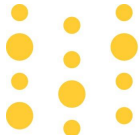




Course Study Group with a bent



Session 1

Hosted by  W&B

Agenda

We begin week 1 ***looking at sections 0 and 1.***

Our goal this week is to make sure you can run the code in these sections in colab and ***understand why transformers work well*** for a myriad of NLP tasks and how they work.

In addition to looking at the high-level **pipeline** API available in Hugging Face for getting predictions, ~~***we'll look at Blurr's high-level API for doing the same***~~ and more. (next week)

Resources and homework will be provided to improve your understanding of transformer based architectures.

Resources

Study Group registration page: <https://wandb.me/fastai-hf>

Study Group discord: <https://discord.gg/DsnRxSyt>

fastai:

1. The **fastai** course (<https://course.fast.ai/>)
2. The **Walk w/ fastai** course (<https://walkwithfastai.com/>)
3. The **FastBook** (available for purchase or free online via Jupyter notebooks)
4. The **FastBook** reading/study group form W&B (<http://wandb.me/fastbook>)

fastai + Hugging Face libraries:

1. AdaptNLP (<https://novetta.github.io/adaptnlp/>)
2. FastHugs (<https://github.com/morganmcg1/fasthugs>)
3. Blurr (<https://ohmeow.github.io/blurr/>)

ML/Data Science in general:

1. The **Chai Time Data Science** podcast (<http://youtube.com/c/chaitimedatascience>)
2. **Weights & Biases** (<https://wandb.ai/>)

0. Setup: **Setting up a working environment**

Use this if you want to keep it simple:

```
!pip install transformers[sentencepiece]
```

If you're going the local virtual environment front, consider using **conda/miniconda** and mamba:

<https://docs.conda.io/en/latest/miniconda.html>

<https://github.com/mamba-org/mamba>

<https://github.com/fastai/fastconda> (a channel to grab all things fastai, etc...)

Anybody using **poetry**? If so, I'd love to see a good example with fastai and hugging face :)

1. Transformer models: **Introduction**

Introduction

Transformer models

Using 🧡 Transformers

Fine-tuning a pretrained model

Sharing models and tokenizers

Diving in

The 🧡 Datasets library

The 🧡 Tokenizers library

Main NLP tasks

How to ask for help

Advanced

Specialized architectures

Speeding up training

A custom training loop

Contributing to Hugging Face

1. Transformer models: **Natural Language Processing**

What is it?

“NLP is a field of linguistics and machine learning focused on understanding everything related to human language.”

What are some common NLP tasks?

1. **Sequence Classification:** Classify whole documents (sentiment, is/not spam, is/not grammatically correct, whether two sentences are logically related)
2. **Token Classification:** Classify individual words (grammatical category like noun, verb, adj.), NER like person, location, organization)
3. **Text Generation:** Complete a prompt with auto-generated text, fill in the blank, given a sentence in one language translate it into another, summarize a larger document
4. **Extractive Q&A:** Given a question and a context, extract the answer based on the information provided in the context

1. Transformer models: Transformers, what can they do?

“The 🧐 [Transformers library](#) provides the functionality to create and use those shared models.”

“The [Model Hub](#) contains thousands of pretrained models that anyone can download and use.”

“The most basic object in the 🧐 Transformers library is the **pipeline**. *It connects a model with its necessary preprocessing and postprocessing steps, allowing us to directly input any text and get an intelligible answer.*” The three main steps it does is:

1. Preprocess the text into a format the model can understand.
2. Pass the preprocessed inputs to the model.
3. Post-process the predictions of the model so they are humanly understandable.

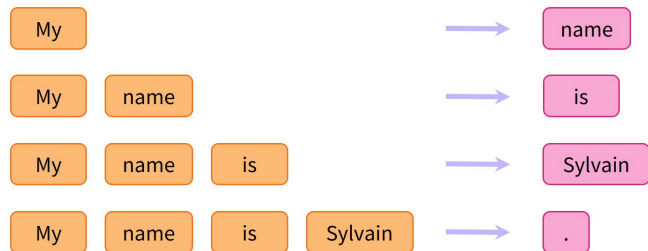
The **pipeline** function is the highest level of the Transformers API ... it *returns an end-to-end object that performs a specific NLP task on one or several texts.*

“All the models can be tested directly through your browser using **the Inference API**” (Let’s see how!)

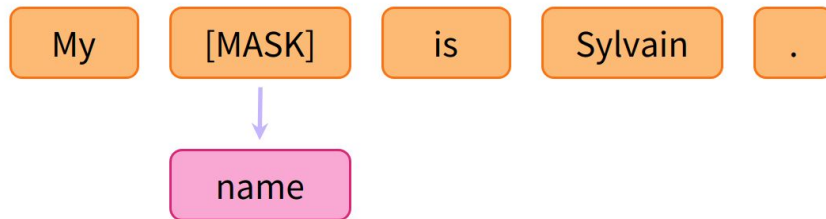
1. Transformer models: **How do Transformers work?**

“All the Transformer models mentioned above (GPT, BERT, BART, T5, etc.) have been *trained as language models*.”

Causal LM



Masked LM



To use these models for specific tasks (like NER, classification, summarization, etc...), we use **transfer learning** where the pre-trained language models are **fine-tuned** “in a supervised way” for the specific task

1. Transformer models: **How do Transformers work?**

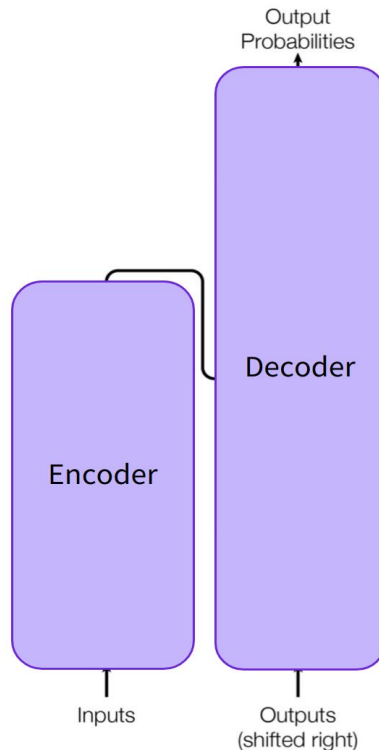
Encoder (left): The encoder receives an input and builds a representation of it (its features). This means that the model is ***optimized to acquire understanding from the input.***

Converts text into numerical representations, i.e. embeddings/features using self-attention as its main component. It is bi-directional. The output is a high level representation of the inputs.

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.***

During training, it “can also accept text inputs”, and uses masked self-attention but is unidirectional and typically used in an autoregressive manner

In seq2seq, **encoder/decoder** model, the inputs are the high level representation outputs of the encoder alongside other inputs to generate a prediction.



1. Transformer models: **How do Transformers work?**

Attention Layers

“... this layer ***will tell the model to pay specific attention to certain words in the sentence you passed it (and more or less ignore the others) when dealing with the representation of each word*** a word by itself has a meaning, but that meaning is deeply affected by the context, which can be any other word (or words) before or after the word being studied.”

Example: “The animal didn't cross the street because it was too tired ”

What does “it” in this sentence refer to? Is it referring to the street or to the animal? It’s a simple question to a human, but not as simple to an algorithm.

When the model is processing the word “it”, self-attention allows it to associate “it” with “animal”.

As the model processes each word (each position in the input sequence), self attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding for this word. (from Jay Alammar’s “Illustrated Transformer” article)

1. Transformer models: **How do Transformers work?**

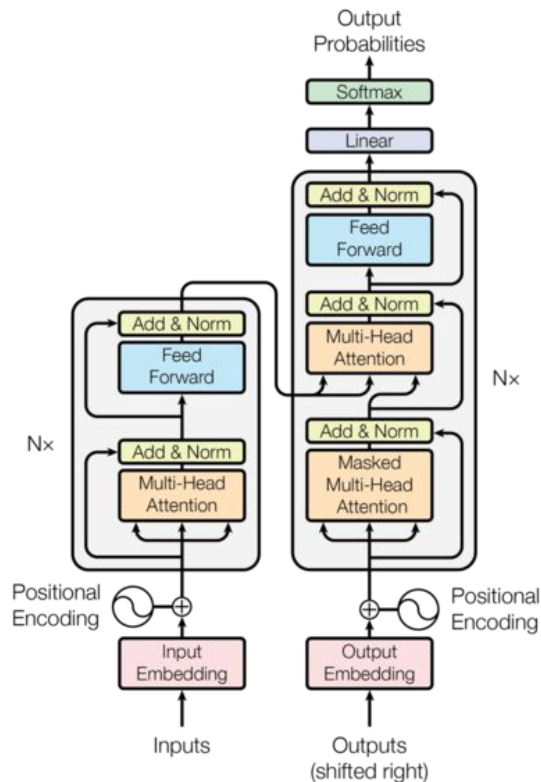
The original architecture

Attention Is All You Need

<https://arxiv.org/abs/1706.03762>

The Illustrated Transformer

<https://jalammar.github.io/illustrated-transformer/>



1. Transformer models: **Encoder models**

“Encoder models use only the encoder of a Transformer model. At each stage, ***the attention layers can access all the words in the initial sentence***. These models are often characterized as having ***“bi-directional” attention***, and are often called ***auto-encoding models***.”

When to use

“Good at extracting meaningful information ... Natural Language Understanding tasks.” When you have a task that requires both semantic and syntactic understanding. Tasks such as text classification, token classification, extractive Q&A, MLMs

Examples of Encoder models:

[ALBERT](#), [BERT](#), [DistilBERT](#), [ELECTRA](#), [RoBERTa](#)

1. Transformer models: **Decoder models**

“Decoder models use only the decoder of a Transformer model. At each stage, for a given word ***the attention layers can only access the words positioned before*** it in the sentence. These models are often called ***auto-regressive models***.”

Key difference: Uses “masked self-attention” which only attends to the words previous to it ... a mask is used to hide the context after.

When to use

For causal tasks; generating sequences ... Natural Language generation. Tasks such text generation.

Examples of Decoder models:

[CTRL](#), [GPT](#), [GPT-2](#), [Transformer XL](#)

1. Transformer models: **Sequence-to-sequence models**

“**Encoder-decoder models** (also called **sequence-to-sequence models**) use both parts of the Transformer architecture. At each stage, the attention layers of the encoder can access all the words in the initial sentence, whereas the attention layers of the decoder can only access the words positioned before a given word in the input.”

When to use

For sequence to sequence tasks; where input distribution is different from output distribution ... output length is independent of input length. They often do not share weights. Tasks such as: summarization, translation, generative Q&A

Examples of Seq2Seq models:

[BART](#) / [mBART](#), M2M100, [Marian](#)MT, Pegasus, ProphetNet, [T5](#) / mT5

1. Transformer models: **Bias and limitations**

“When you use these tools, you therefore need to keep in the back of your mind that the original model you are using could very easily generate sexist, racist, or homophobic content. Fine-tuning the model on your data won’t make this intrinsic bias disappear.”

Bias **can result from the data** the pre-trained or fine-tuned model is trained on.

Bias **can result from the metric** the model was optimized for.

→ *See chapter 3 of the FastBook for more information on data ethics and mitigation strategies*

Homework

1. Read the **Attention Is All You Need** paper (and Jay Alammar's "*The Illustrated Transformer*")
2. Read the **BERT** paper (and Jay Alammar's "*The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*")
3. Get acquainted with one or more of the fastai Hugging Face libraries; see if you can build a basic classification model of some sort