NLP and Deep Learning MAT3399

Lecture 8: Transformers

Tuan Anh Nguyen @ Aimesoft
ted.nguyen95@gmail.com

Content taken from Hugging Face NLP Course and Attention Is All You Need

What are Transformers?

Transformers: A type of neural networks architecture that achieve state-of-the-art performance on many NLP tasks (Even on some Computer Vision tasks as well).
Transformers are usually used as baselines for solving NLP problems nowadays.

```
from transformers import pipeline

classifier = pipeline("sentiment-analysis")
classifier("I've been waiting for a HuggingFace course my whole life.")
```

```
[{'label': 'POSITIVE', 'score': 0.9598047137260437}]
```

```
from transformers import pipeline

question_answerer = pipeline("question-answering")
question_answerer(
    question="Where do I work?",
    context="My name is Sylvain and I work at Hugging Face in Brooklyn",
)
```

```
{'score': 0.6385916471481323, 'start': 33, 'end': 45, 'answer': 'Hugging Face'}
```

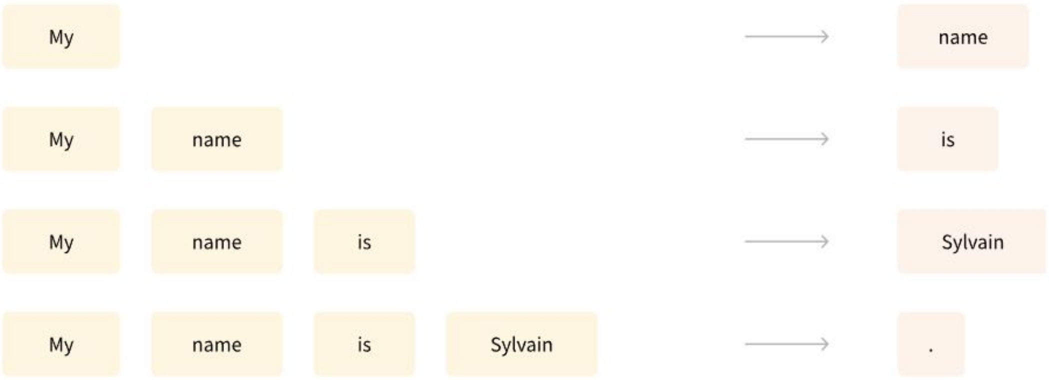
```
from transformers import pipeline

classifier = pipeline("zero-shot-classification")
classifier(
    "This is a course about the Transformers library",
    candidate_labels=["education", "politics", "business"],
)
```

```
{'sequence': 'This is a course about the Transformers library',
 'labels': ['education', 'business', 'politics'],
 'scores': [0.8445963859558105, 0.111976258456707, 0.043427448719739914]}
```

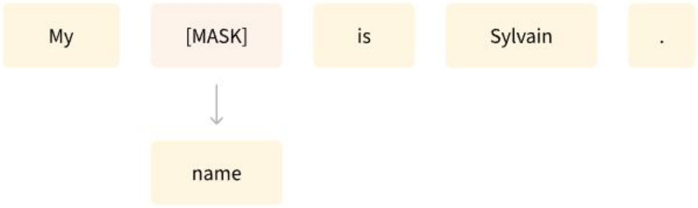
How Do Transformers Work?

TL;DR: Transformers are language models.

Causal Language Model

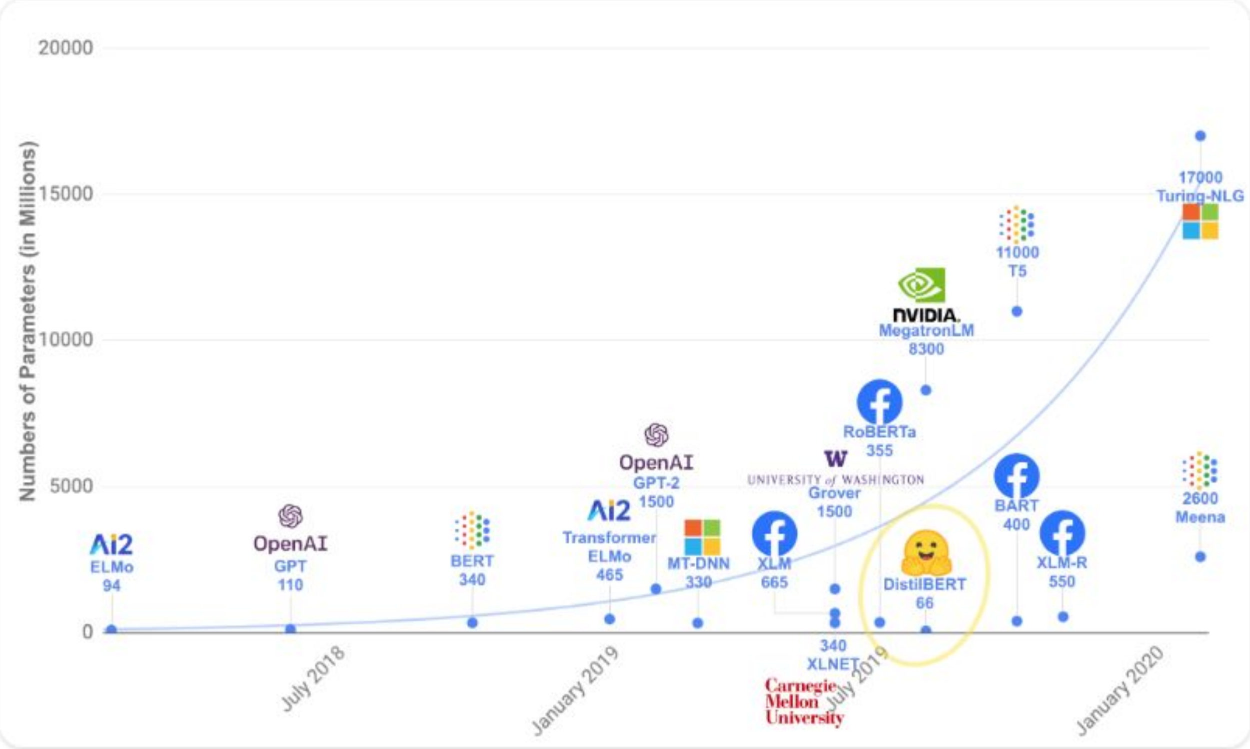


Masked Language Model



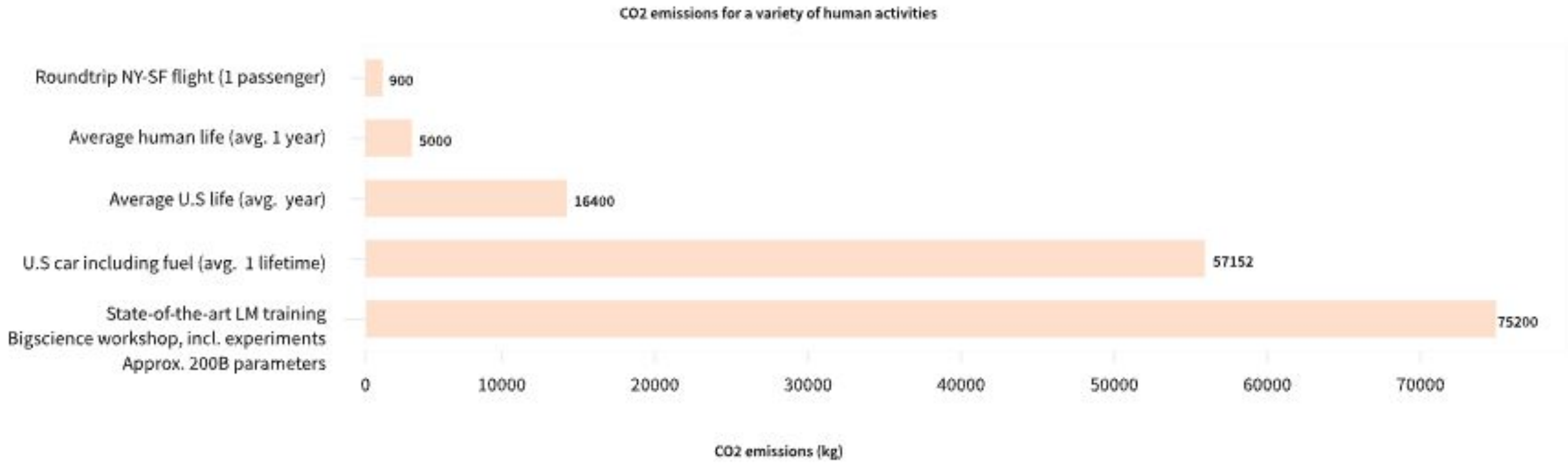
How Big are Transformers?

Transformers are trained on **HUGE** amount of data



How are these compared to your FFNN model and RNN from previous lessons?

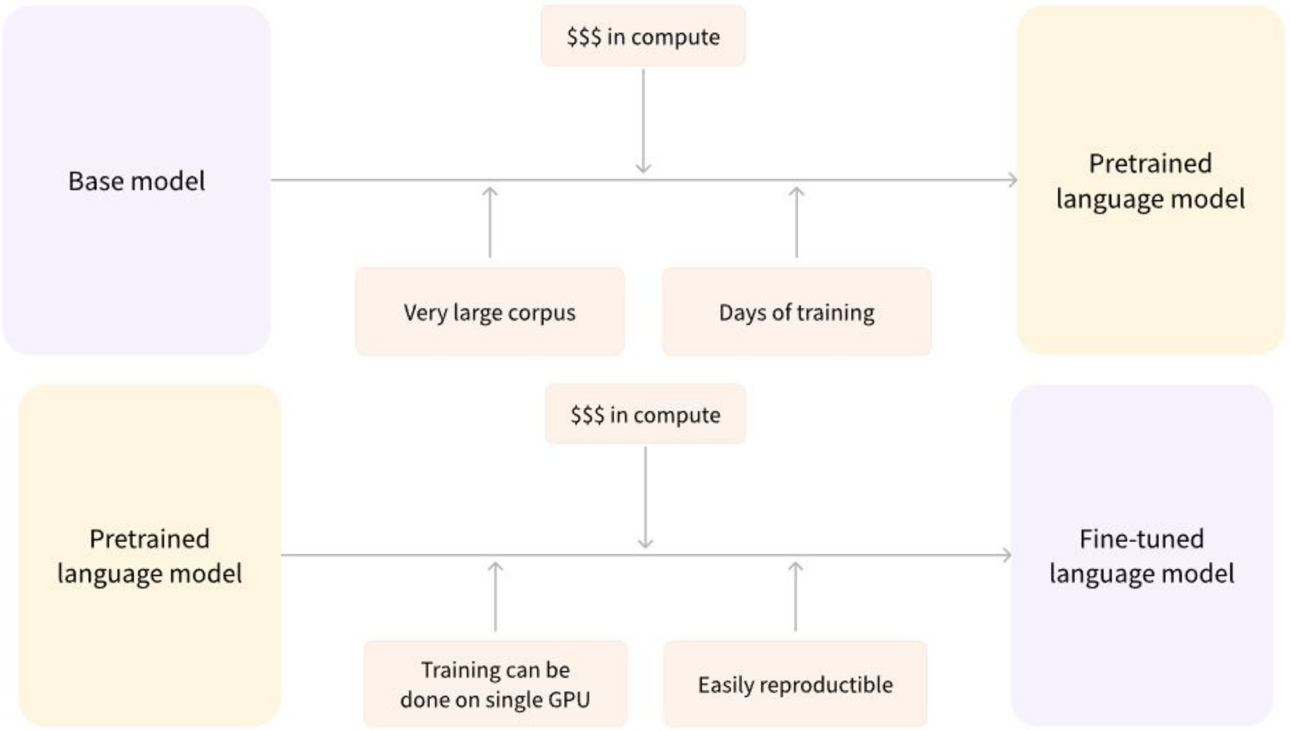
Training Transformers is Costly



=> We take advantage of pretrained models and transfer learning

Transfer Learning

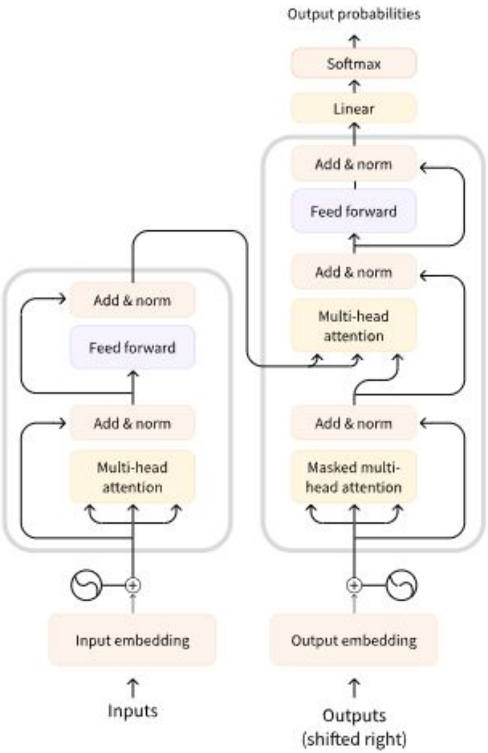
- The fine-tuning process is thus able to take advantage of knowledge acquired by the initial model during pretraining.
- Since the pretrained model was already trained on lots of data, the fine-tuning requires way less data to get decent results.
- For the same reason, the amount of time and resources needed to get good results are much lower.



General Architecture

- Encoder (left): The encoder receives an input and builds a representation of it (its features). This means that the model is optimized to acquire understanding from the input.
- Decoder (right): The decoder uses the encoder’s representation (features) along with other inputs to generate a target sequence. This means that the model is optimized for generating outputs.

Encoder-only models: sentence classification, named entity recognition.
Decoder-only models: text generation.
Encoder-decoder models: generative tasks that require an input, such as translation or summarization.

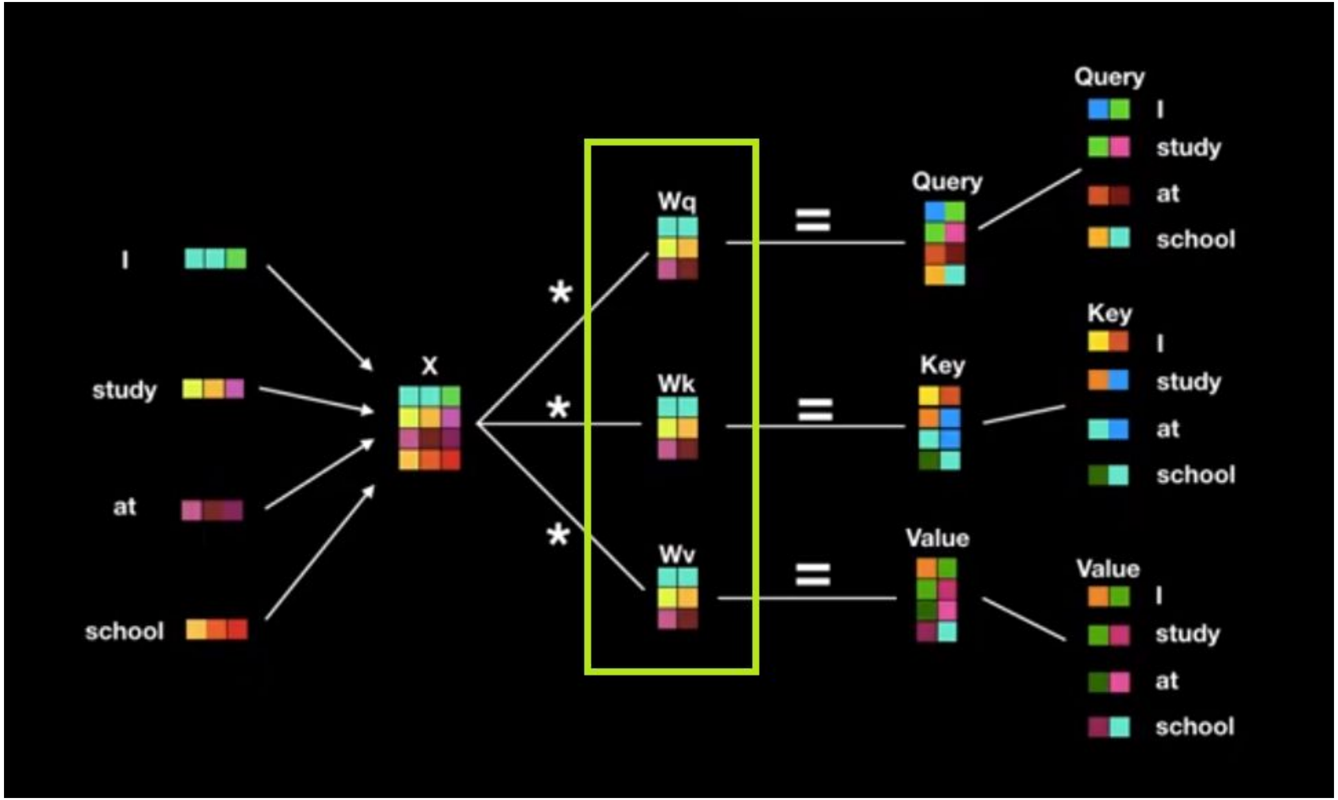


What makes transformers good? Answer: Attention

Self Attention

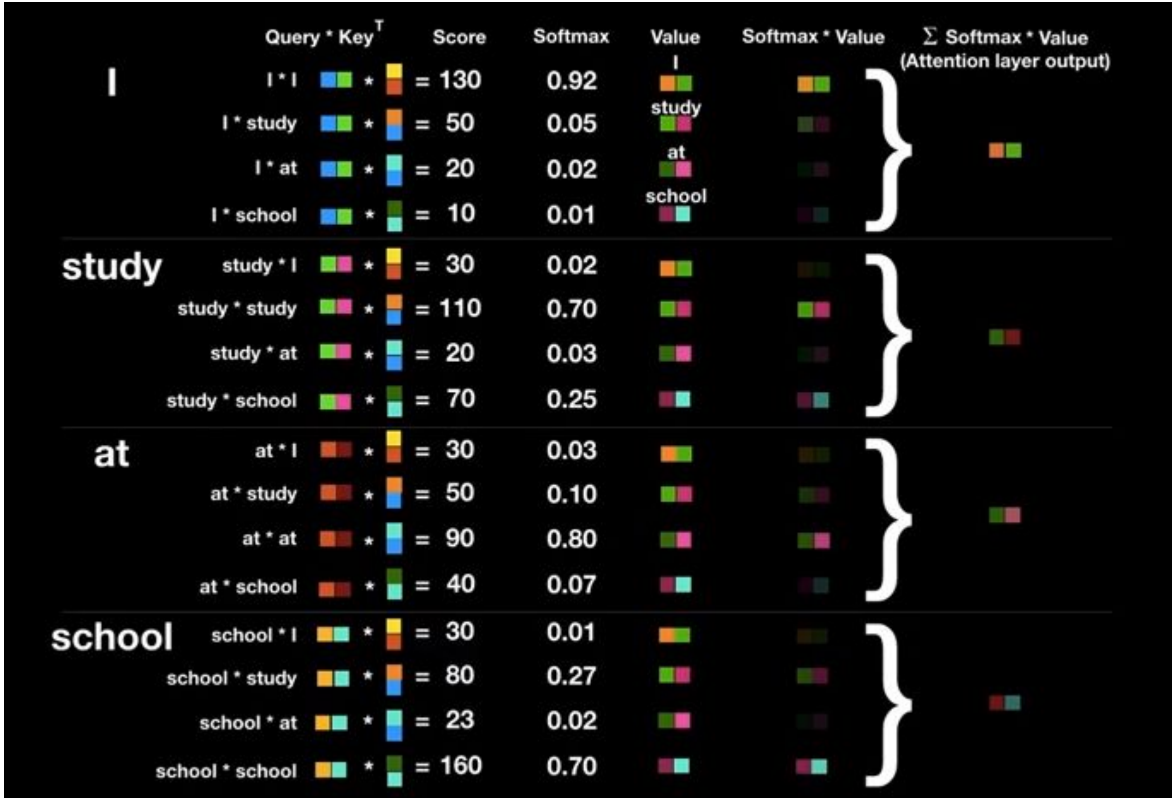
Wq, Wk, Wv are learnable parameters.

Query, Key, Value (or Q, K, V) are used to calculate final representation of the token.



Self Attention – Calculate Score for All Tokens

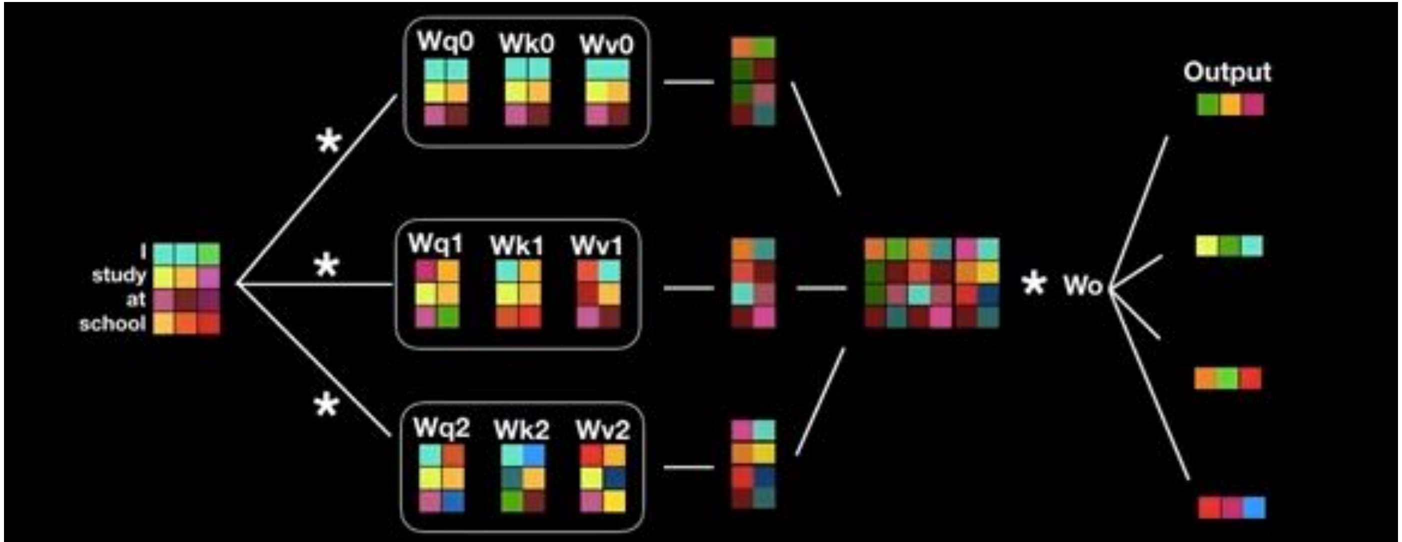
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



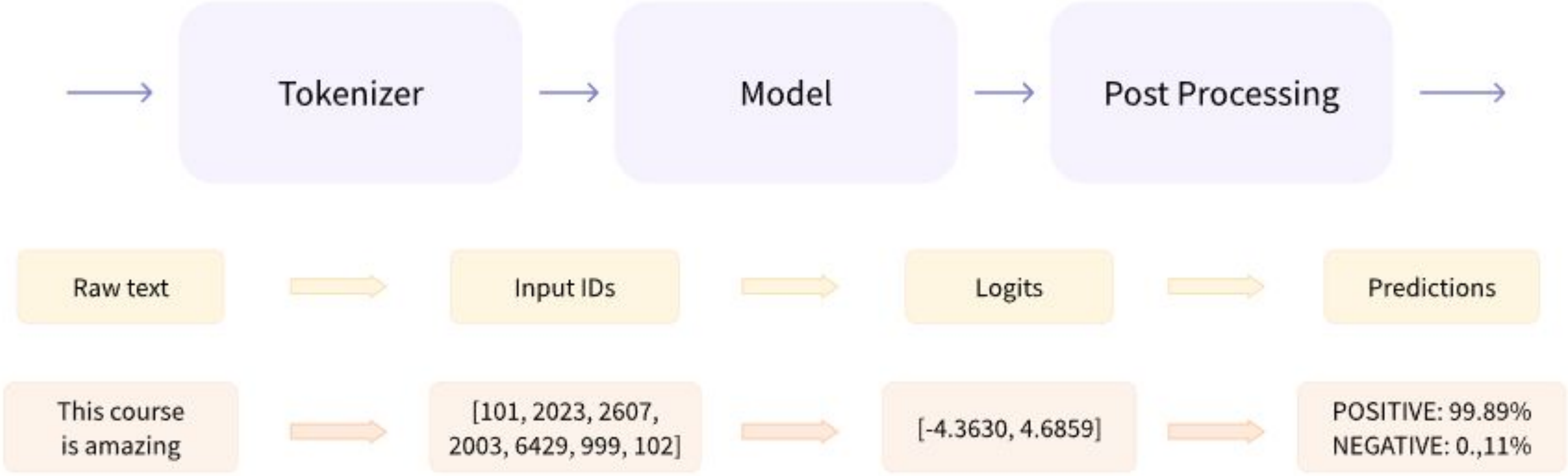
Multi-head Attention

Repeat the previous process multiple times, with different sets of (Wq, Wk, Wv), we have Multi-head attention.

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



The pipeline



The tokenizer in Transformers is not the same as the tokenizer you already learned from previous lesson

Outputs from Tokenizers

```
from transformers import AutoModel

checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
model = AutoModel.from_pretrained(checkpoint)
```

```
raw_inputs = [
    "I've been waiting for a HuggingFace course my whole life.",
    "I hate this so much!",
]
inputs = tokenizer(raw_inputs, padding=True, truncation=True, return_tensors="pt")
print(inputs)
```

```
{
    'input_ids': tensor([
        [  101,  1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172, 2607,  2026,  2878,  2166,
        [  101,  1045,  5223,  2023,  2061,  2172,   999,   102,     0,     0,     0,     0,     0,     0
    ]),
    'attention_mask': tensor([
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
    ])
}
```

Different models have different methods and different sets of tokens

Running the Model

```
from transformers import AutoModelForSequenceClassification

checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
model = AutoModelForSequenceClassification.from_pretrained(checkpoint)
outputs = model(**inputs)
```

```
print(outputs.logits)
```

```
tensor([[-1.5607,  1.6123],
        [ 4.1692, -3.3464]], grad_fn=<AddmmBackward>)
```

Coding Exercise

Try lxyuan/distilbert-base-multilingual-cased-sentiments-student on sentiment data and evaluate. Compare to your FFNN model.

Finetune `distilbert-base-uncased` using your sentiment dataset. Test and compare to the above model.

See how to finetune model:
https://huggingface.co/docs/transformers/v4.17.0/en/tasks/sequence_classification