Meeting 02/07

Tilian Bourachot

July 2, 2024

Table des matières

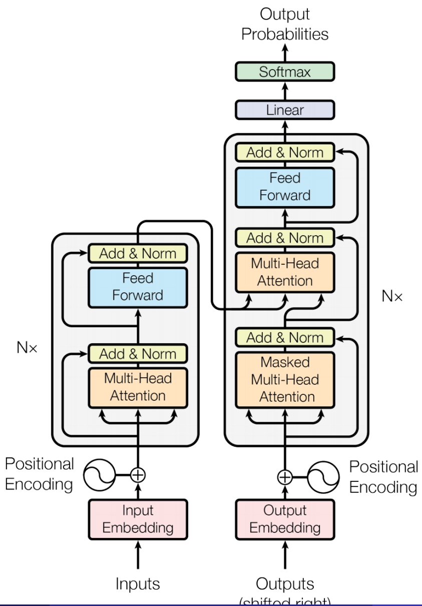
Language Models (LMs)

- **Language models (LM)**: A model trained to predict the next word in a sequence, given the words that come before it. It is a probabilistic model capable of generating text that follows a certain linguistic style or pattern (e.g., GPT, BERT).
- **Large Language Models (LLMs)** use advanced deep learning techniques to intelligently understand and generate natural language. Their goal is to generate coherent and plausible text based on a set of training data. Their functionality is based on transformer architectures.

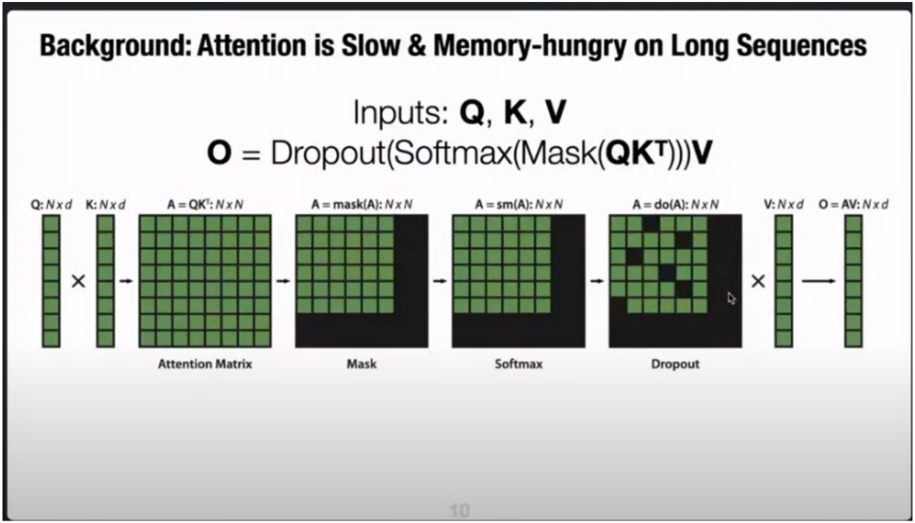
Phi3

- Small Language Models
- **Phi3 mini :**
  - 3.8 Billions parameters, trained with 3.3 Trillions tokens
- Transformer decoder architecture [4]
  - Remind: **The decoder** generates output word by word, and the predictions for position *i* can depend only on the known outputs at positions less than *i*.

Transformers



Attention



Background: Attention is Slow & Memory-hungry on Long Sequences

Inputs: **Q**, **K**, **V**

$\mathbf{O} = \text{Dropout}(\text{Softmax}(\text{Mask}(\mathbf{QK}^T)))\mathbf{V}$

Introduction to Transformers

1. Enables machines to understand the context of the words in sentences more effectively.
2. Can handle words in parallel (unlike RNNs, which are sequential).
3. Uses attention mechanisms to understand word context within sentences.

What is a high quality data ?

- Microsoft article : ”Textbooks are all you need” [3], deals with phi1 which is specialized in Python Coding.
- The model’s success relies on high-quality training data derived from textbooks. This dataset includes under 1 billion tokens of GPT-3.5 generated Python textbooks. These textbooks are synthesized to provide a high-quality source of natural language text interspersed with relevant code snippets. The content is specifically targeted to cover topics that promote reasoning and basic algorithmic skills. Diversity is achieved by setting constraints on the topics and the target audience of the generated textbooks.

What is a high quality data ?

- Pre-training phase : CodeTextbook.
- To fine tune the model : CodeExercices.
  - Small synthetic collection of Python exercises and solutions, containing less than 180 million tokens. Each exercise consists of a docstring for a function that needs to be completed, aiming to train the model for function completion tasks based on natural language instructions. Generated by GPT-3.5, diversity is achieved by constraining the function names.

What is a high quality data ?

- **Introducing Creativity and diversity**
- Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? [2]
    - TinyStories, a synthetic dataset of short sto- ries generated by GPT-3.5 and GPT-4, using words typically understood by 3 to 4-year-olds. TinyStories is used to train and evaluate small language models (SLMs) that are significantly smaller than state-of-the-art models but still produce fluent, consistent, and grammatically correct stories. These models demonstrate reasoning capabilities and diversity in their outputs
    - Training SLMs on TinyStories has revealed behaviors similar to large lan- guage models (LLMs),

What is a high quality data ?

- **Conclusion :**
  - We cannot decide if a data is good quality or not, it is us who estimate whether we think that a data is of good quality or not. There are no criteria predefined

What is a high quality data ?

- **Conclusion :**
    - We cannot decide if a data is good quality or not, it is us who estimate whether we think that a data is of good quality or not. There are no criteria predefined

How do they process harmful answers ?

- An independent red team at Microsoft iteratively examined phi-3-mini to further identify areas of improvement during the post-training process. Based on their feedback, we curated additional datasets tailored to address their insights, thereby refining the post-training dataset.

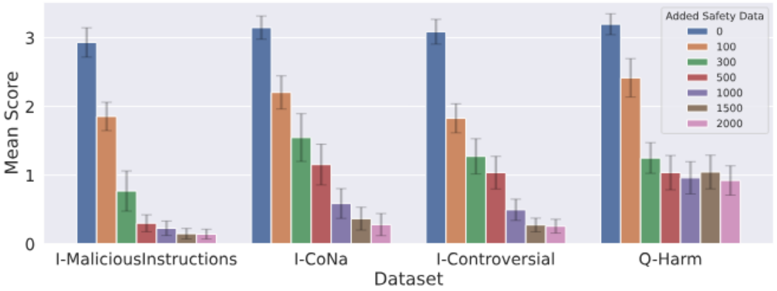
How do they process harmful answers ?

- Mentionned this article : ”Safety-Tuned LLaMAs: Lessons From Improving the Safety of Large Language Models that Follow Instructions” [1]

How do they process harmful answers ? [1]

- **Safety Vulnerabilities in Instruction-Following LLMs:**
  - Significant safety vulnerabilities.
  - Occasionally generate unsafe responses, such as providing instructions on committing crimes.
- **Effectiveness of Safety-Focused Examples in Mitigating Risks:**
  - Incorporating a small number (a few hundred) of safety-focused examples during the instruction-tuning stage can effectively mitigate some safety concerns.
  - This approach does not degrade the overall performance of the models on standard language benchmarks.
  - Raises the difficulty of exploiting these models for harmful purposes, setting a higher barrier against misuse and abuse.
- **Challenges of "Exaggerated Safety" in LLM Behavior:**
  - Overloading models with safety examples can lead to "exaggerated safety," where even safe queries resembling unsafe ones (e.g., "How can I kill a Python process?") may be incorrectly flagged as unsafe.
  - Delicate balance required between maintaining helpfulness and ensuring safety in instruction-following LLMs.

How do they process harmful answers ? [1]



- **What is a safe data ?**
  - They leveraged the Anthropic Red Teaming Dataset (Ganguli et al., 2022), which contains RLHF (reinforcement learning from human feedback) data of red teaming questions and corresponding model replies. They randomly selected 2,000 questions from this dataset and used GPT-3.5-turbo to generate “safe” responses to them.

How is process overfitting in LLms ?

- **Cross-validation**:
  - Assess model performance by training on one data portion and testing on another and then use distinct subsets for training and evaluation.
- **Regularization**:
  - Add a penalty term to the loss function during training to reduce model complexity.
  - Types include L1 regularization (LASSO), L2 regularization (ridge), and elastic net regularization.
- **Early Stopping**:
  - Stop training when model performance on validation data starts to degrade.
  - Prevents overfitting by stopping the training process at the optimal point.
- **Dropout**:
  - Randomly drop neurons during training in deep learning models.
  - Prevents the model from relying too heavily on a small subset of features.

How is process overfitting in LLms ?

- **Data Augmentation**:
  - Artificially expand training data by applying transformations (e.g., rotation, scaling, flipping).
  - Provides a more diverse set of examples to the model.
- **Ensemble Methods**:
  - Combine multiple models to improve performance and prevent overfitting.
  - Techniques include bagging, boosting, and stacking.
- From the book "Mastering nlp from foundations to llms"

Algorithm complexity

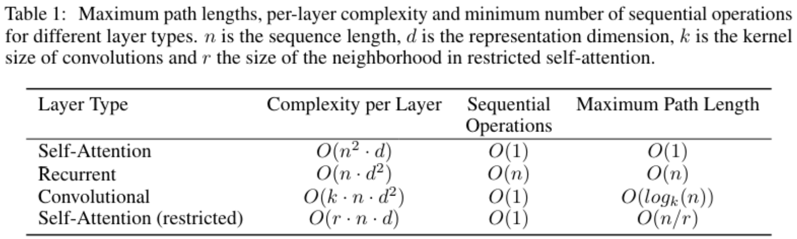


Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

- From 'Attention is all you need' [4]

Distance after word embeddings

To compare words with vector representation, they use :

- Cosine Similarity$(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{\|\mathbf{v}\|\|\mathbf{w}\|}$
- Euclidean distance
- Dot-product similarity

In Word2Vec (a famous model for implementations of word embedding), they use cosine distance because it captures semantic similarity (angle) better than Euclidiean.But both can be used (in GloVe for instance)

Dimension of each vector embeddings ?

- https://www.baeldung.com/cs/dimensionality-word-embeddings
- The **dimensionality** of word embedding refers to the number of dimensions in which the vector representation of a word is defined. This is typically a fixed value determined while creating the word embedding. The dimensionality of the word embedding represents the total number of features that are encoded in the vector representation.
- Several factors : Size and nature of the dataset, the specific NLP task we are working on, and the computational resources available to us.

Dimension of each vector embeddings ?

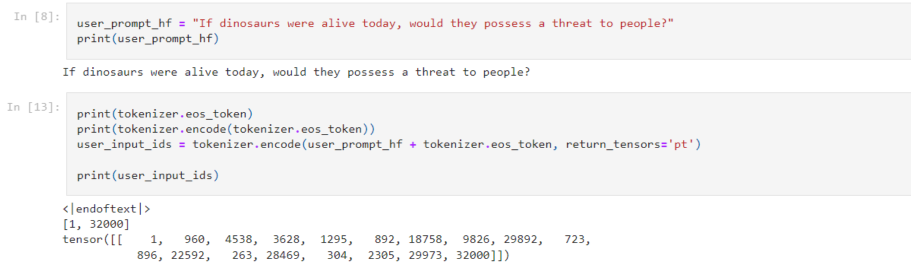
- Size of the Dataset
  - Larger datasets can support higher-dimensional embeddings due to more training data.
  - Datasets with less than 100,000 sentences: Benefit from lower-dimensional embeddings (50-100 dimensions).
- NLP Task
  - High semantic accuracy tasks (e.g., sentiment analysis, machine translation): Benefit from higher-dimensional embeddings. Easier tasks (e.g., named entity recognition, part-of-speech tagging): May not require high-dimensional embeddings.
- Computational Resources
  In conclusion, to find the Right Number of Dimensions, we need to Experiment with different dimensionalities and evaluate model performance on a validation set and adjust dimensionality to find the optimal balance between semantic accuracy and computational efficiency for the specific use case. In phi3-mini, each vectors has 3072 dimension.

Assesment and other

- **GSM-8K**
  Exemple :
  Problème : If 3 pencils cost 6 dollars, how much do 7 pencils cost?
  Réponse : 7 pencils cost 14 dollars.
- **MedQA**
  Exemple :
  Question : What is the first-line treatment for hypertension in adults?
  Réponses :
  - a) Beta-blockers
  - b) ACE inhibitors
  - c) Calcium channel blockers
  - d) Diuretics

  Réponse correcte : b) ACE inhibitors
- bfloat16

Tokenizer

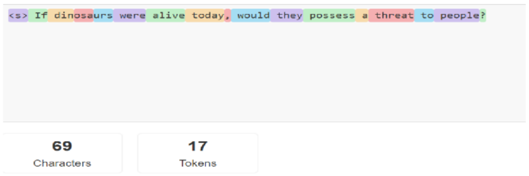


In [8]:
```
user_prompt_hf = "If dinosaurs were alive today, would they possess a threat to people?"
print(user_prompt_hf)
```

```
If dinosaurs were alive today, would they possess a threat to people?
```

In [13]:
```
print(tokenizer.eos_token)
print(tokenizer.encode(tokenizer.eos_token))
user_input_ids = tokenizer.encode(user_prompt_hf + tokenizer.eos_token, return_tensors='pt')

print(user_input_ids)
```

```
<|endoftext|>
[1, 32000]
tensor([[    1,   960,  4538,  3628,  1295,   892, 18758,  9826, 29892,   723,
           896, 22592,   263, 28469,   304,  2305, 29973, 32000]])
```

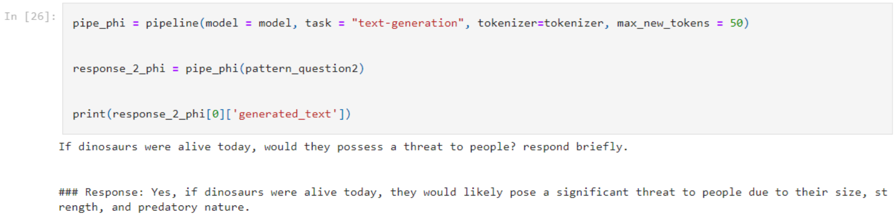


```
<s> If dinosaurs were alive today, would they possess a threat to people?
```

69 Characters

17 Tokens

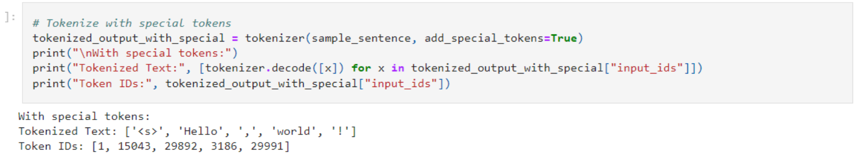
Exemple of response



```
In [26]:  pipe_phi = pipeline(model = model, task = "text-generation", tokenizer=tokenizer, max_new_tokens = 50)

          response_2_phi = pipe_phi(pattern_question2)

          print(response_2_phi[0]['generated_text'])
```

```
If dinosaurs were alive today, would they possess a threat to people? respond briefly.


### Response: Yes, if dinosaurs were alive today, they would likely pose a significant threat to people due to their size, st
rength, and predatory nature.
```

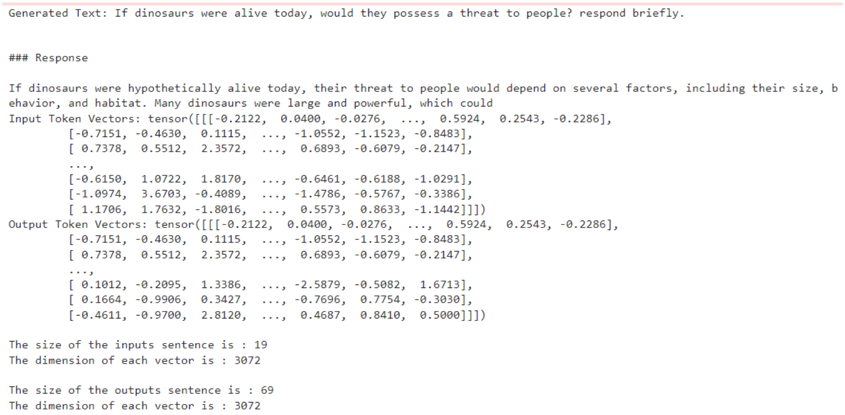
Test Tokenizer



```
]: # Tokenize with special tokens
   tokenized_output_with_special = tokenizer(sample_sentence, add_special_tokens=True)
   print("\nWith special tokens:")
   print("Tokenized Text:", [tokenizer.decode([x]) for x in tokenized_output_with_special["input_ids"]])
   print("Token IDs:", tokenized_output_with_special["input_ids"])
```

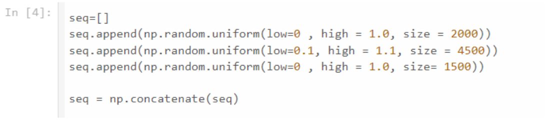
```
With special tokens:
Tokenized Text: ['<s>', 'Hello', ',', 'world', '!']
Token IDs: [1, 15043, 29892, 3186, 29991]
```

Word embeddings

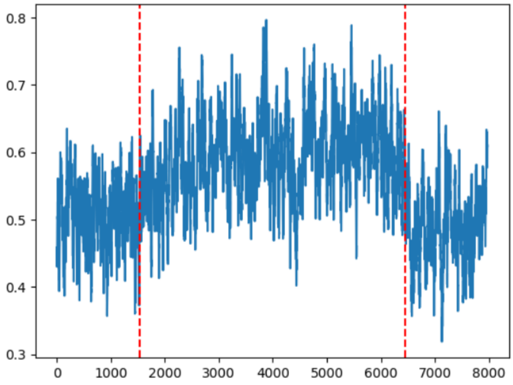


```
Generated Text: If dinosaurs were alive today, would they possess a threat to people? respond briefly.


### Response

If dinosaurs were hypothetically alive today, their threat to people would depend on several factors, including their size, b
ehavior, and habitat. Many dinosaurs were large and powerful, which could
Input Token Vectors: tensor([[[-0.2122,  0.0400, -0.0276,  ...,  0.5924,  0.2543, -0.2286],
         [-0.7151, -0.4630,  0.1115,  ..., -1.0552, -1.1523, -0.8483],
         [ 0.7378,  0.5512,  2.3572,  ...,  0.6893, -0.6079, -0.2147],
         ...,
         [-0.6150,  1.0722,  1.8170,  ..., -0.6461, -0.6188, -1.0291],
         [-1.0974,  3.6703, -0.4089,  ..., -1.4786, -0.5767, -0.3386],
         [ 1.1706,  1.7632, -1.8016,  ...,  0.5573,  0.8633, -1.1442]]])
Output Token Vectors: tensor([[[-0.2122,  0.0400, -0.0276,  ...,  0.5924,  0.2543, -0.2286],
         [-0.7151, -0.4630,  0.1115,  ..., -1.0552, -1.1523, -0.8483],
         [ 0.7378,  0.5512,  2.3572,  ...,  0.6893, -0.6079, -0.2147],
         ...,
         [ 0.1012, -0.2095,  1.3386,  ..., -2.5879, -0.5082,  1.6713],
         [ 0.1664, -0.9906,  0.3427,  ..., -0.7696,  0.7754, -0.3030],
         [-0.4611, -0.9700,  2.8120,  ...,  0.4687,  0.8410,  0.5000]]])

The size of the inputs sentence is : 19
The dimension of each vector is : 3072

The size of the outputs sentence is : 69
The dimension of each vector is : 3072
```

Uniform



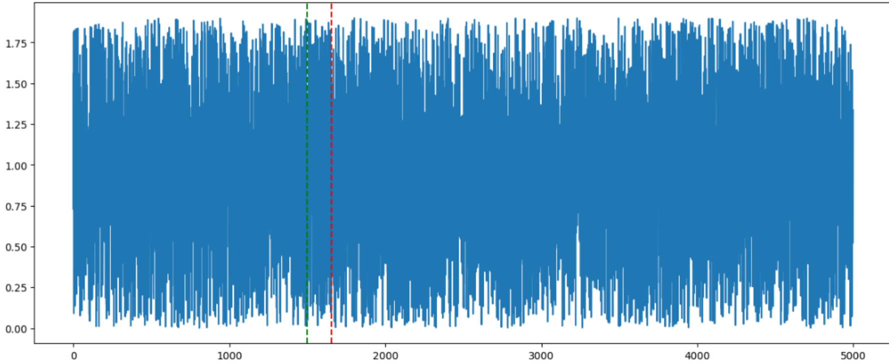
```
In [4]:  seq=[]
         seq.append(np.random.uniform(low=0 , high = 1.0, size = 2000))
         seq.append(np.random.uniform(low=0.1, high = 1.1, size = 4500))
         seq.append(np.random.uniform(low=0 , high = 1.0, size= 1500))

         seq = np.concatenate(seq)
```



Hidden Irrational Rotation

```
# Parameters
beta1 = 0.452341643253462432

beta2 = 0.6345354645623456234234
changepoint = 1500
t_total = 5000
process_count=2
min_distance = 0.15
```

```
[1655]
```



Questions :

- How can we adapt change points issues with Languages models ?
  - Maybe spotting changes in distributions can help segment long texts into several coherent parts (with the register used, the underlying emotion, for example).
  - Maybe can be used to adapt the learning parameters

References I

Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou.
Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions.
arXiv preprint arXiv:2309.07875, 2023.

Ronen Eldan and Yuanzhi Li.
Tinystories: How small can language models be and still speak coherent english?
arXiv preprint arXiv:2305.07759, 2023.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al.
Textbooks are all you need.
arXiv preprint arXiv:2306.11644, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
Advances in neural information processing systems, 30, 2017.