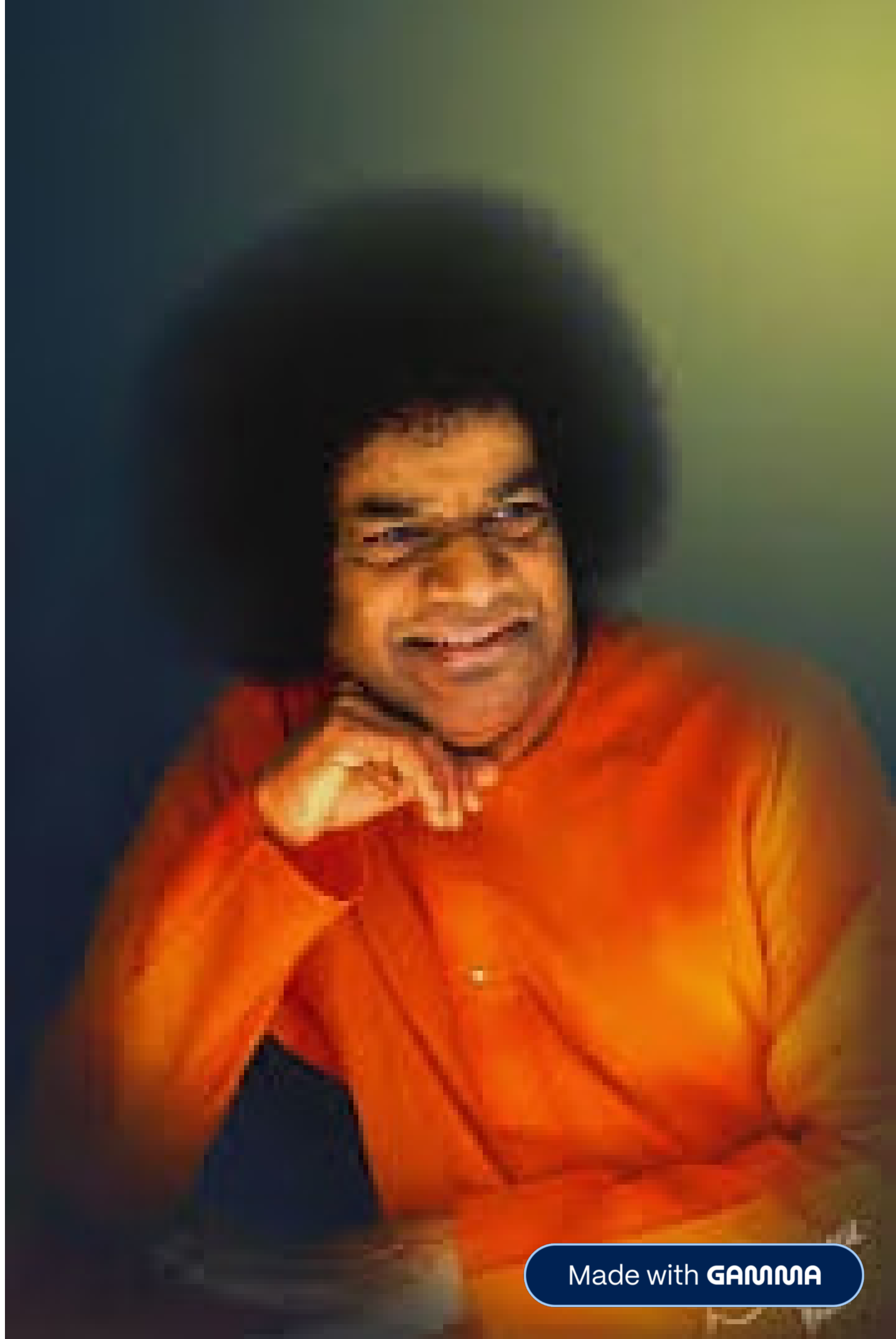
Bidirectional Encoder Representations from Transformers - BERT Base Uncased

Journey till Pretraining

by Vishwanath Shetiya

Offering humble Pranams at Swami's Lotus Feet



Contents

Introduction

- BERT - Bidirectional Encoder Representations from Transformers.
- Introduced by Google in 2018.
- Uses the transformer neural network architecture.
- It is an encoder-based model.
- Understanding the input sequence and context.
- Different types:
    - BERT Base (Cased and Uncased)
    - BERT Large (Cased and Uncased)
    - Others are the optimized or fine-tuned versions of these, e.g., Robustly Optimized BERT, SpanBERT, etc.

Why BERT Base Uncased Model?

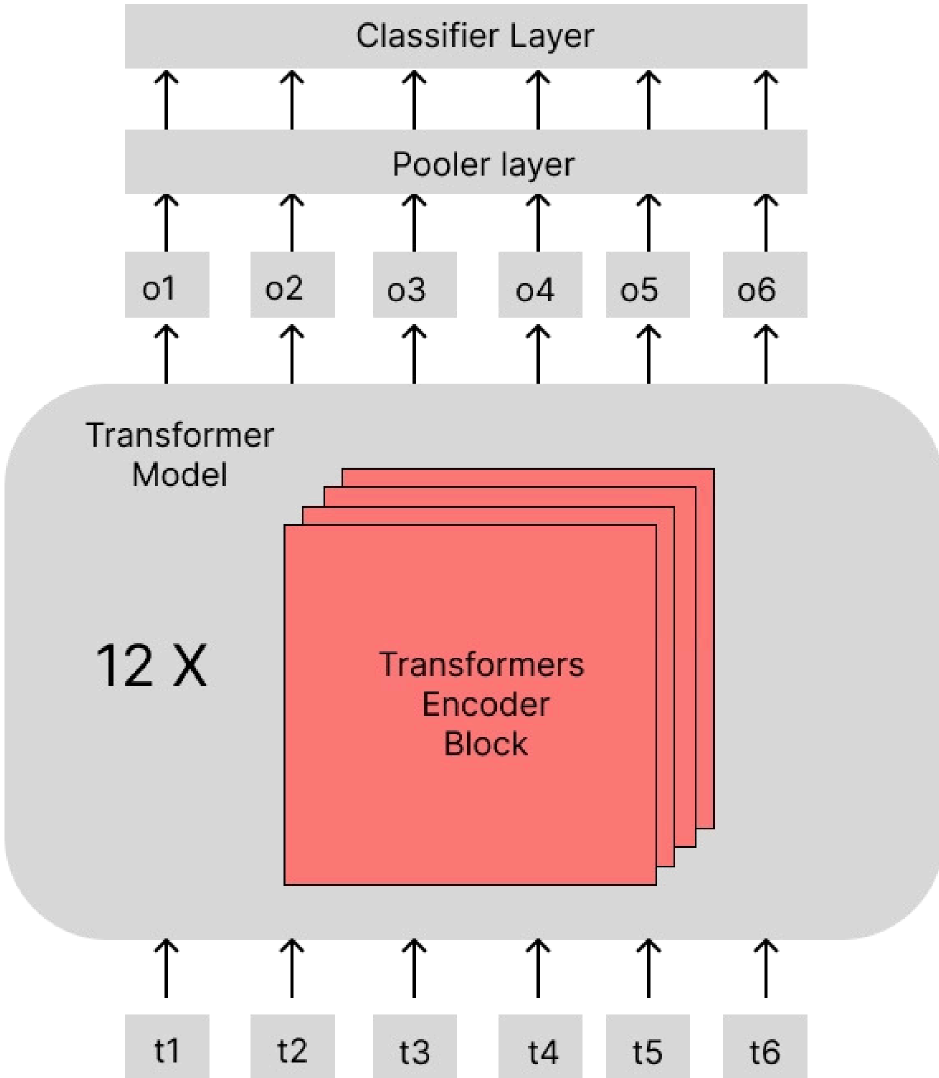
- Faster in training and inference.
- Less computaionally expensive.
- For many NLP tasks, it is a starting point.
- Since it is an uncased version, it helps with tasks that are case-insensitive.
- Transfer learning.

BERT Base Uncased Architecture:

- It is an encoder-based model.
- The Bert Base uncased model is made up of 12 layers/blocks of transformer encoder blocks.
- Each encoder block consists of
  - Multi-head Self-attention
    - Heads - 12
  - Feed Forward network with GELU activation function.

$$\mathrm{GELU}(x) = x * \Phi(x)$$

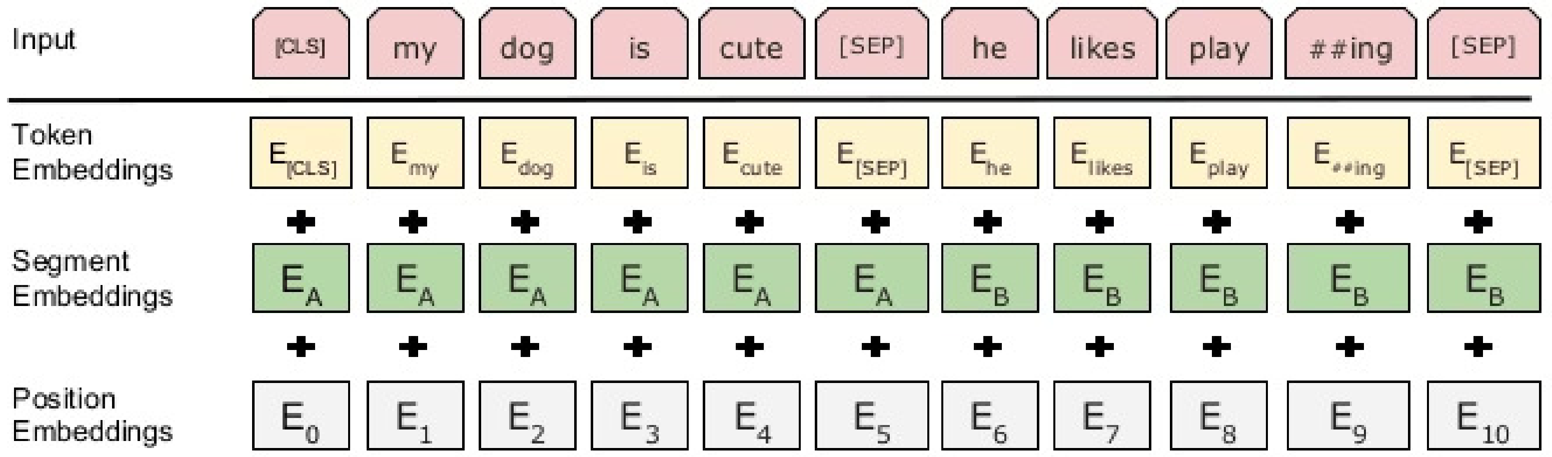
- Hidden states - 768
- The "bidirectional" aspect of BERT refers to how it considers both left and right tokens when processing a token.
- A pooler layer follows the Transformer Model stack.
- Depending on the task or fine-tuning, a classifier layer is introduced.



Input and Output Representation

- **Input:**
- BERT is trained on datasets from Wikipedia and book corpus, and it has a vocabulary list of 30,000 tokens.
- For a given sequence, we have tokens based on WordPiece tokenization.
- These tokens are mapped to an index in BERT's vocabulary, where those indices are assigned to a vector in the embedding space based on the WordPiece tokenization.
- A token called [CLS] is added at the beginning of each input sequence.
  - [CLS] is a classification token, used specifically for classification tasks (used in pretraining).
  - The output of this token will be an aggregate representation of the entire input sequence.
- [SEP] is also used in the input sequence to separate the segments in input sequences. It is used even at the end of the input sequence (i.e., a single segment).
- **Output:**
  - Hidden states are the output from the transformer block, where we get the output of each token.
    - The hidden state of [CLS] holds the contextual information of the input sequence.
  - Pooled output is the vector representation of the [CLS] token that acts as the aggregate representation of the input sequence.

Input and Output Representation



Pooler Layer:

- A fully connected (linear) layer that projects the hidden state of the [CLS] token to a new space.
- After passing through the dense layer, the output is passed through a non-linear activation function.
- Activation function: $\tanh(x)=(e^x+e^-x)/(e^x-e^-x)$.
- This helps in giving a single vector of the same dimension as the input vectors, and it will represent the whole sequence.
- And this pooled output will go through another layer depending on tasks.
- Used only when [CLS] token will be used for downstream tasks.

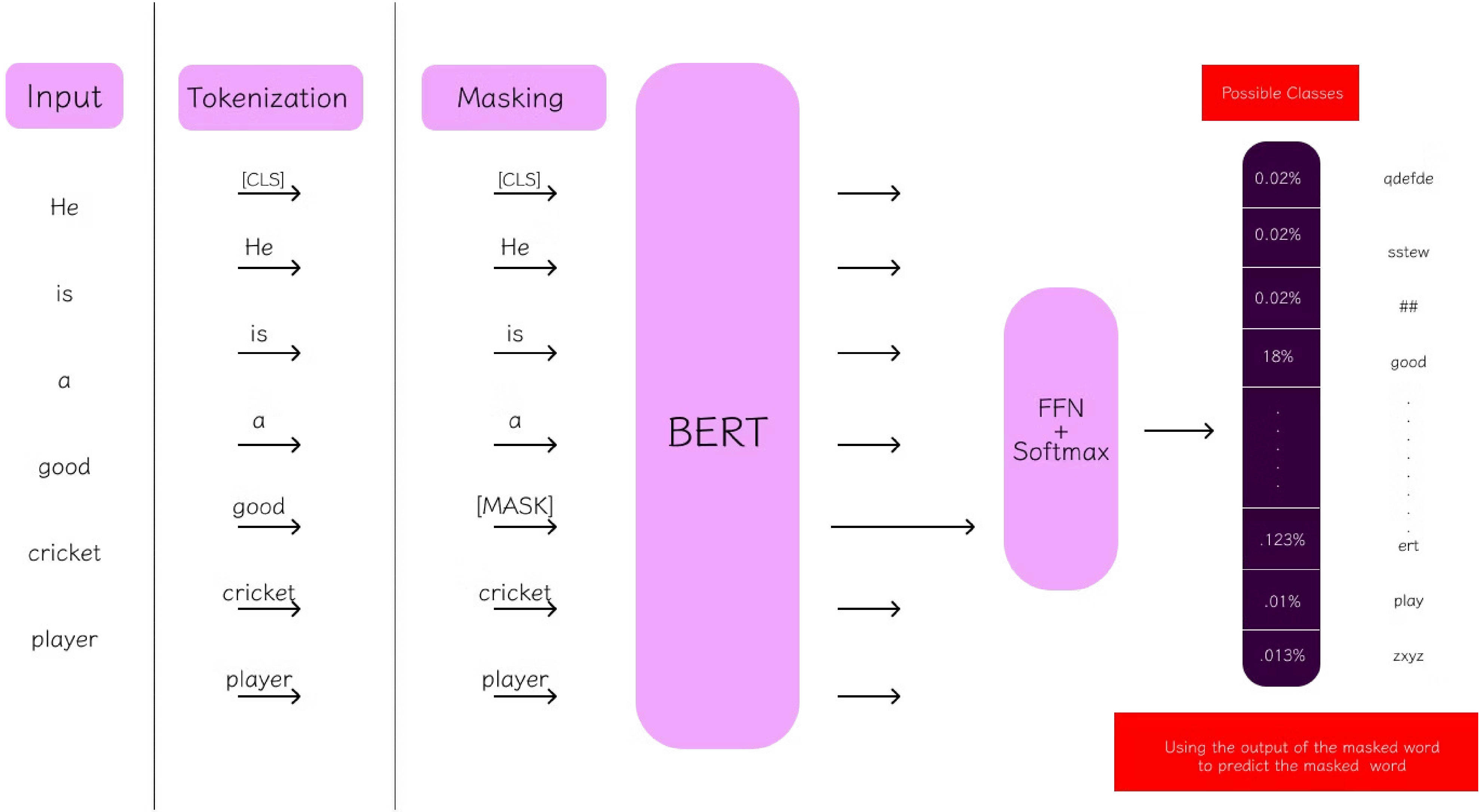
Pretraining phase:

- Self-supervised learning.
- Two ways they are pretrained:
  - Masked language model
  - Next sentence prediction
- helps in general understading of language it is trained on.
- helps in transfer learning.

Masked language model:

- The model learns to predict the missing word or masked word based on the information or the context of surrounding words.
- When given input to the model, 15% of the input is chosen and they are replaced 80% of the time with [MASK], 10% of the time with different tokens, and 10% of the time with the same token.
- [CLS] is added at the beginning and [SEP] is added at the end.
- The model masks according to the condition given above.
- A classification layer is added to the model that projects the output of the model over the vocabulary.
- Each vector goes through a linear layer of the classification layer and gets projected over the vocabulary.
- Masked token outputs go through a softmax layer during MLM pretraining and we get the probability distribution.
- Loss is also calculated using them both using cross-entropy.
- The optimizer used is AdamW.

Masked Language model

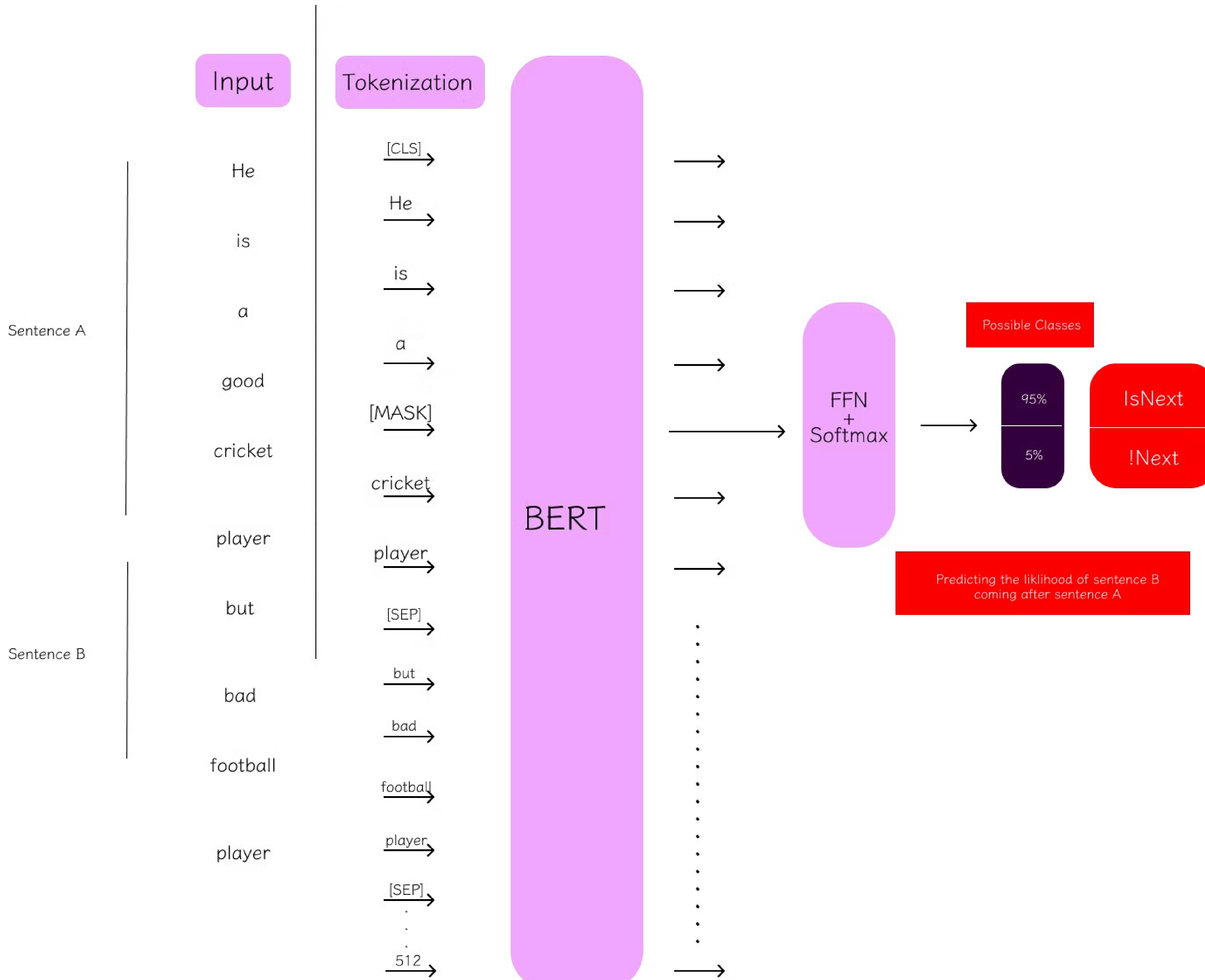


Made with GAMMA

Next sentence prediction

- The model learns to predict whether two sentences appear consecutively or not.
- Input:
  - [CLS] is added at the beginning and [SEP] added in between and at the end.
  - The tokens are created, sentence embeddings are added, positional encoding is done.
  - The sentence that follows sentence A is 50% of the time the next sentence and 50% of the time some random sentence.
- It goes through the model and the output of the [CLS] token is projected to a vector of 1×2, giving us the logits, then passed to softmax to give a probability distribution.
- Hence, comparing with the ground truth, we calculate the loss using binary cross entropy.
- And it is optimized using AdamW.

Next sentence prediction



Made with GAMMA

Putting it all Together

- **MLM** and **NSP** are pretraining tasks that are done together in BERT.
- **MLM** helps the model learn word representations and context, while **NSP** helps the model learn sentence relationships.
- We use combined loss to optimize.
- Helps in downstream tasks as it has the word and sentence level knowledge.

Applications:

- Text Classification
  - spam messages, emotions, etc.
- Question Answering
  - Provide a question and passage; it should mark the start and end of the answer.
- Named Entity Recognition (NER)
  - Help us to differentiate entities in terms of person, location, organization, other, etc.
- Language Translation
  - Helps in understanding the language so that it can be used for translation purposes by other models.

Conclusion

- We discussed the architecture of BERT base uncased.
- How its input and output are represented.
- Its pretraining phases.
- BERT's various applications in which it is used.

References

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understandinghttps://arxiv.org/abs/1810.04805
- Attention Is All You Need - https://arxiv.org/abs/1706.03762
- BERT base model (uncased) - https://huggingface.co/google-bert/bert-base-uncased
- https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
- https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/