# BERT

# References


- https://huggingface.co/blog/bert-101
- http://jalammar.github.io/illustrated-bert/
- https://humboldt-wi.github.io/blog/research/information_systems_1920/bert_blog_post/

# BERT

- Bidirectional Encoder Representations from Transformers, is a Machine Learning (ML) model for natural language processing
- It was developed in 2018 by researchers at Google AI Language
- Solution to most of the common language tasks:
  - Sentiment Analysis
  - Question answering
  - Text Prediction: Predicts your text when writing an email (Gmail)
  - Text Generation: Can write an article about any topic with just a few sentence inputs
  - Summarization: Can quickly summarize long documents
  - Polysemy resolution: Can differentiate words that have multiple meanings (like ‘bank’) based on the surrounding text

# Large training data

- A massive dataset of 3.3 Billion words has contributed to BERT's continued success.
  - BERT was trained on Wikipedia (~2.5B words) and
  - Google's BooksCorpus (~800M words)
- Training on a dataset this large takes a long time
  - BERT's training was made possible thanks to the novel Transformer architecture
  - Used 64 TPUs over the course of 4 days.
- Demand for smaller BERT models is increasing
  - to use BERT within smaller computational environments (like cell phones and personal computers)
  - 23 smaller BERT models were released in March 2020
  - DistilBERT offers a lighter version of BERT - runs 60% faster while maintaining over 95% of BERT's performance.

https://huggingface.co/blog/bert-101

# Transformers

- As we have seen, Transformers work by leveraging “Attention”
- Similar to how we humans process information through attention
  - We are incredibly good at forgetting/ignoring mundane daily inputs that are not of much importance
- Transformers create differential weights signaling which words in a sentence are the most critical to further process
- A transformer does this by successively processing an input through a stack of transformer layers called the **encoder**.
- If necessary, another stack of transformer layers - the **decoder** - can be used to predict a target output
- BERT however, doesn’t use a decoder

https://huggingface.co/blog/bert-101

# Unidirectional vs Bidirectional language models

- Transformers, GPT, ELMo - use unidirectional language models to learn general language representations
- These techniques restrict the power of the pre-trained representations
  - For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention of the Transformer
- BERT alleviates the previously mentioned unidirectionality constraint by using a “masked language model” (MLM) pre-training objective, by the Cloze task (Taylor, 1953)
- The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context
- In addition to the masked language model, BERT also uses a “next sentence prediction” task that jointly pretrains text-pair representations.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - Jacob Devlin et all

# Masked Language Model

- MLM enables/enforces bidirectional learning from text by masking (hiding) a word in a sentence
  - and forcing BERT to bidirectionally use the words on either side of the covered word to predict the masked word
- A random 15% of tokenized words are hidden during training
- BERT's job is to correctly predict the hidden words
- Thus, directly teaching the model about the English language

https://huggingface.co/blog/bert-101

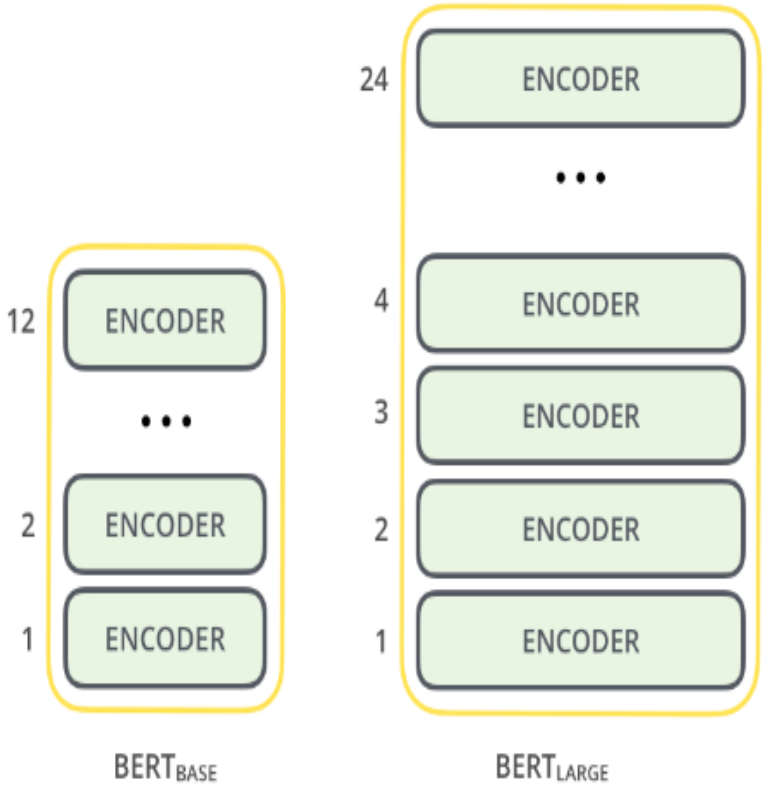
# Next Sentence Prediction

- NSP (Next Sentence Prediction) is used to help BERT learn about relationships between sentences by predicting if a given sentence follows the previous sentence or not.
  - Paul went shopping. He bought a new shirt. (correct sentence pair)
  - Ramona made coffee. Vanilla ice cream cones for sale. (incorrect sentence pair)
- In training, 50% correct sentence pairs are mixed in with 50% random sentence pairs to help BERT increase next sentence prediction accuracy.
- BERT is trained on both MLM (50%) and NSP (50%) at the same time.

https://huggingface.co/blog/bert-101

# BERT

- The paper presents two model sizes for BERT:
  - **BERT BASE** – stack of 12 encoders
  - **BERT LARGE** – stack of 24 encoders - A ridiculously huge model which achieved the state of the art results reported in the paper
- BERT is basically a trained Transformer Encoder stack.




Both BERT model sizes have a large number of encoder layers (which the paper calls Transformer Blocks)

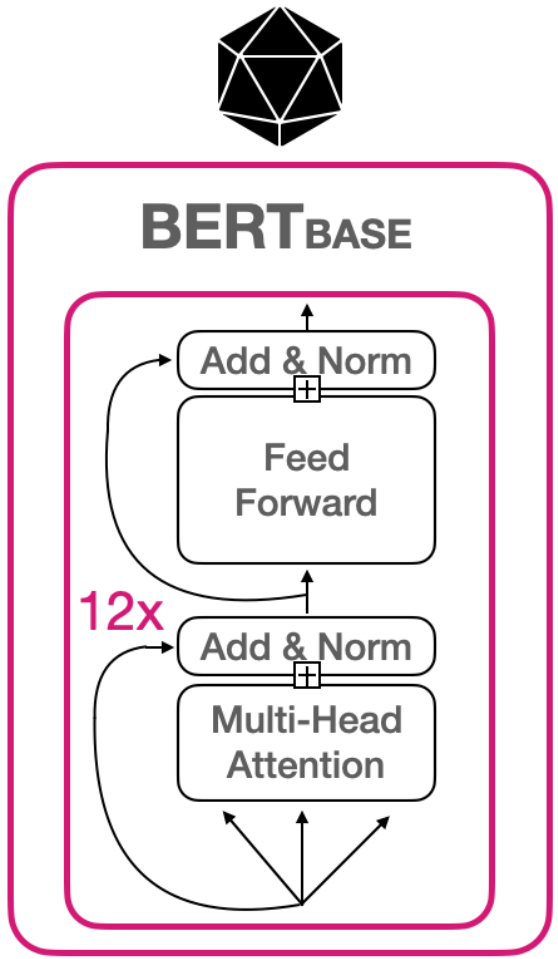
- twelve for the Base version, and twenty four for the Large version
- These also have larger feedforward-networks (768 and 1024 hidden units respectively)
- and more attention heads (12 and 16 respectively)

The default configuration in the reference implementation of the Transformer in the initial paper

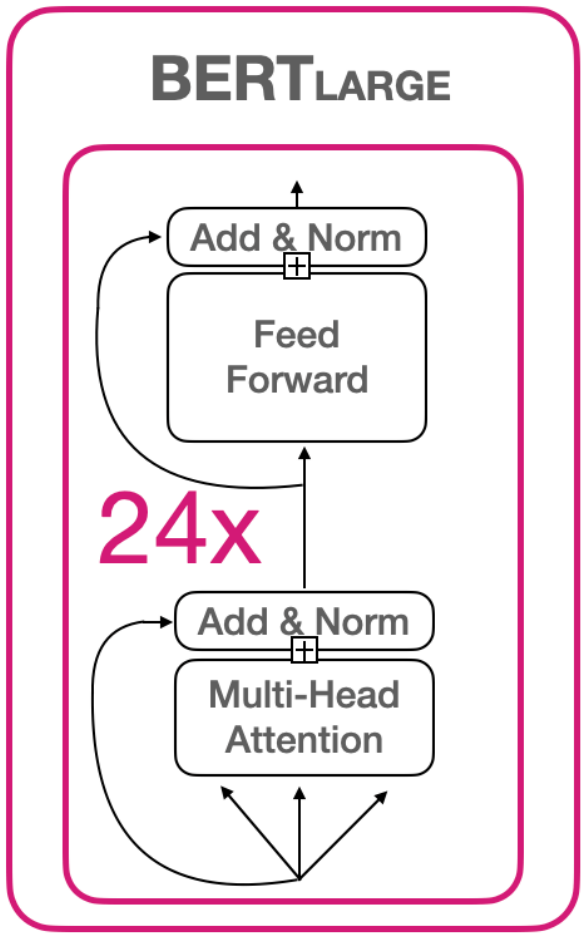
- 6 encoder layers, 512 hidden units, and 8 attention heads

http://jalammar.github.io/illustrated-bert/

# BERT Size & Architecture




110M Parameters

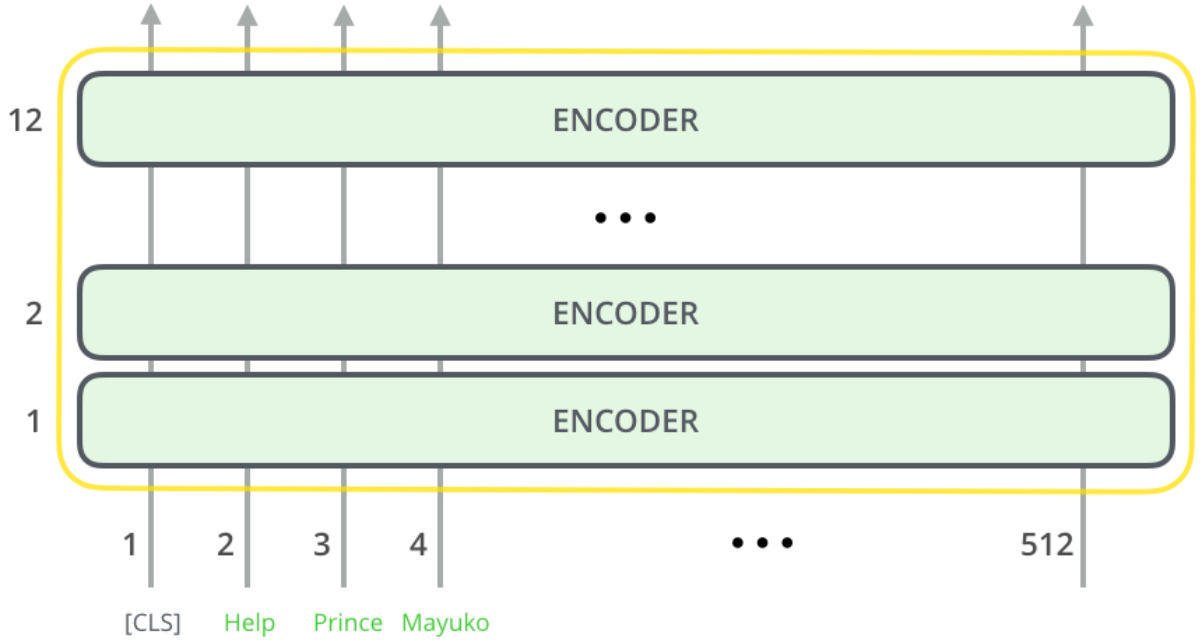



340M Parameters

https://huggingface.co/blog/bert-101

# BERT - Classification

- The first input token is supplied with a special [CLS] token. CLS here stands for Classification.
- Just like the vanilla encoder of the transformer, BERT takes a sequence of words as input which keep flowing up the stack
- Each layer applies self-attention, and passes its results through a feed-forward network, and then hands it off to the next encoder.

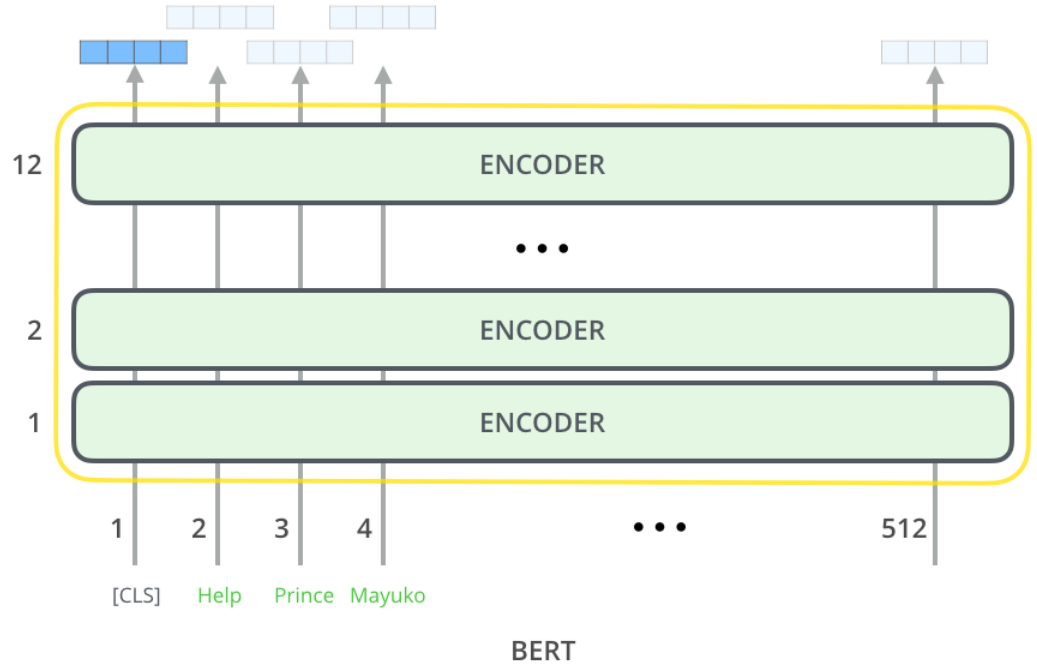



- In terms of architecture, this has been identical to the Transformer up until this point (aside from size, which are just configurations we can set). It is at the output that we first start seeing how things diverge.


http://jalammar.github.io/illustrated-bert/

# BERT - Classification

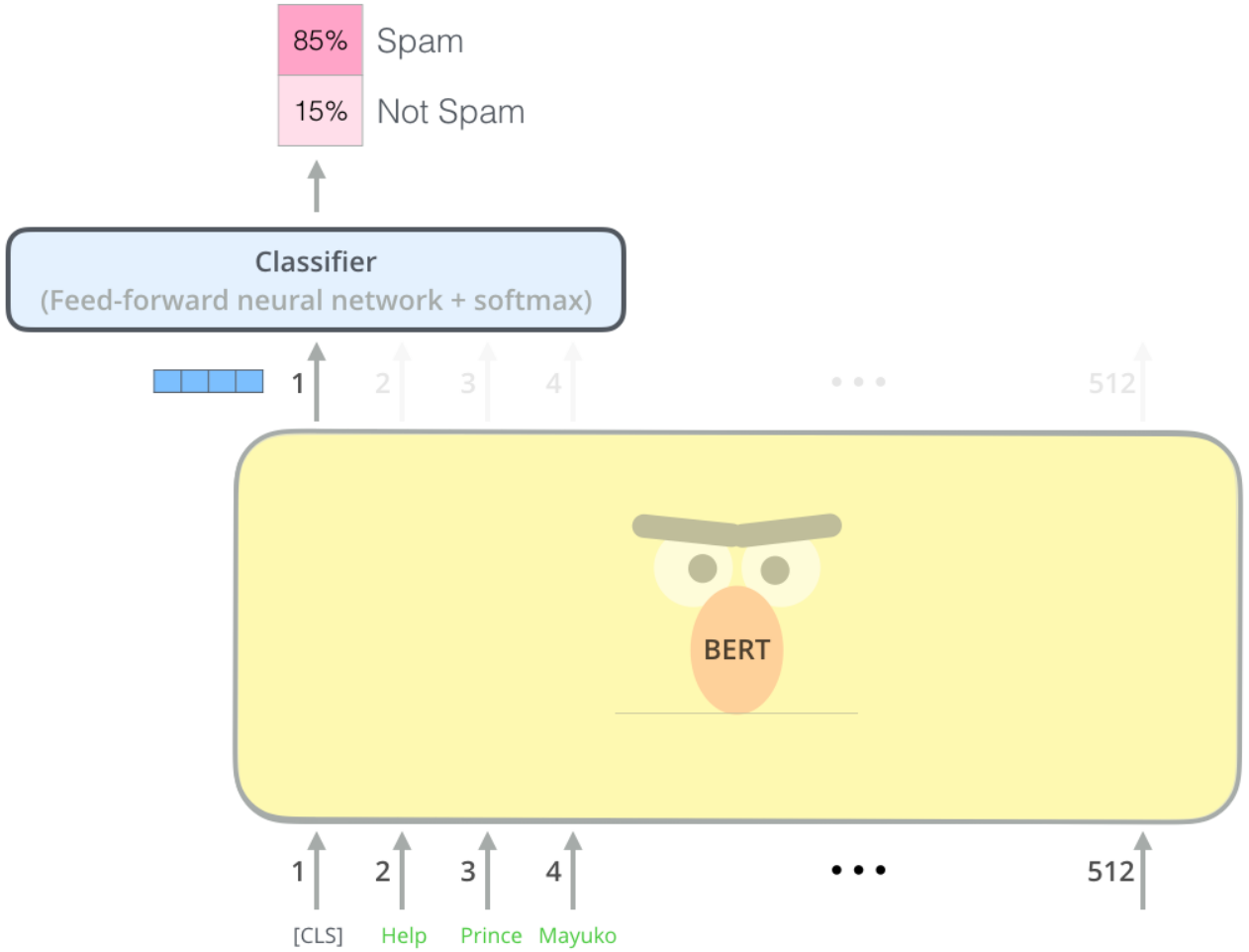
- Each position outputs a vector of size hidden_size (768 in BERT Base).
- For the sentence classification, we focus on the output of only the first position (that we passed the special [CLS] token to).




- That vector can now be used as the input for a classifier of our choosing
- The paper achieves great results by just using a single-layer neural network as the classifier.

http://jalammar.github.io/illustrated-bert/

# BERT - Classification




If you have more labels (for example if you're an email service that tags emails with "spam", "not spam", "social", and "promotion"), you just tweak the classifier network to have more output neurons that then pass through softmax

http://jalammar.github.io/illustrated-bert/

http://jalammar.github.io/illustrated-bert/

## 1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

### Semi-supervised Learning Step

**Model:**

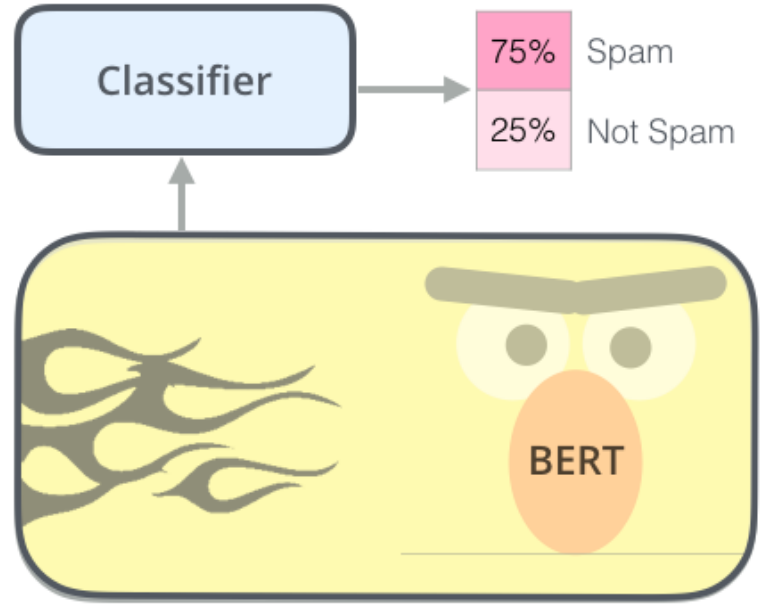

**Dataset:**


**Objective:** Predict the masked word (langauge modeling)

## 2 - Supervised training on a specific task with a labeled dataset.

### Supervised Learning Step

**Model:** (pre-trained in step #1)




**Dataset:**

| Email message | Class |
|---|---|
| Buy these pills | Spam |
| Win cash prizes | Spam |
| Dear Mr. Atreides, please find attached… | Not Spam |