Introduction to BERT

BERT

Bi-directional **E**ncoder **R**epresentation from **T**ransformers

- **Bi-directional**: Auto-encoding Language Model
- **Encoder**: Uses only the Encoder from the Transformer
- **Representation**: Relying on self-attention
- **Transformers**: The encoder is taken from the Transformer architecture

BERT at a distance

Consider the following sentence:

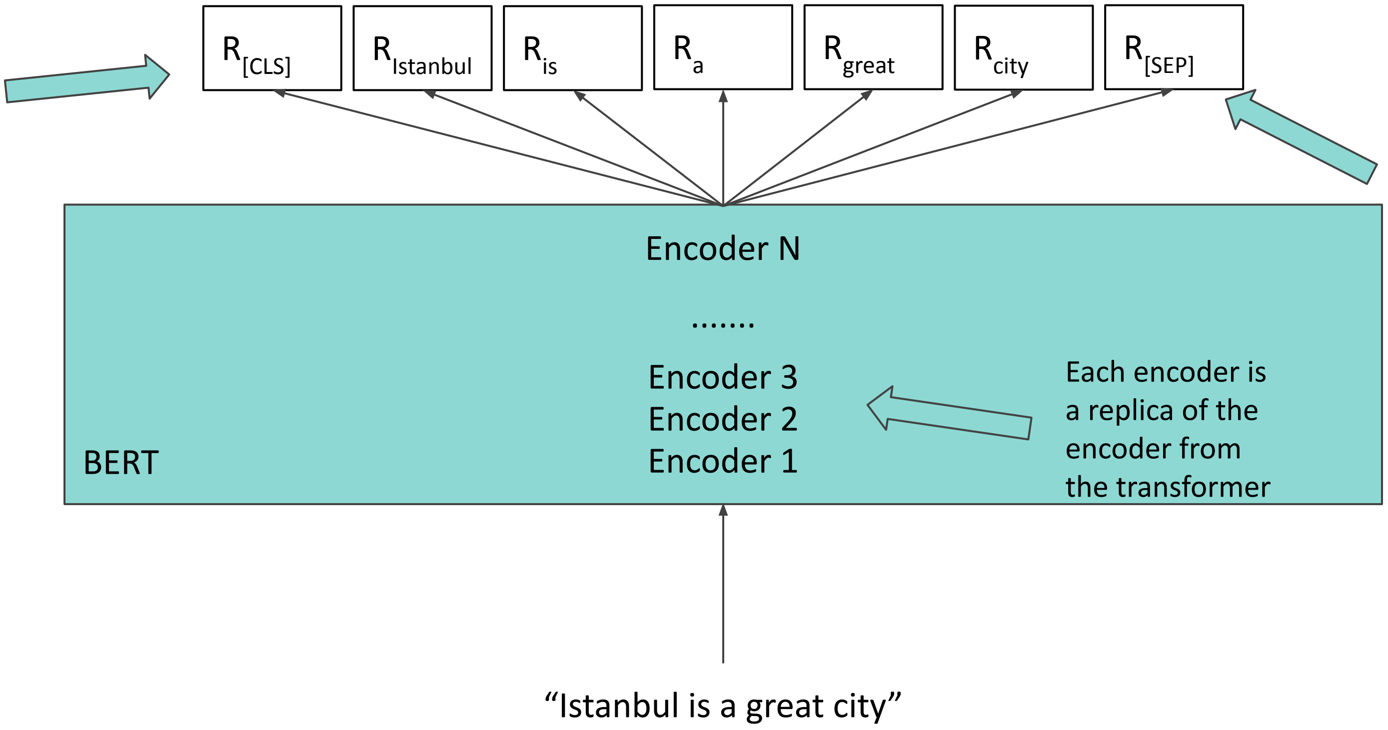
`'I love my pet Python'.`

We feed this sentence into BERT to get a **context-ful** representation (vector embedding) of **every** word in the sentence.

The encoder understands the context of each word in the sentence using a multi-head attention mechanism (which relates each word to every other word in the sentence)

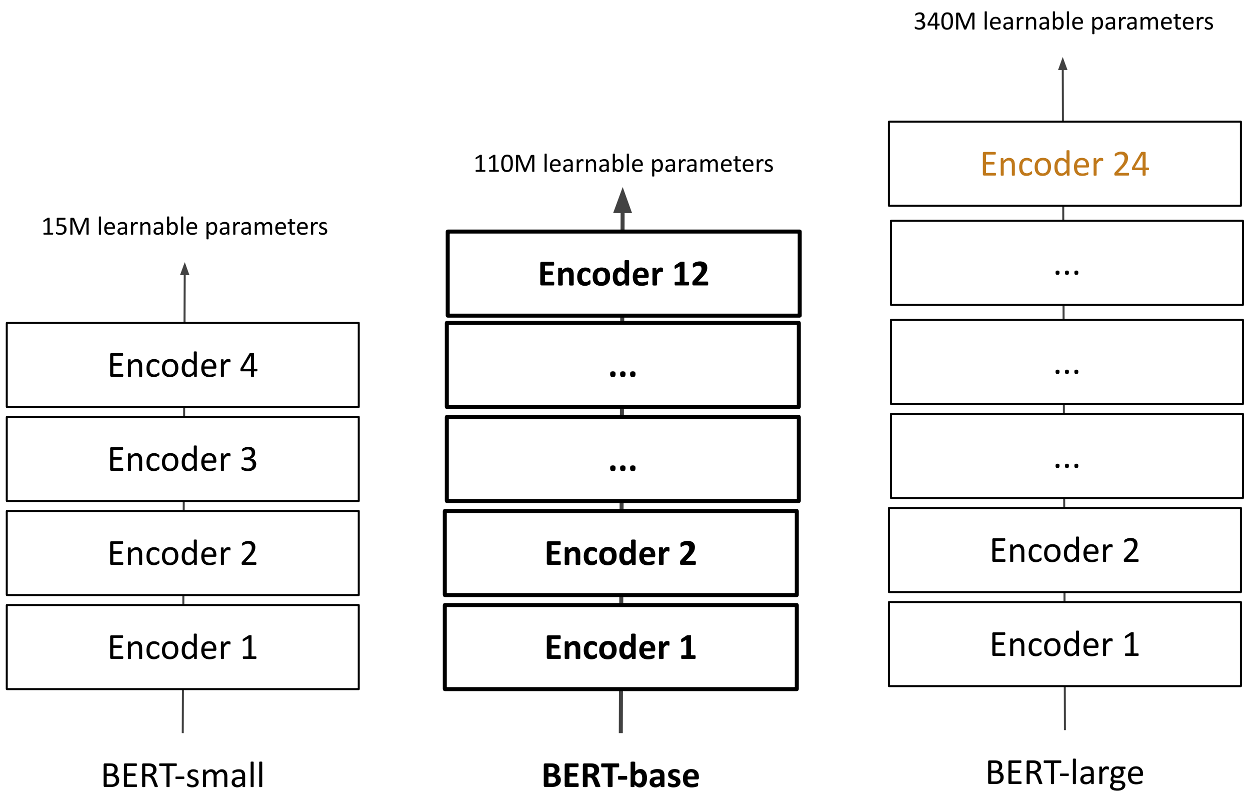
BERT at a distance

This vector is used to represent the entire input. We will see it again later on



Each R_{token} is a vector that represents a token in the input

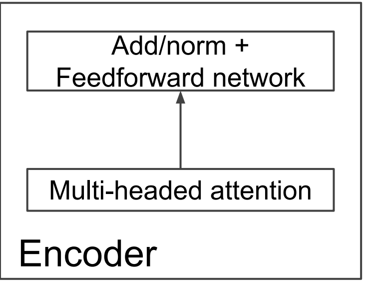
BERT comes in many sizes

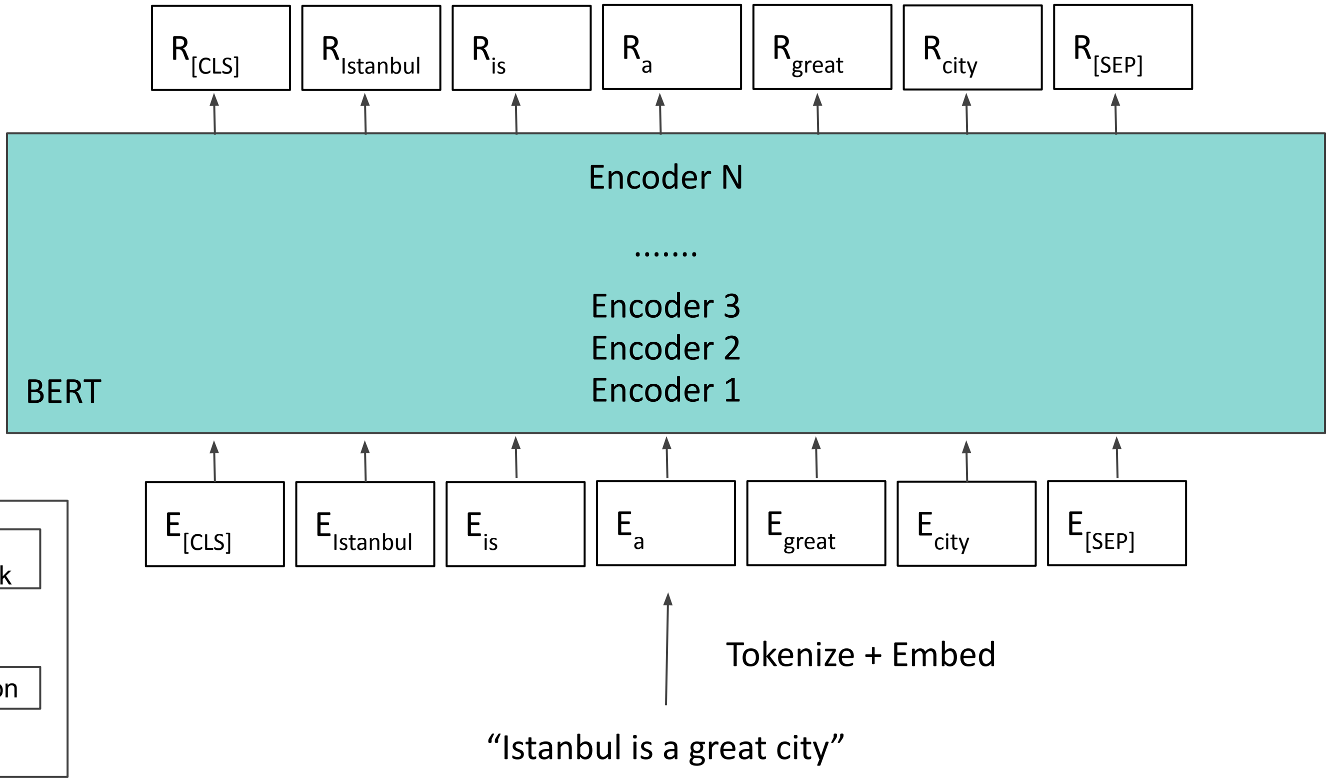


And many more!!

BERT at a slightly closer distance

Recall that each encoder is a combination of multi-headed attention and feedforward / add / norm layers





Representations of words with context

Consider the following sentences:

`'I love my pet Python'`

vs

`'I love coding in Python'`

Quick check. After tokenizing, what index will the word “python” be at? Assume all words are in the vocab

Representations of words with context

Consider the following sentences:

`'I love my pet Python'`

vs

`'I love coding in Python'`

Quick check. After tokenizing, what index will the word "python" be at? Assume all words are in the vocab

5 (zero-index) Don't forget [CLS] would be 0!

['[CLS]',	'i',	'love',	'my',	'pet',	'**python**',	'[SEP]']
0	1	2	3	4	**5**	6

Representations of words with context

Consider the following sentences:

`'I love my pet Python'`

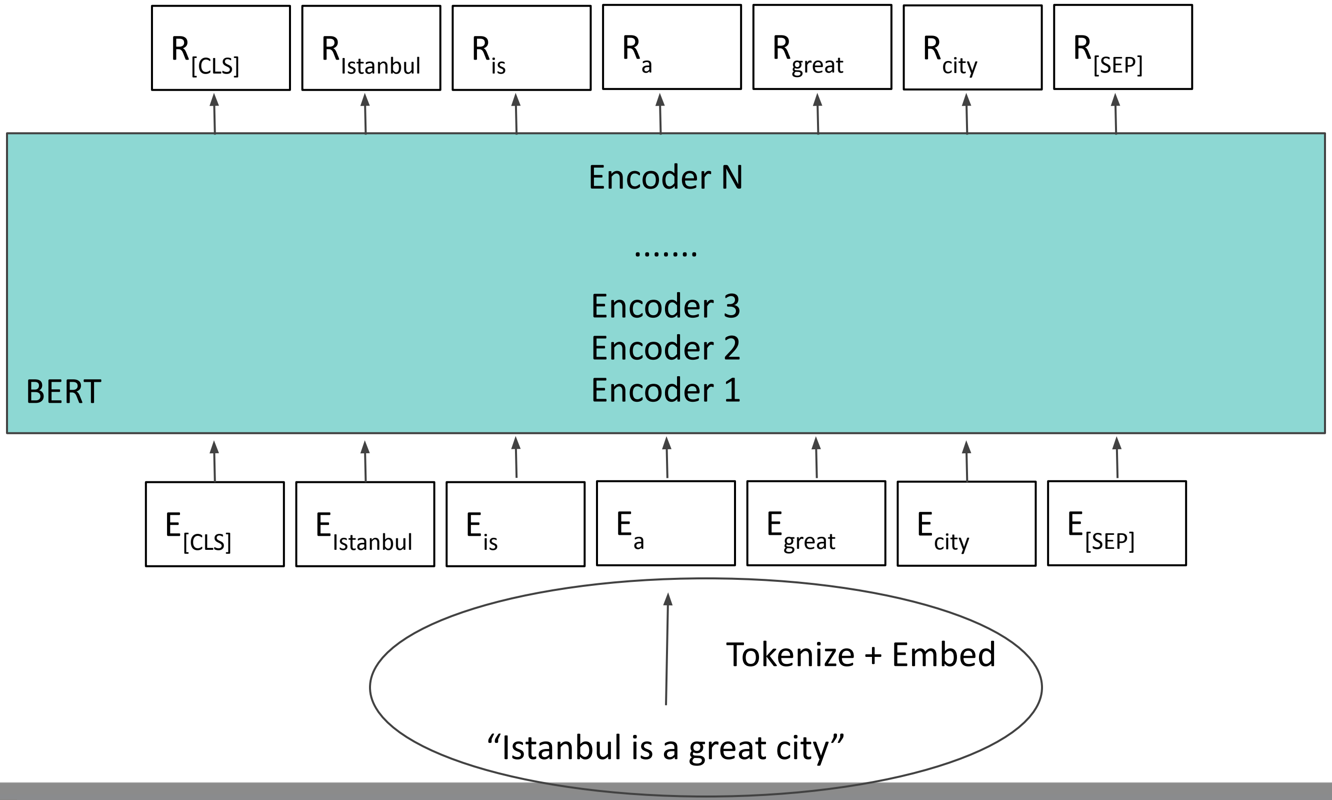
vs

`'I love coding in Python'`

The token “python” will end up with a vector representation from each sentence via BERT. What’s interesting is that the vector representation “python” will be different for each sentence because of the surrounding words in the sentence.

Wordpiece tokenization

Tokenize + Embedding Layer



BERT’s Wordpiece Tokenization

Consider the following sentence:

“Another beautiful day”

To tokenize this, we split into a list of tokens in our vocabulary over over 30,000 tokens. We also add two special tokens **[CLS]** at the beginning of the phrase and **[SEP]** at the end. [CLS] is meant to represent the entire sequence and [SEP] is meant to represent separation between sentences.

[“[CLS]”, “another”, “beautiful”, “day”, “[SEP]”]

BERT’s Wordpiece Tokenization

Consider the following sentence:

“Sinan loves a beautiful day”

```
In [121]: 'sinan' in tokenizer.vocab
Out[121]: False
```

Story of my life..

BERT’s Wordpiece Tokenization

Consider the following sentence:

“Sinan loves a beautiful day”

the “##” indicates a subword

[“[CLS]”, **“sin”, “##an”**, “loves, “a”, “beautiful”, “day”, “[SEP]”]

BERT’s tokenizer is great at handling tokens that are OOV (out of vocabulary) by breaking them up into smaller chunks of known tokens

BERT's Wordpiece Tokenization

Note on tokenization:

BERT has a **maximum sequence length of 512 tokens**. This was implements for the sake of efficiency.

Any sequence less than 512 tokens it will be padded to reach 512 and if it is over 512, the model may error out

Tokenization – Uncased vs Cased

We can either use **uncased** or **cased** BERT tokenization.

Uncased

- Removes accents
- lower-cases the input

Café Dupont --> cafe dupont

Cased

- Does nothing to the input

Café Dupont --> Café Dupont

Tokenization – Uncased vs Cased

- Uncased tokenization is usually better for most situations because case generally doesn't contribute to context

- Cased tokenization works well in cases where case does matter (like Named Entity Recognition)

- Note that this has little to do with the BERT architecture, just in tokenization

Transfer Learning with BERT

Transfer Learning – Fine-tuning

There are generally three approaches to fine-tuning:

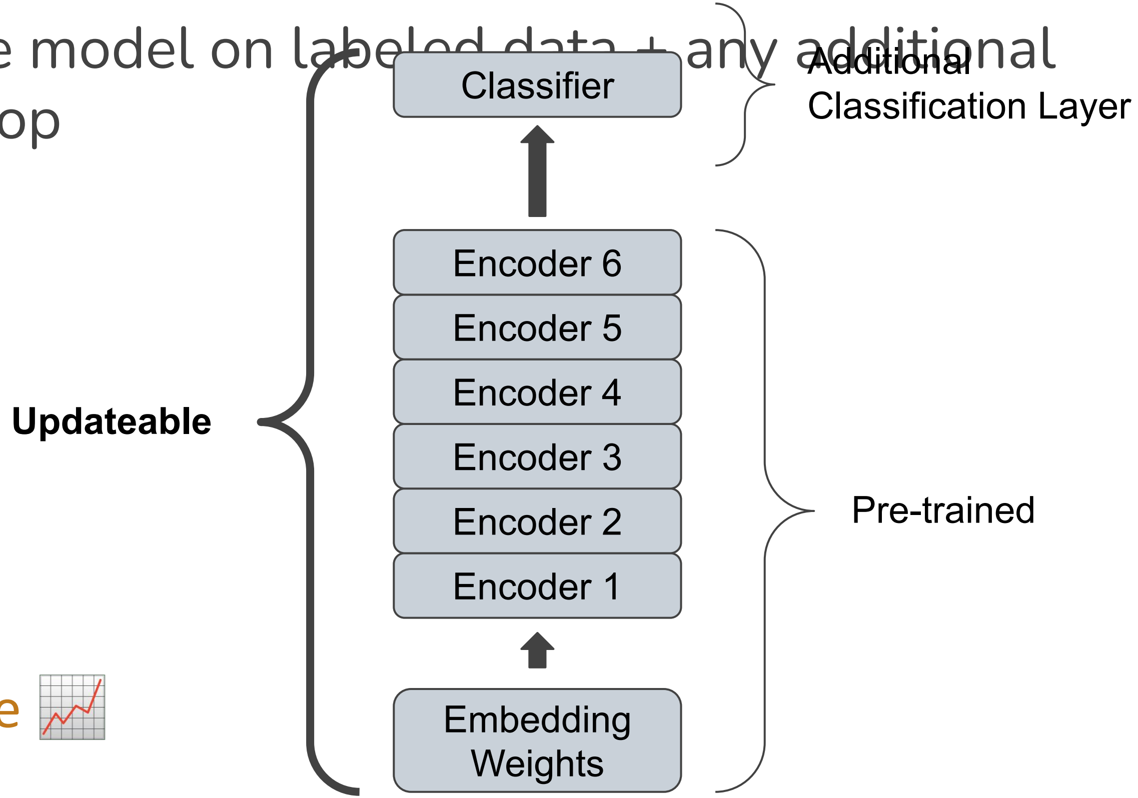
1. Update the whole model on labeled data + any additional layers added on top

2. Freeze a subset of the model

3. Freeze the whole model and only train the additional layers added on top

Fine-tuning strategies

Update the whole model on labeled data + any additional layers added on top

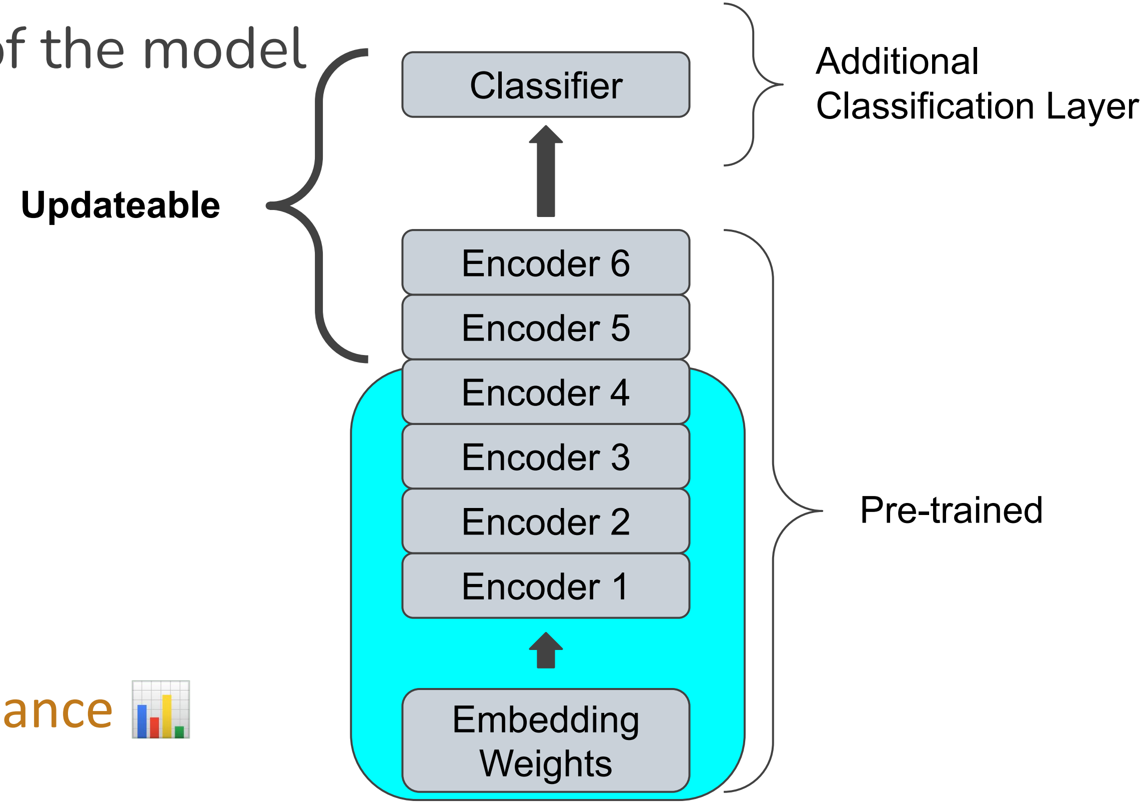


Slowest 🐢

(Usually) best performance 📈

Fine-tuning strategies

Freeze a subset of the model

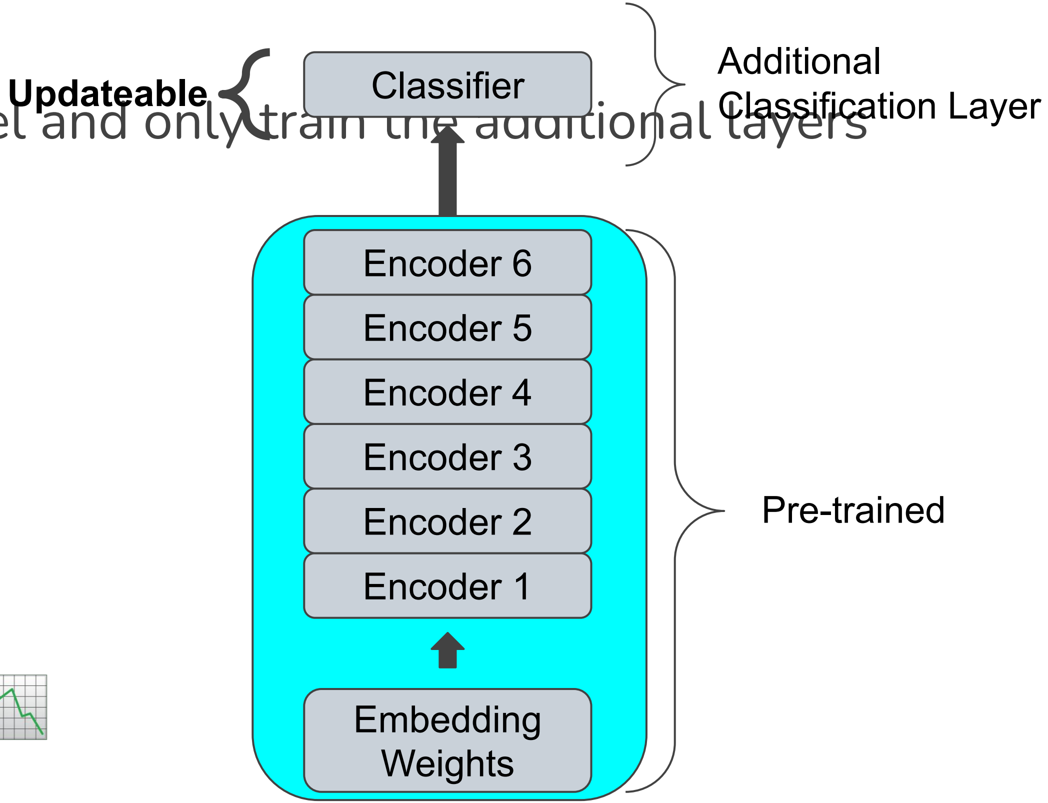


Average Speed ⏰

(Usually) average performance 📊

Fine-tuning strategies

Freeze the whole model and only train the additional layers added on top



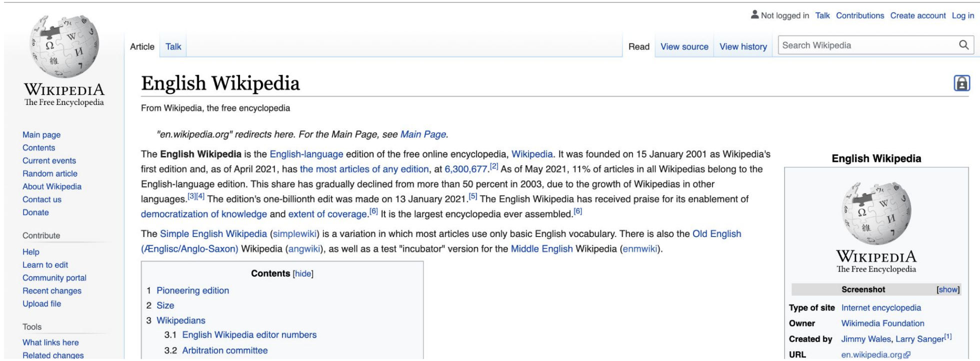
Fastest 🐇

(Usually) worst performance 📉

Pre-training BERT – Corpus

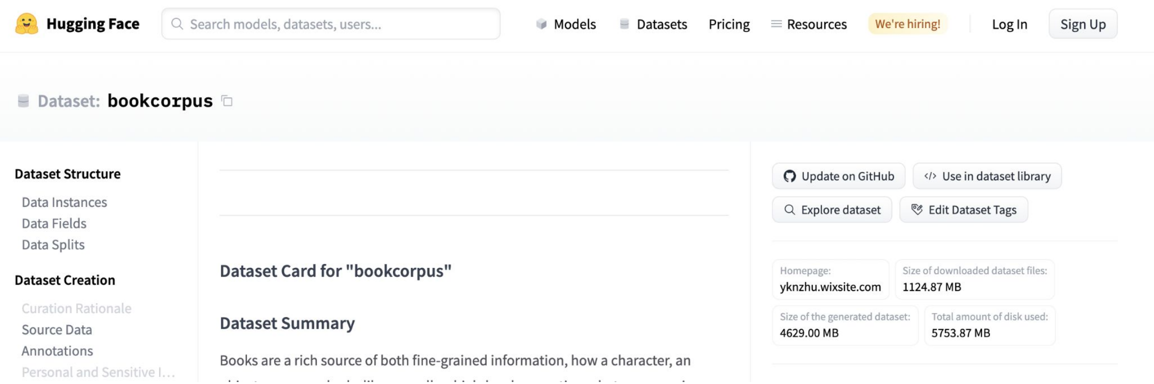
English Wikipedia (2.5B words)

https://en.wikipedia.org/wiki/English_Wikipedia



BookCorpus (800M words)

huggingface.co/datasets/bookcorpus



Pre-training BERT

Pre-training is where BERT really starts to stand out. BERT is pre-trained on two tasks:

1. The Masked Language Model
2. Next Sentence Prediction

These tasks are not generally “useful” tasks but they help BERT learn how words / language work in general

The Masked language modeling task

Pre-training BERT

Masked Language Modelling (MLM)

- Replace 15% of words in corpus with special [MASK] token and ask BERT to fill in the blank

- Think back to our “__ at the light” example. This is the MLM task

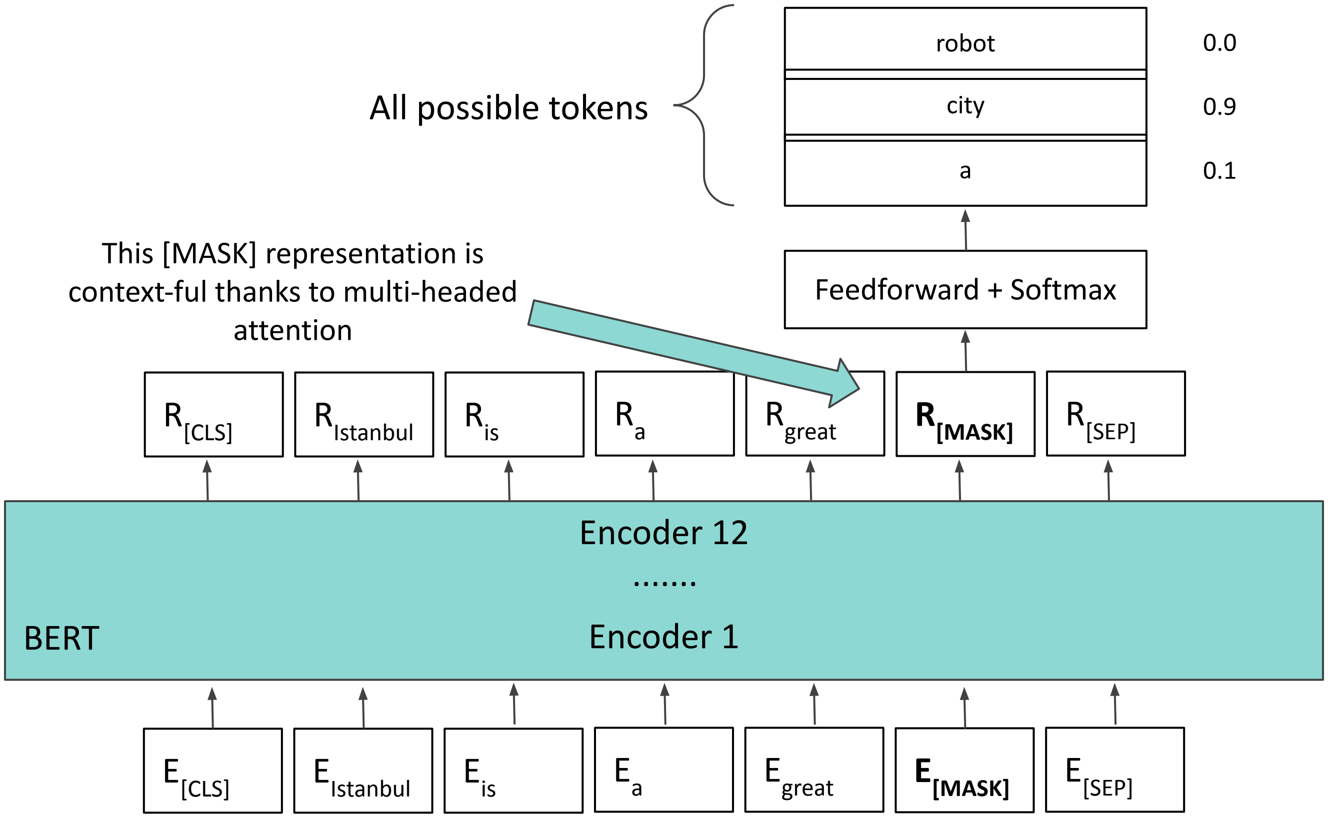
Pre-training BERT

Masked Language Modelling (MLM)

"Istanbul is a great [MASK] to visit"

Guess the word

Pre-training BERT - MLM



The Next Sentence Prediction task

Pre-training BERT

Masked Language Modelling (MLM)

- Replace 15% of words in corpus with special [MASK] token and ask BERT to fill in the blank
- Think back to our “__ at the light” example. This is the MLM task

Next Sentence Prediction (NSP)

- Classification problem
- Given two sentences, did sentence B come **directly** after sentence A?
  - True or False

Pre-training BERT

Masked Language Modelling (MLM)

"Istanbul is a great [MASK] to visit"

Guess the word

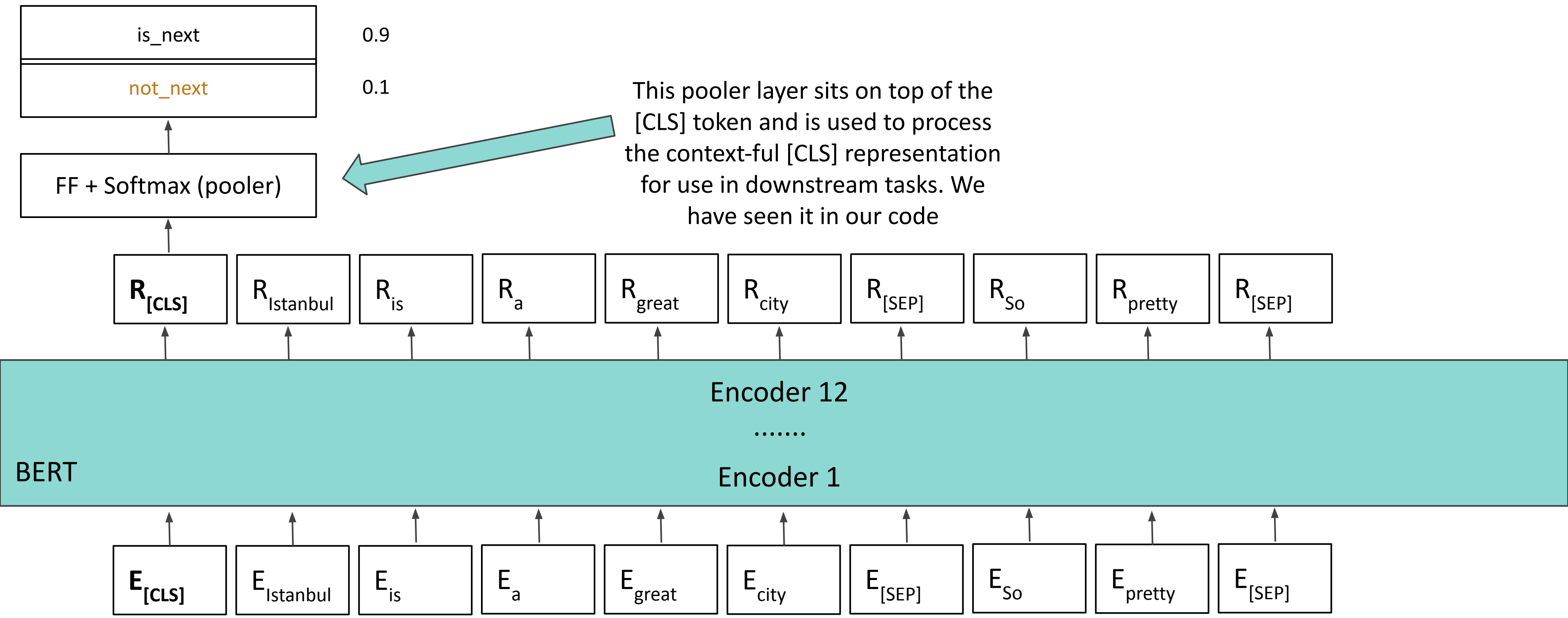
Next Sentence Prediction (NSP)

A: "Istanbul is a great city to visit"

B: "I was just there."

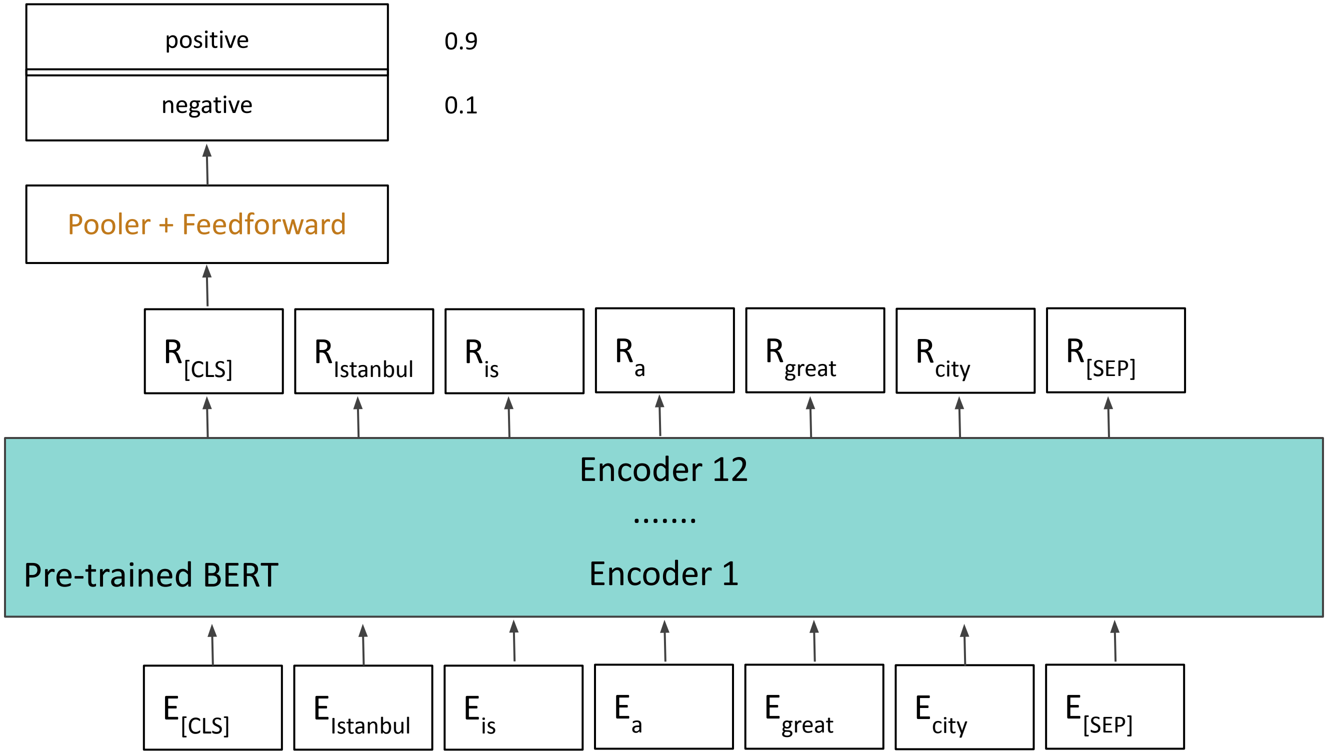
Did sentence B come directly after sentence A? Yes or No

Pre-training BERT - NSP

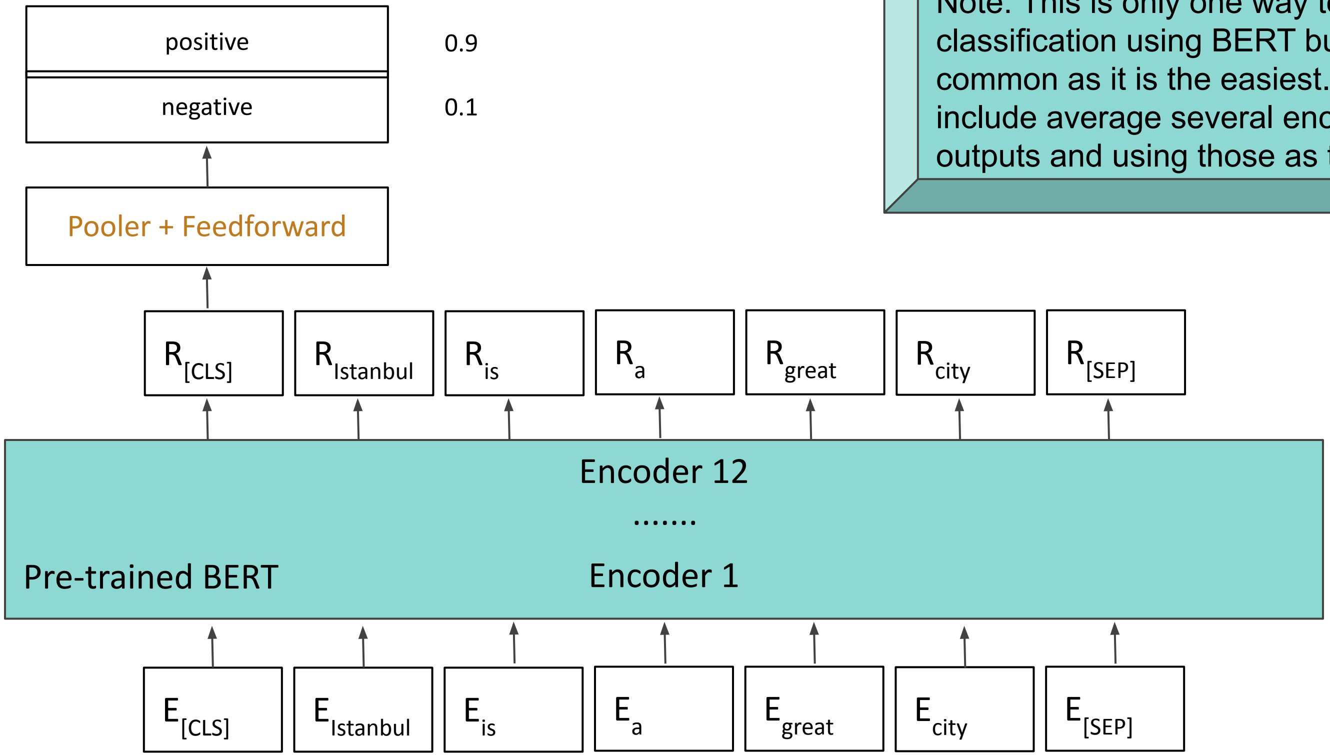


BERT for sequence classification

Fine-tuning BERT – Classification



Fine-tuning BERT – Classification



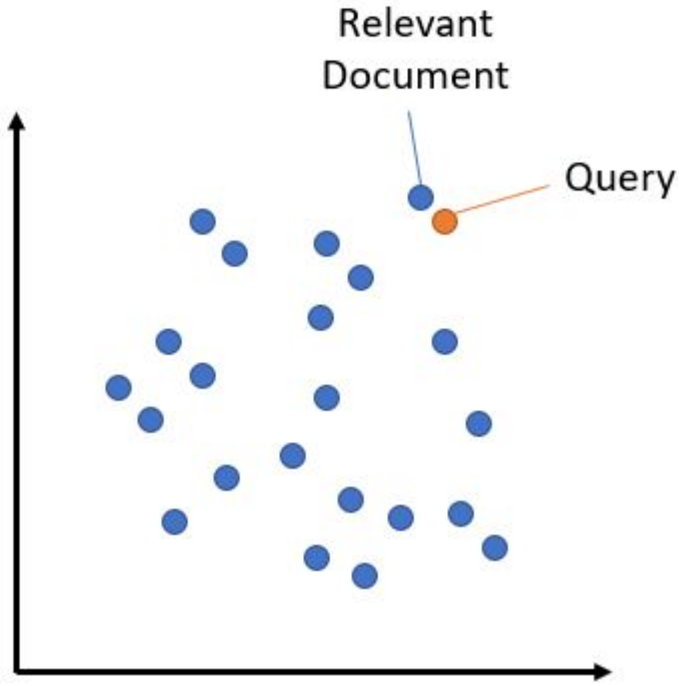
Note: This is only one way to structure classification using BERT but it is the most common as it is the easiest. Other ways include average several encoders hidden outputs and using those as text vectors

Siamese BERT-networks for semantic searching

The Task

Semantic Search

Retrieving relevant documents from a natural language query

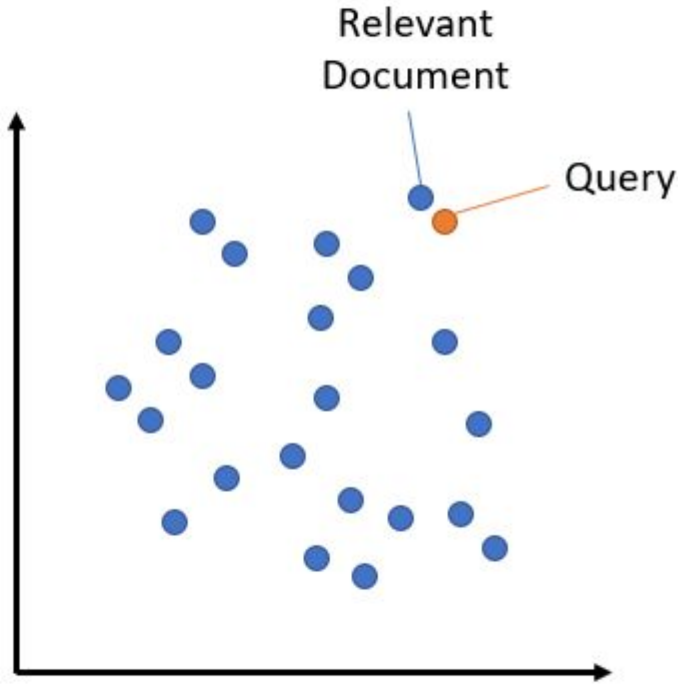


Source: https://www.sbert.net/examples/applications/semantic-search/README.html

The Task

Semantic Search

Unlike traditional search engines, semantic search algorithms use contextual embeddings to perform look-ups, providing for closer context matches than lexical matches



Source: https://www.sbert.net/examples/applications/semantic-search/README.html

Types of Semantic Search

Symmetric Search

- Documents and Queries are roughly the same size and carry the same amount of semantic content

Example:

retrieving news article titles given a query

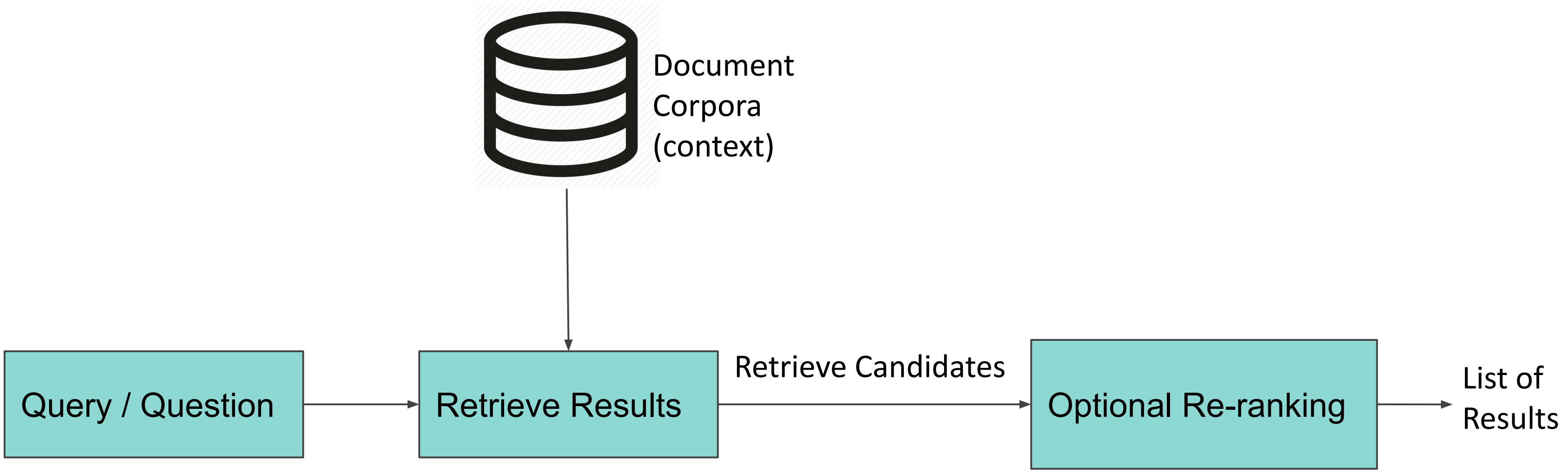
Asymmetric Search

- Documents are usually longer than the queries and carry larger amounts of semantic content

Example:

retrieving an entire paragraph from a textbook to answer a question

Asymmetric Semantic Search



Siamese Architecture / Bi-encoder

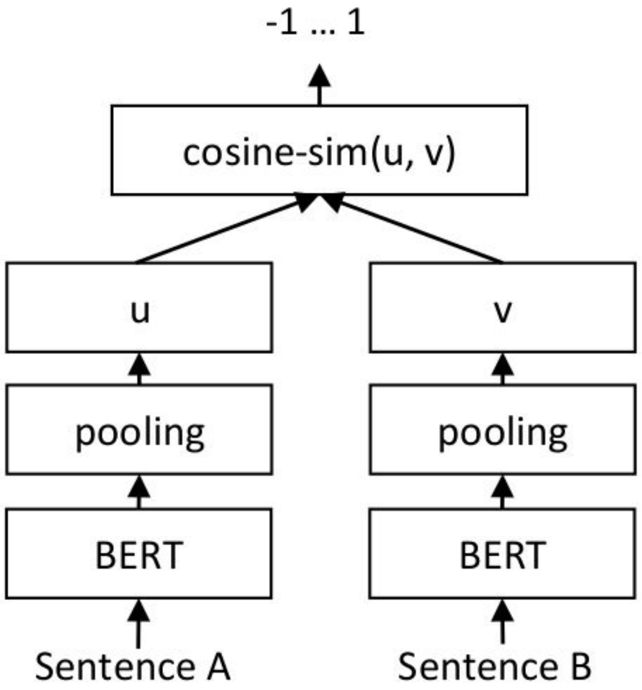
Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

Nils Reimers and Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

Source: https://arxiv.org/abs/1908.10084

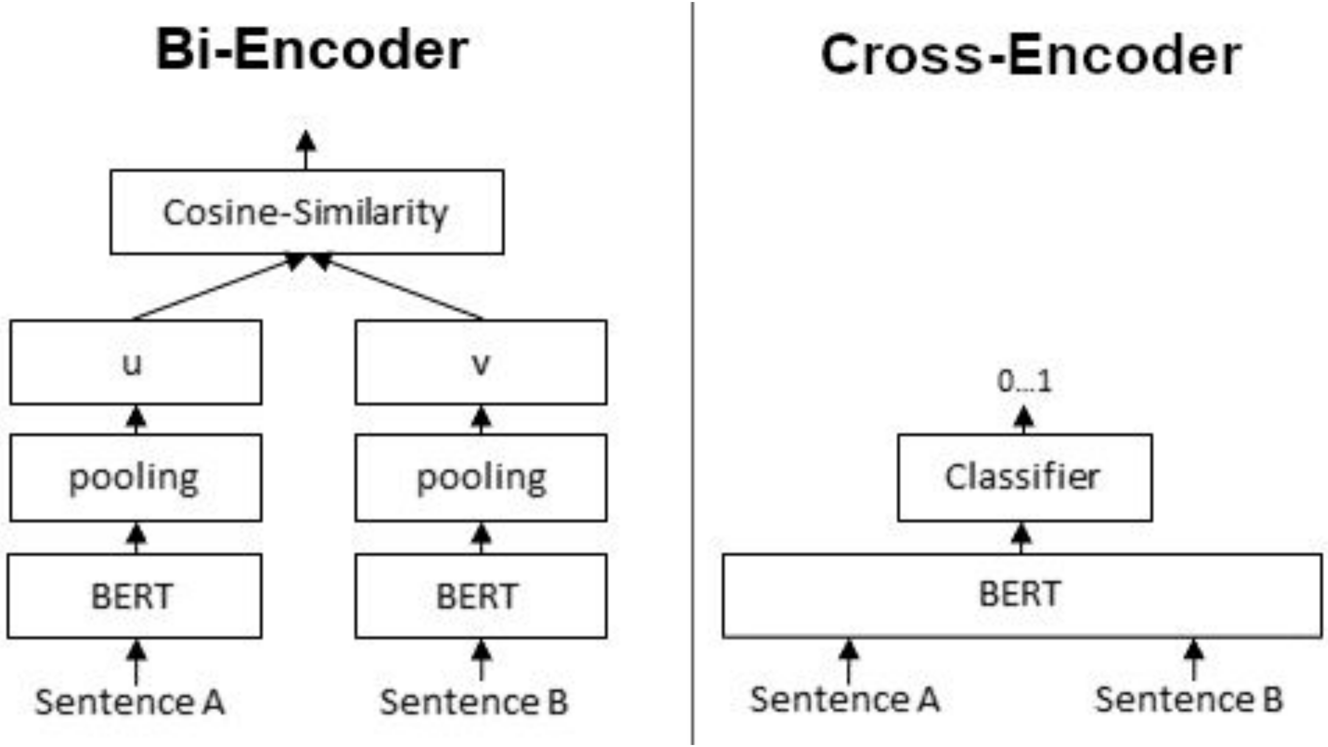
Siamese Architecture / Bi-encoder

A siamese bi-encoder archite n
embeddings that can be com



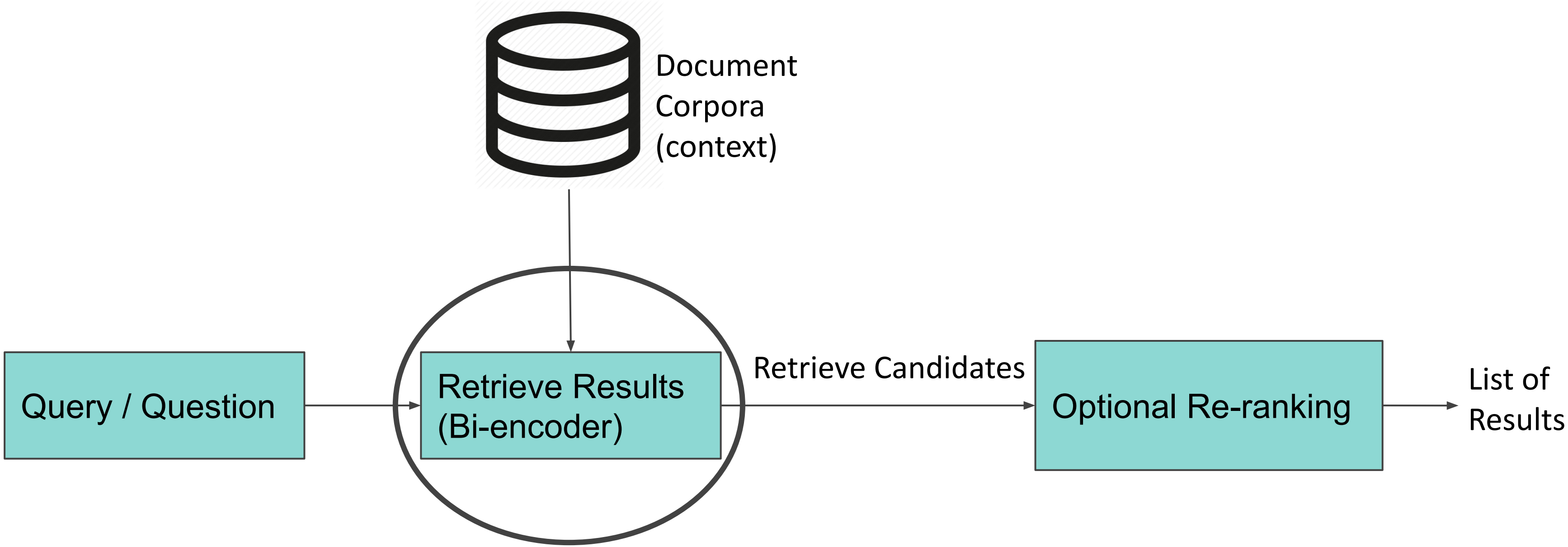
Source: https://arxiv.org/abs/1908.10084

Bi-encoder vs Cross-encoder



Source: https://www.sbert.net/examples/applications/cross-encoder/README.html

Asymmetric Semantic Search



Asymmetric Semantic Search