NIHR | Applied Research Collaboration South West Peninsula

@penCHORD_UoE

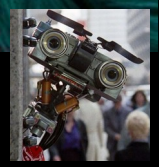
@peninsula_ARC

HSMA Masterclass

BERT for Natural Language Processing

Masterclass Guru : Dr Daniel Chalk

“Can you tell me how to get, how to get a brand new GPU?”



HSMA 5

#hsma5isalive

NIHR | Applied Research Collaboration South West Peninsula

@penCHORD_UoE

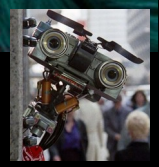
@peninsula_ARC

HSMA Masterclass

BERT for Natural Language Processing

Masterclass Guru : Dr Daniel Chalk

“Can you tell me how to get, how to get a brand new GPU?”



HSMA 5

#hsma5isalive

Recommended Reading

BERT is not an easy thing to understand, but this introduction to BERT helped me really understand the concepts of how it works :
https://huggingface.co/blog/bert-101

As well as this one :
https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

And the sentiment analysis example we'll look at today is based on this tutorial :
https://www.datacamp.com/tutorial/tutorial-natural-language-processing

What is BERT?

BERT (Bidirectional Encoder Representations from Transformers) is a Machine Learning model that was developed in 2018 by Google.

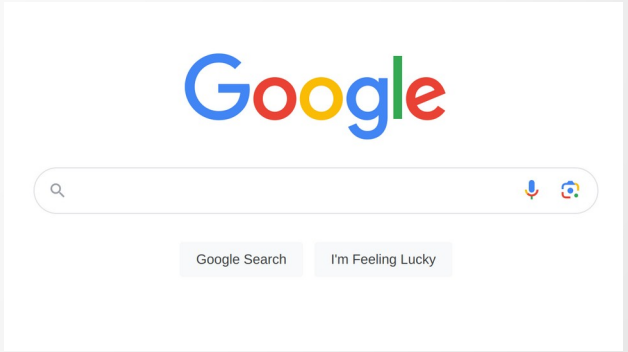
It has transformed the world of Natural Language Processing and is now used extensively for NLP tasks – you have likely interacted with BERT without even knowing it!

In this session, we will introduce BERT, explain a bit about how it works, and show you a simple example of BERT in action.

What can BERT be used for?

BERT can be used for a wide range of NLP applications, such as :

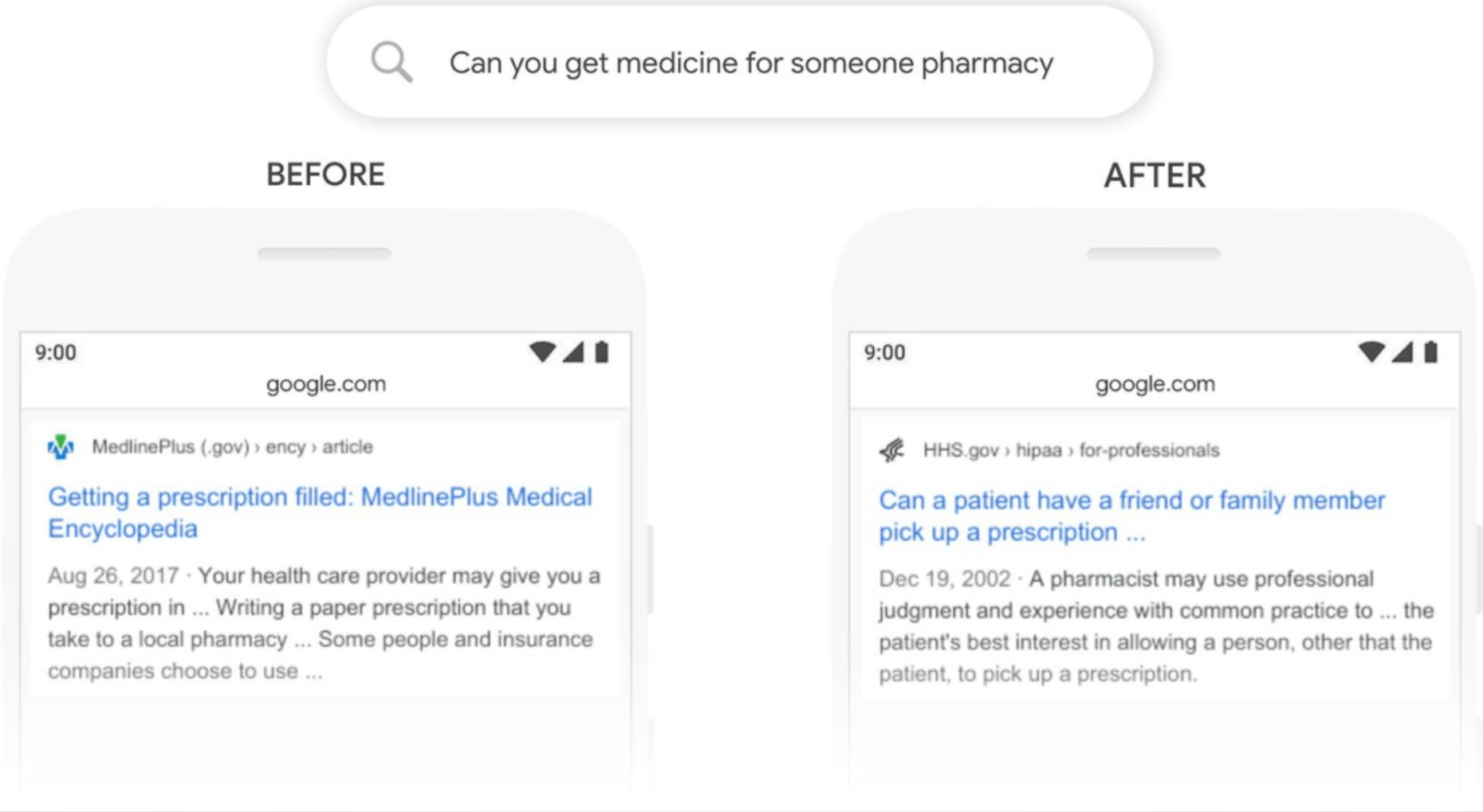
- Sentiment Analysis
- Named Entity Recognition
- Generating natural language text
- Answering questions



And more…

Today, we'll look at an example of BERT in action on Sentiment Analysis, and compare its performance on the same data we used earlier in the course.

It’s all about context...



https://blog.google/products/search/search-language-understanding-bert/

Large Language Models

BERT is an example of a *Large Language Model*. These are models that have been trained on extremely large corpuses of text data, and can generate and understand natural language extremely well thanks to their *Transformer-based Architecture* (more on that shortly).

GPT-2 and GPT-3 (upon which Chat-GPT sits to provide a “chatbot” interface) are other examples of large language models.

The original BERT model was trained on 3.3 billion words from Wikipedia and Google’s BookCorpus

Bidirectional

Let’s now start to unpick the BERT acronym to explain how it works.

Bidirectional Encoder Representations from Transformers

To understand what we mean by “bidirectional” we need to understand that BERT uses something known as *Masked Language Modelling (MLM)*.

Let’s play...

To explain Masked Language Modelling, let’s play a quick game of...


JOHN
BERYL
ARTHUR
SPIKE
BLANKETY
BLANK

Blankety Blank

Fill in the blank and put your answer in the chat :

“Delilah cut off Samson’s hair to sap his strength, but I just heard that first she cut off his **BLANK** to slow him down”

Go…

Blankety Blank

Fill in the blank and put your answer in the chat :

“Delilah cut off Samson’s hair to sap his strength, but I just heard that first she cut off his **BLANK** to slow him down”

Go…

Blankety Blank

“Delilah cut off Samson’s hair to sap his strength, but I just heard that first she cut off his **BLANK** to slow him down”

Let’s look at your answers...

Blankety Blank

My answer was : “legs”

“Delilah cut off Samson’s hair to sap his strength, but I just heard that first she cut off his **legs** to slow him down”

Blankety Blank

Fill in the blank and put your answer in the chat :

"My partner cooks for me every night. It's not that I don't appreciate the effort, but I wish they knew how to turn the **BLANK** on."

Go...

Blankety Blank

Fill in the blank and put your answer in the chat :

"My partner cooks for me every night. It's not that I don't appreciate the effort, but I wish they knew how to turn the **BLANK** on."

Go...

Blankety Blank

"My partner cooks for me every night. It's not that I don't appreciate the effort, but I wish they knew how to turn the **BLANK** on."

Let’s look at your answers...

Blankety Blank

My answer was “oven”

"My partner cooks for me every night. It's not that I don't appreciate the effort, but I wish they knew how to turn the **oven** on."

Masked Language Modelling

A Masked Language Model is one which is trained to predict a “masked” (hidden) word in a sentence given the context of the words surrounding it.

It does this by looking in *both* directions (both before and after the missing word) – this is what we mean by *bidirectional*.

The words selected to be masked are normally random, and a % of the total words – a random 15% in the case of BERT.

Masked Language Modelling

You just did this when we played Blankety Blank :

“Delilah cut off Samson’s hair to sap his strength, but I just heard that first she cut off his **BLANK** to slow him down”

BEFORE : she cut off his hair (part of his body) to sap strength, and cut off something else
AFTER : the something else she cut off was to slow him down.

So it’s a part of the body, that would slow him down. Therefore we conclude it is “legs”.

Masked Language Modelling

You just did this when we played Blankety Blank :

"My partner cooks for me every night. It's not that I don't appreciate the effort, but I wish they knew how to turn the **BLANK** on."

BEFORE : partner cooks every night, but there's an indication that it's not entirely without issue, because they don't know how to turn something...
AFTER : ...on

So it's something they need to turn on to cook (and there's a possible note of sarcasm, so it might be something basic). Could it be, an "oven"?

Masked Language Modelling

As well as all the times we play Blankety Blank with friends (surely at least twice a week), we also do this :

- in general conversation (if we didn't hear something, we'll try to fill in the blank)
- when we're reading (we'll often read quickly and our brain is doing a bit of work filling in the blanks)

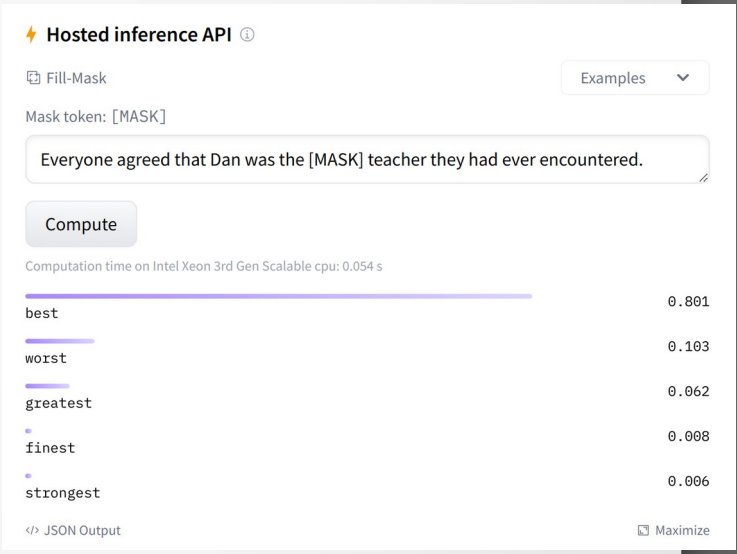
But, just like a Masked Language Model (or a Blankety Blank panelist) we won't always predict correctly...

BERT Masking API

You can play around with BERT's masking predictions using the API at the following link :

https://huggingface.co/bert-large-uncased-whole-word-masking

This allows you to input a sentence and blank out part of it (using the token [MASK]) – you can then see what BERT predicts the missing word to be, along with the associated probabilities (level of confidence) of each of the five top answers.



Next Sentence Prediction

As well as Masked Language Modelling, BERT is also trained on *Next Sentence Prediction (NSP)*. In NSP, the model is shown pairs of sentences where in half of them the second sentence correctly follows the first, and in half of them it doesn't.

For example :
- "Dan bought a new computer. It was much faster than his old one." (CORRECT SENTENCE PAIR)
- "Dan had a late lunch on Tuesday. Donkeys are not native to Canada."

BERT is trained 50/50 on MLM and NSP.

Transformers

Let's next turn our 'attention' (you'll be laughing about that in a moment) to the 'Transformers' bit of the acronym

Bidirectional Encoder Representations from **Transformers**

And to do that, we need to learn about the concept of *attention*. Or rather, cast our minds back to the Named Entity Recognition training when we discussed attention.

Were you paying attention? ;)

Pay Attention!

Let’s study the following paragraph :

My cat is called Prince, and he’s very demanding. Every day, he asks for treats at 1100 (his “elevenses”). You could set your watch by him, so accurate is he in asking for them on time. Sometimes, when I’m teaching, he has to wait for them, and he gives me a look that he’s clearly not happy about this.

Pay Attention!

As human beings, when we read things, we relate the things we read back to stuff we read earlier in the text :

My cat is called Prince, and he's very demanding. Every day, he asks for treats at 1100 (his "elevenses"). You could set your watch by him, so accurate is he in asking for them on time. Sometimes, when I'm teaching, he has to wait for them, and he gives me a look that he's clearly not happy about this.

Pay Attention!

But the old Convolutional Neural Network approach to NLP doesn't allow for this. It looks at a bit of the text at a time – so the closeness of words to the thing you are trying to classify / predict becomes important.

Arguably, it becomes *too* important, and it becomes poor at identifying *long-range dependencies* (words that relate back to things much earlier in the text).

An alternative approach is to use **Attention**.

Pay Attention!

With Attention, we can put portions of text against each in a matrix, and represent the relationships of each word with each other.

This is useful for translation :

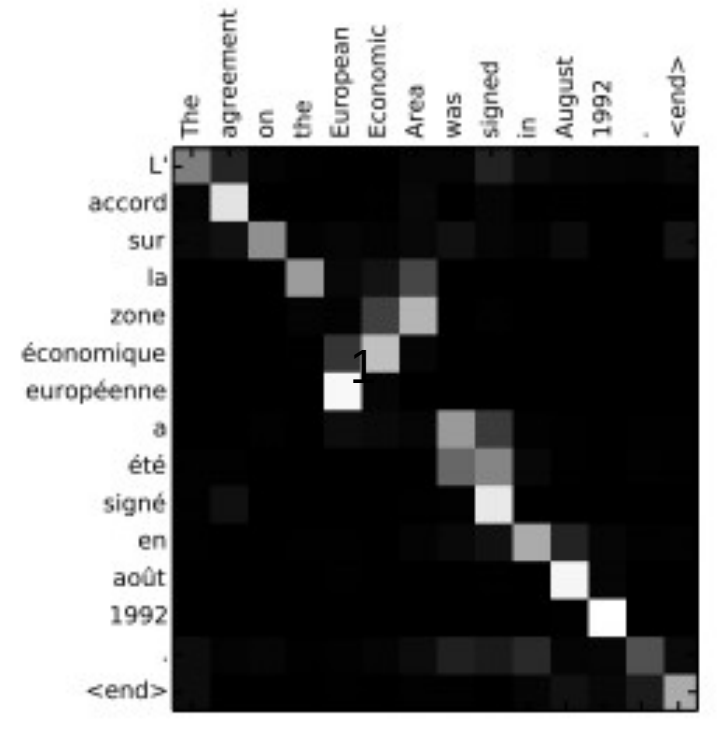


Image from https://wiki.pathmind.com/attention-mechanism-memory-network

But we can also put the same portion of text on each axis, and represent the relationships of words with other words in the text (*Self-attention)*. This emulates what we're doing as humans when we read some text.

Word Meanings and Context

Consider the following sentence :

“John asked Mary if he could borrow her pen.”

As human beings, it’s likely we’ll assume that the thing John wants to borrow from Mary is a writing implement, rather than a female swan, or an animal enclosure, even though the same word (“pen”) would be used in all of these cases. Sometimes some additional text can help us understand the context :

“This is because John was busy writing his memoirs.”

Word Meanings and Context

In natural language, the same word can be used in different places in the same text to mean very different things – this is known as *polysemy*. The thing that helps us understand is the **context** of the word in the sentence / wider text.

Clues within the text (and external to it – common phrases etc) can help us to understand the context in which a word (and therefore the sentence) should be interpreted.

Transformers

Transformers use *Attention* to try to understand the context of a word, and then *encode* that context-information alongside the word.

This means that, unlike a Convolutional Neural Network, which only looks at words in the context of words that surround them within a defined proximity, Transformers can pull information from *anywhere* in the text to try to understand the context of a word or words.

This makes them very powerful tools for NLP but also deep learning more generally.

> *"Since their introduction in 2017, Transformers have rapidly become the state-of-the-art approach to tackle tasks in many domains such as natural language processing, speech recognition, and computer vision. In short, if you're doing deep learning, then you need Transformers!"*

Lewis Tunstall, Hugging Face ML Engineer & Author of Natural Language Processing with Transformers

The Encoder

We know that Transformers (robots in disguise?) use attention to link things up to give context. But they don't want to link up *everything* because most things won't matter. Instead, they use *weights* to indicate which aspects (in NLP, words in our text) are most important.

They do this by using an *Encoder*.

Bidirectional **Encoder Representations** from Transformers

(Note : this is a nice explanation of Encoders and Decoders in Transformers : https://kikaben.com/transformers-encoder-decoder/)

The Encoder

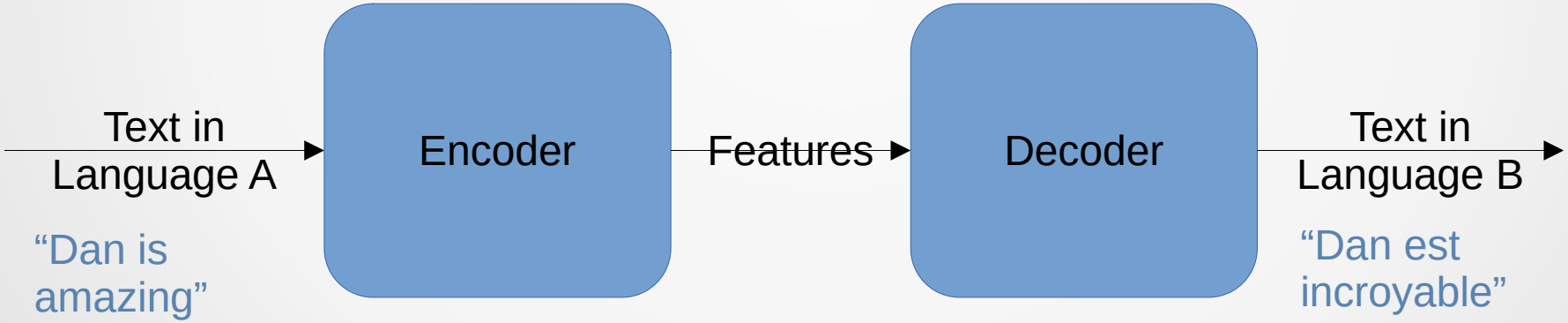
In a Transformer, an Encoder takes raw text input (which has been converted into integers by a pre-processing model), and extracts features from the input text.

In most Transformer models (such as BERT) the Encoder is not a single Encoder, but actually a number of Encoder layers, where the output from the final layer represents the features.

The features extracted by BERT include the *self-attention* structure we talked about before, which allows it to determine the weights for these self-linkages, to work out what's most important for determining context.

Decoders

In a Transformer model, a *decoder* takes the outputs from the *encoder* (ie the relevant features of the text that have been extracted) and uses this to generate a text output. Therefore, you can use Transformers to undertake translation for example :



However, BERT does *not* use a Decoder (just an Encoder), because it is a *pre-trained model* that does not generate outputs such as language generation "out of the box". Instead, we use something known as *fine-tuning*.

Fine-tuning

A BERT model (there are many, not just the original) is pre-trained on large corpuses of data and is able to predict context data of text using the encodings it learned during training. This is important because the training of a BERT model takes huge amounts of computational resource.

When we come to use a BERT model for our task, we take that training and apply *fine-tuning*. That is, we get the initial pre-trained model tuned to our particular problem and application (and get it to learn the nuances of our own data / application area).

We can either build this fine-tuning ourselves, or use one of the many FOSS pre-trained models that are already fine-tuned for certain tasks, such as Twitter sentiment analysis (https://huggingface.co/finiteautomata/bertweet-base-sentiment-analysis), toxic comment detection (https://huggingface.co/unitary/toxic-bert?) and many more…

There are over 220,000 models here : https://huggingface.co/models. Just be aware the quality may vary significantly, so investigate thoroughly anything you choose to use.

GPUs and TPUs

The original training of a BERT model is incredibly computational resource intensive. However, although much less complex, even fine-tuning BERT models is only practical if you have access to a CUDA-enabled GPU (Graphics Processing Unit) or a TPU (Tensor Processing Unit).

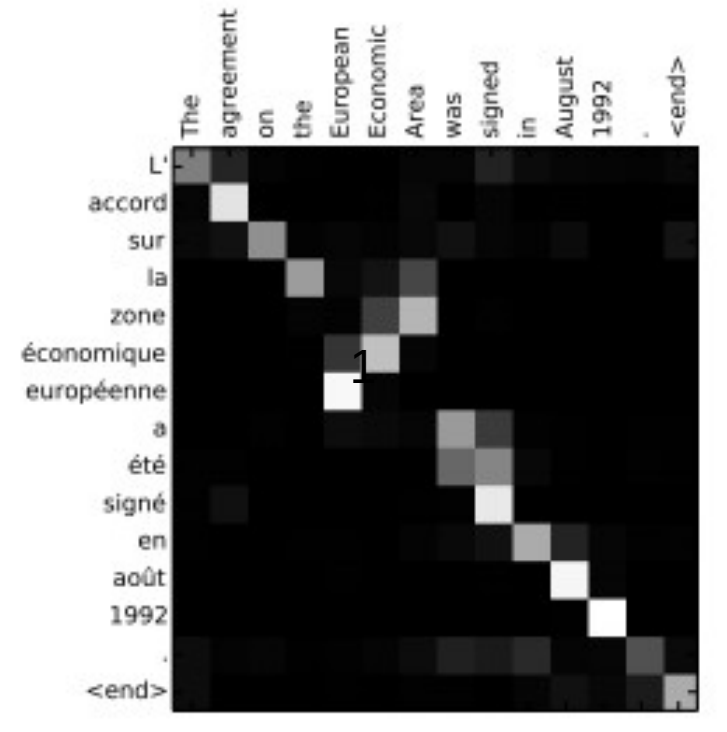
You may have access to a GPU locally (in your own machine) or you can use a cloud-based service (such as Google CoLab) to access such resources (but be aware, reliable access to good cloud resources requires money).

Summary

That was a lot of information to take in. So let’s summarise...

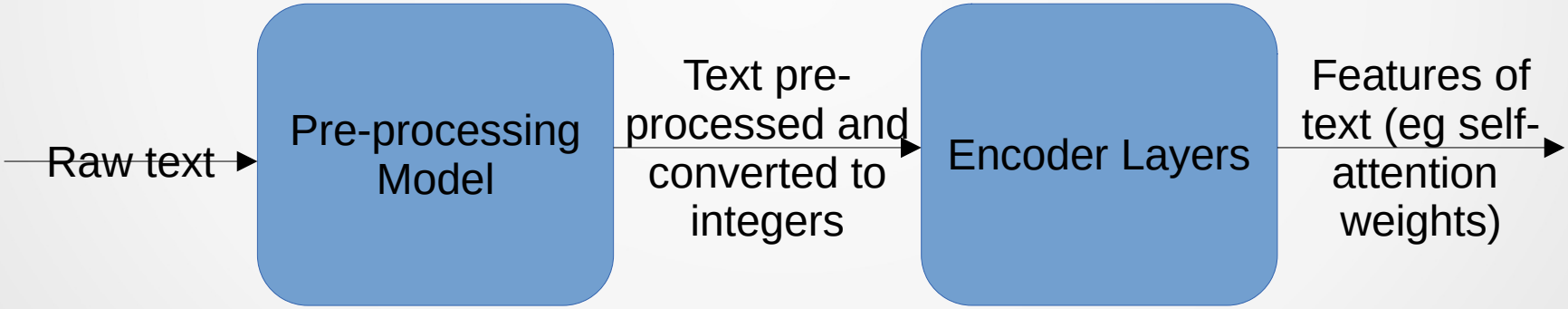
Summary

BERT models are pre-trained on huge amounts of text, to try to understand context using the concept of self-attention. They use the Encoder mechanism from Transformers, without the Decoder bit.



Summary

Encoders take in pre-processed text (as numbers), extract features (such as self-attention weights) and spit them out the other side. BERT uses multiple layers of encoder.



Summary

BERT models are trained using a combination of Next Sentence Prediction and Masked Language Modelling (playing Blankety Blank) to get good at extracting these text features

Summary

BERT models take huge amounts of computational resource and time to learn. So we first take a pre-trained BERT model off the shelf...

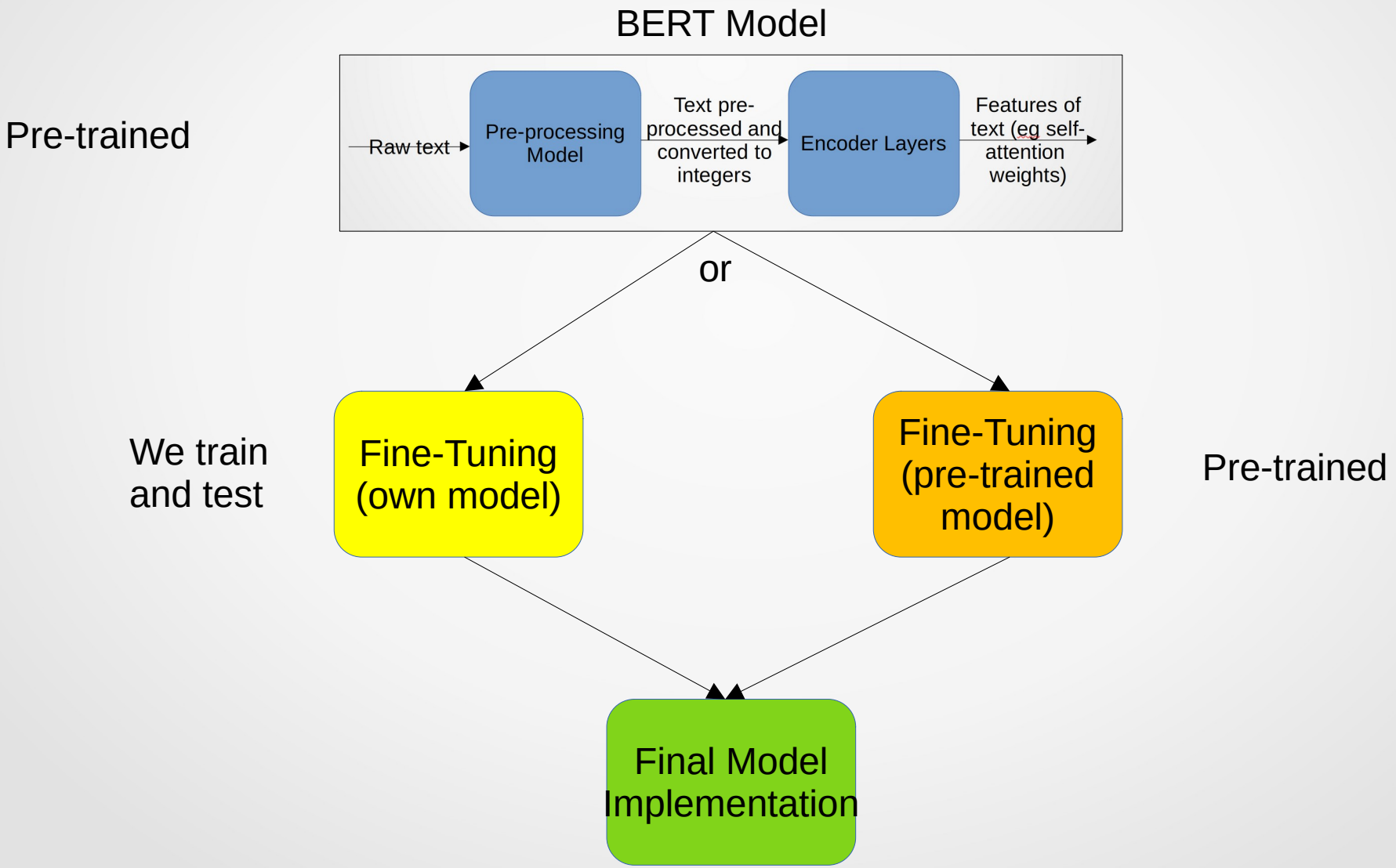
Summary

…then we use that initial pre-trained model which has all the valuable context information learned, and we fine-tune it for our particular application and problem area (eg determining the sentiment of movies).

Summary

We can do that either by building and training our own fine-tuning model (which we bolt on to the pre-trained BERT model) or using another pre-trained model for the fine-tuning.

Summary


BERT Model
Pre-trained
Raw text
Pre-processing Model
Text pre-processed and converted to integers
Encoder Layers
Features of text (eg self-attention weights)
or
We train and test
Fine-Tuning (own model)
Fine-Tuning (pre-trained model)
Pre-trained
Final Model Implementation

Bias

Cast your minds back to the “Ethics in AI” session. You’ll recall we talked about how because large language models (such as BERT) go out into the online world and learn from vast swathes of text data, they also soak up all the biases and unpleasantness that’s out there too…

Let’s look at an example of that in action.

We’ll use a simple piece of code to do this (bert_mlm.py) although we could also use the online tool at Hugging Face we mentioned earlier (https://huggingface.co/bert-large-uncased-whole-word-masking)

ELMo

Around a year before BERT, there was another encoder model that made quite a storm – ELMo (Embeddings from Language Model). Yes, they really do like Sesame Street…

ELMo works slightly differently – it doesn't use Masked Language Modelling (Blankety Blank). It *is* Bidirectional, but not in the same way – it basically has a model for text before a word, and a model for text after a word, and joins the models together. But that doesn't allow for looking in both directions simultaneously, which is how BERT works (ie gathering information on context from both directions to make a prediction).

But it was also a very impressive encoder model that could deal with polysemy, like BERT.

Exercise 1

After a 10 minute break, you'll split into your groups and spend 30 minutes having a play around with the two ways in which BERT learns – Masked Language Modelling (MLM) and Next Sentence Prediction (NSP) :

1. For the MLM, you can either use the online API at Hugging Face (see earlier slide), or use the file bert_mlm.py (be aware a ~500mb download will be started the first time you run it). Try different sentences with different words blanked out, and see what BERT predicts. Perhaps explore potential bias too in the learning – what insights do you generate? **Note – for this .py file, use the bert_mc_cpu environment.**

2. For the NSP, use the file bert_nsp.py (this will also require a ~500mb download unless you've already run it for the other file). Read through the code as a group and make sure you understand how it's working. Run it and observe the outputs. Try different pairs of sentences – some where the pair of sentences follow-on, some where they don't – and see how well BERT performs. **Note – for this .py file, use the bert_torch_cpu environment.**

BERT Code Walkthrough

We'll now walk through a piece of code that uses BERT for Sentiment Analysis. Specifically, we'll fine-tune and use a BERT model to predict the sentiment of movie reviews using exactly the same dataset we used in the Sentiment Analysis training (where we built our own Neural Network from scratch).

Let's see how BERT performs compared to that model we built previously.

The files we are walking through are
bert_datacamp_tutorial_part_1.py and
bert_datacamp_tutorial_part_2.py

NOTE : unless you have access to a CUDA-enabled GPU, you will not be able to run this code in a reasonable amount of time.

Exercise 2

For the remainder of the session, you'll split into your groups. Your task is to compare the practical Sentiment Analysis performance across three models :

- The BERT Model trained over 5 epochs (bert_datacamp_tutorial_part_2.py loading in the imdb_bert_model_5_epochs model)
- The BERT Model trained over 2 epochs (bert_datacamp_tutorial_part_2.py loading in the imdb_bert_model_2_epochs model)
- The basic Neural Network model we built in the Sentiment Analysis session (load_sa_session_model.py which loads in the model_from_sa_session model)

You should try different movie reviews and compare them across the three models. For the BERT models, you specify the reviews in the examples list in the bert_datacamp_tutorial_part_2.py code (recommend you do one at a time for this), then run the code, select the model you want to load and it'll predict the sentiment along with the confidence. For the old SA session model, put your review in the my_review.txt file, then run the code in load_sa_session_model.py.

What insights do you generate from this? How do the models compare?

Note – **you should use the bert_mc_cpu environment for all the files in this** task. You do **not** need a GPU or fast CPU for this exercise; you're loading and using models that have already been trained. Note – if you get an error about the resource being exhausted at any point, just restart VSCode.
Also – see next slide...

Test Set Performance Across 3 models

Old SA Session Model :

```
EVALUATION ON TEST DATASET
Loss : 0.31
Accuracy : 0.86
1/1 [==============================] - 0s 53ms/step
REVIEW:
A fantastic movie that really pushed the boundaries of storytelling.

Predicted sentiment is POSITIVE ([0.66298175])
```

BERT Model (5 Epochs) :

```
Loss: 0.6114457845687866
Accuracy: 0.8866400122642517
```

```
A fantastic movie that really pushed the boundaries of storytelling.
0.999111
POSITIVE
```

BERT Model (2 Epochs) :

```
Loss: 0.30150705575942993
Accuracy: 0.8870000243186951
```

```
A fantastic movie that really pushed the boundaries of storytelling.
0.997163
POSITIVE
```