nlp/natural-language-processing-nlp-with-python-tutorial-for-beginners-1f54e610a1a0) field. BERT was the first transformer that really got the attention but not the last (sadly, we can say the same for the movie series).

BERT opened the way to [BART](https://arxiv.org/abs/1910.13461), [RoBERTa](https://arxiv.org/abs/1907.11692), and other large transformers models. It showed that a stack of self-attention layers and more parameters was astonishingly good for many tasks (named entity recognition, translation, question answering, etc…).

Then in 2020 [OpenAI](https://openai.com/) strongly entered the competition with [GPT-3](https://openai.com/api/) (a giant model with about 175 billion parameters). It was impressive but it stayed on the throne just for a while, Google and a few other companies released a parade of just bigger models. We saw [Gopher](https://www.deepmind.com/blog/language-modelling-at-scale-gopher-ethical-considerations-and-retrieval) (280 billion), [PALM](https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html) (540B), and [LaMDA](https://blog.google/technology/ai/lamda/) (137B). With the exception of [chinchilla](https://arxiv.org/abs/2203.15556) (70 billion, not very small anyway) the principle was the same: gather more data and increase the number of parameters.

It was a race with few participants. The truth was that BERT showed the world that only the tech blue-chip companies could compete in the game. GPT3 is estimated to cost just [10–20 million dollars to train](https://lastweekin.ai/p/gpt-3-is-no-longer-the-only-game) if we just consider the electricity bill (imagine how much it costs to buy all the [GPU](https://towardsai.net/p/deep-learning/what-is-a-gpu-are-gpus-needed-for-deep-learning-7b315ed80f16) for the training).

**Training period of BLOOM:**

Over 1,000 academics worked together at the BigScience workshop to create a sizable, multilingual deep-learning model starting in May 2021. Members of the Institute for Development and Resources in Intensive Scientific Computing (IDRIS) and Grand Equipement National De Calcul Intensif participated in the partnership (GENCI).

These gave the workshop access to the supercomputer Jean Zay 28 PFLOPS. To train the model, the researchers forked the Megatron-DeepSpeed source and used three different parallelism dimensions to achieve a training throughput of up to 150 TFLOPs. Training the final BLOOM model took 117 days, according to NVIDIA, and is "the greatest throughput one can get with A100 80GB GPUs."

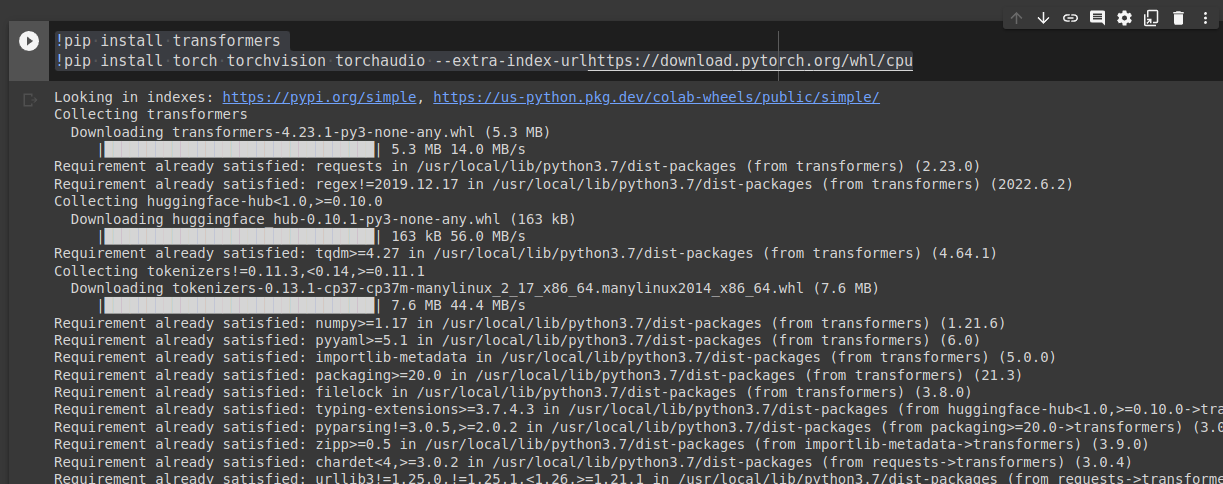
**Code:**

We’re going to be using the 1.1B parameter version of the general Bloom model in PyTorch, running using google colab GPU.

First of all installing the Transformers Library:-

!pip install transformers

!pip install torch torchvision torchaudio --extra-index-url<https://download.pytorch.org/whl/cpu>



Importing the needed libraries:-

import transformers

from transformers import BloomForCausalLM

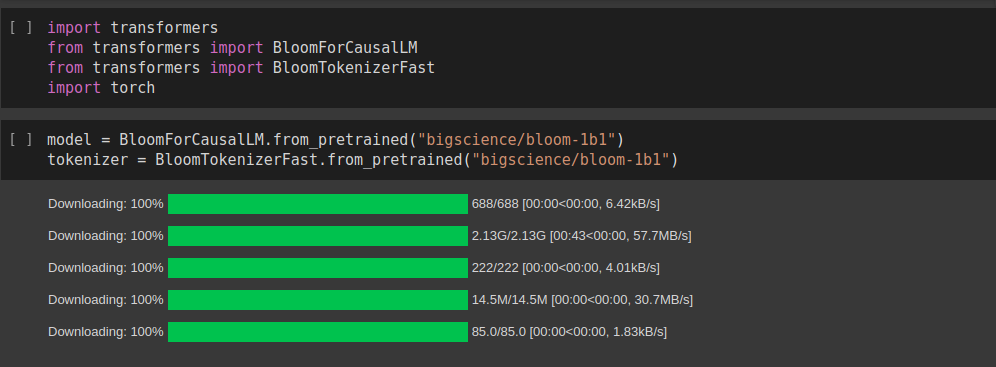
from transformers import BloomTokenizerFast

import torch

Now we download the pre-trained Bloom 1.3B parameter general LLM:-

model = BloomForCausalLM.from\_pretrained("bigscience/bloom-1b1")

tokenizer = BloomTokenizerFast.from\_pretrained("bigscience/bloom-1b1")



prompt = "The SQL command to extract all the users whose name starts with A is: "

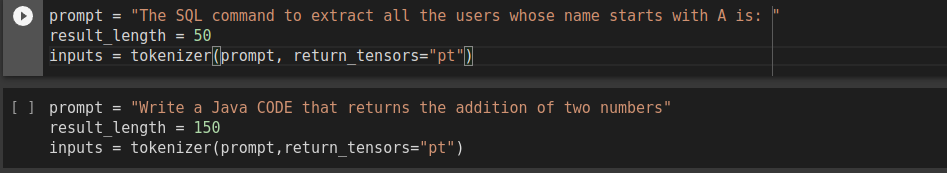
result\_length = 50

inputs = tokenizer(prompt, return\_tensors="pt")

prompt = "Write a Java CODE that returns the addition of two numbers"

result\_length = 150

inputs = tokenizer(prompt,return\_tensors="pt")



**result\_length** calibrates the size of the response (in tokens) we get for the prompt from the model.

inputs contains the embedding representation of prompt, encoded for use specifically by PyTorch. If we were using TensorFlow we’d pass **return\_tensors="tf"**.

**Beam Search:**

The most likely word sequence is produced by Beam Search after keeping track of the n-th (num beams) most probable word sequences. Sounds wonderful, but this approach fails when the output length is widely changeable, as it does when producing open-ended text.

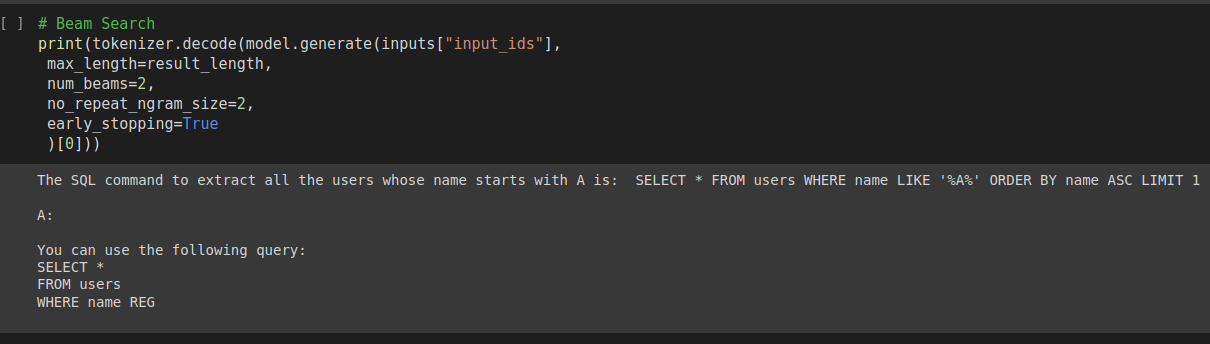
Additionally, the distribution of the results from both beam search and greedy are not very similar to how humans could approach the same task (i.e. both are liable to produce fairly repetitive, boring text).

# Beam Search print(tokenizer.decode(model.generate(inputs["input\_ids"], max\_length=result\_length,

num\_beams=2,

no\_repeat\_ngram\_size=2,

early\_stopping=True )[0]))

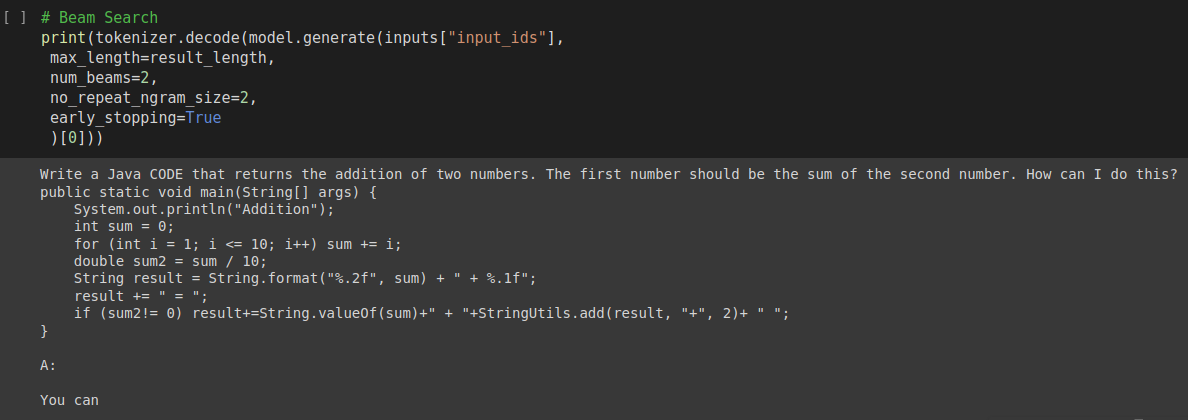


# Beam Search print(tokenizer.decode(model.generate(inputs["input\_ids"], max\_length=result\_length,

num\_beams=2,

no\_repeat\_ngram\_size=2,

early\_stopping=True )[0]))



**Architecture/Algorithm:**

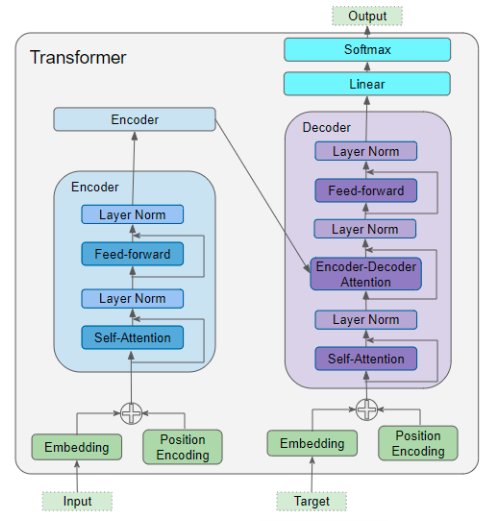
BLOOM is a transformer that employs multi-head attention and has 70 layers.

In order to understand what this statement means, let’s take a look at what each of these terminologies mean and how they work to understand BLOOM.  
  
*Transformer:*  
A transformer architecture is one in which a text sequence is given as the input for the model, and another text sequence is generated as the output.

Diagram

Description automatically generated

In it’s raw form, it just consists of stacks of encoders and decoders. Each of these stacks have their own embedding layer for their respective output. After all this, there is an output layer, which generates the final output.

**

The above image is a flowchart representation of the transformer model.

*Attention:*

While Processing a word, the attention is the concept that encourages the model to focus more on the other words present in the input, in order to gain more inference or context of the statement.

In simple words, it is like concentrating on the verbs, adjectives, adverbs etc. present in the sentence, as these words explain about what the subject is doing and it’s context.

For example, consider the following statements :-

* The *cat* drank the milk because **it** was hungry.
* The cat drank the *milk* because **it** was sweet.

In the above 2 statements, the word “it” refers 2 different things. The **it** from the first sentence refers to the **cat**, whereas the **it** from the second statement refers to the **milk**.  
  
Let’s take a look on how attention helps the transformer to understand the concept:-



In the above picture, darker colours represent higher attention.

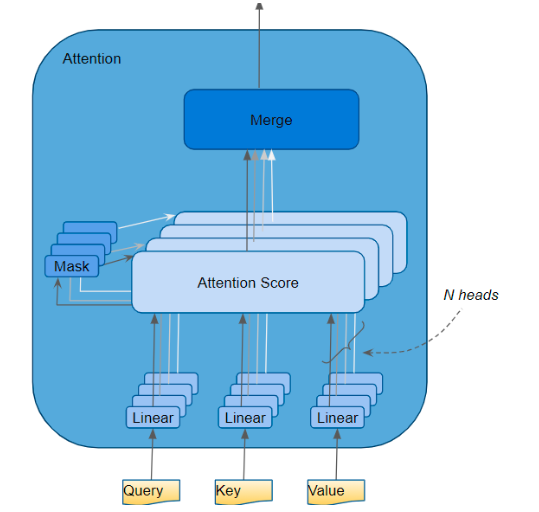
Now, let’s take only the first statement and see how the words are chosen to have priorities based on attention scores.



Each word is scored based on the importance of the word in the sentence and the relation of the word with every other word present in the sentence and context.

*Multi-head Attention:*

It is a module for attention mechanism in which multiple attention mechanisms are run parallel to each other. The output of these mechanisms are then concatenated and are stored in the required dimensions.



The Attention layer takes its input in the form of three parameters, known as the Query, Key, and Value. All three parameters are similar in structure, with each word in the sequence represented by a vector.

Text

Description automatically generated with medium confidence

**Conclusion:**

In this blog, we had explored about the BLOOM model, about it’s usage, architecture and it’s implementation. BLOOM has a wide range of applications in the real world due to it’s NLP capabilities other than the traditional uses. Some of applications would be a smarter IDE for programming languages, better translators, chat bot applications, etc.

**Images from:**

<https://towardsdatascience.com/>

**References:**

<https://towardsdatascience.com/>

<https://data-science-blog.com/blog/2021/04/07/multi-head-attention-mechanism/>

<https://d2l.ai/chapter_attention-mechanisms-and-transformers/multihead-attention.html>

<https://towardsai.net/p/l/a-new-bloom-in-ai-why-the-bloom-model-can-be-a-gamechanger>

<https://paperswithcode.com/method/multi-head-attention>

**GitHUB Link:**

<https://github.com/Atulya-Prabhanjan/BLOOM>