UE21CS343BB2
Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Anashua Dastidar, Teaching Assistant

Introduction

BERT is a language representation model pre-trained on a very large amount of unlabeled text corpus over different pre-training tasks. It was proposed in the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., 2018)[1].

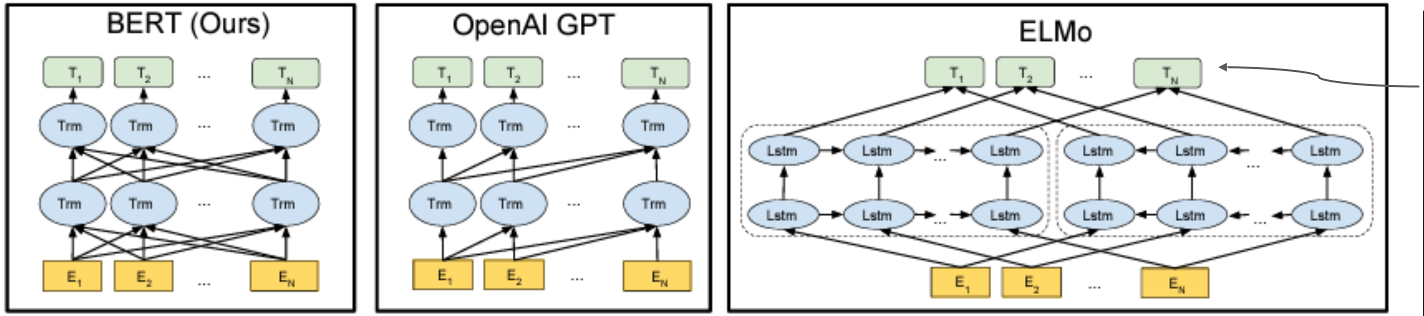
The main idea behind BERT was to create a pre trained large language model which could be used for various downstream tasks which could either be

1. Feature Based tasks (generating word embeddings)

1. Fine Tuning

Why Bidirectional ?

Some of the popular language models before the BERT paper was released were ELMo[2] and Open AI's GPT[3] . However the authors of the BERT paper argued that the current techniques used in creating these models restricted the power of pre trained models . The major limitation being that they were unidirectional. The authors said that such techniques were suboptimal when applied to sentence level tasks and could be harmful when applied to token level tasks such as question answering.

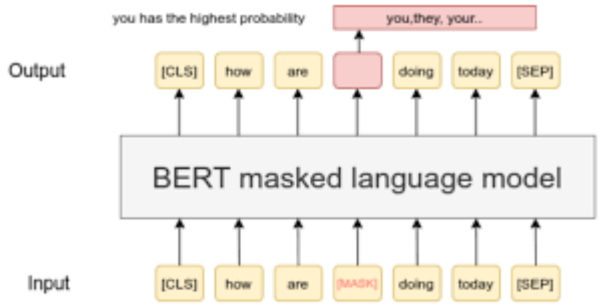


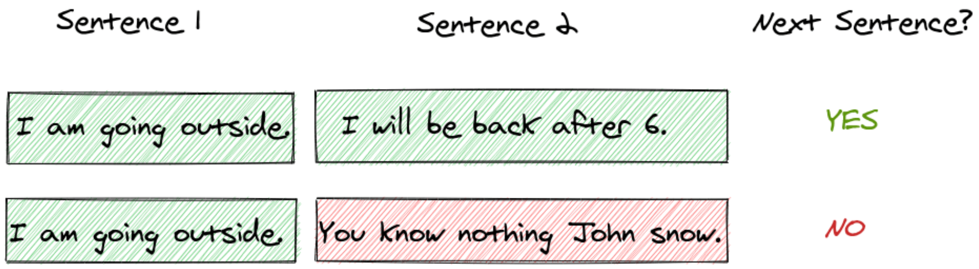
ELMo concatenates the outputs of the left-right model and the right-left model which is a shallow approach.

Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

How does BERT overcome this issue?

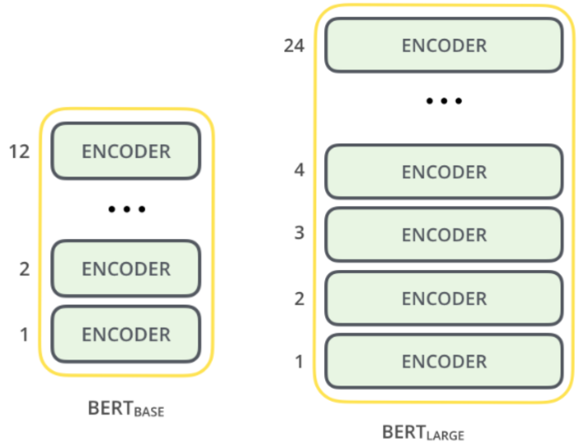
- BERT alleviates the previously mentioned unidirectionality constraint by using a pre training step called masked language modelling. The idea is to randomly mask some tokens from the objective sequence and the model must predict the original token based on only the context .
  - Masked language modelling enables the representation to fuse together the left and right context which allows us to pretrain a deep bidirectional transformer
- In addition to masked language modelling another pre training task is NSP or Next sentence prediction, this is used to jointly pre trains text-pair representations.





BERT Architecture

- There are two steps to the BERT framework the first being pre training with a large corpus and the second being fine tuning to various downstream tasks.
- BERT's model architecture is a multi-layer bidirectional Transformer **encoder** based on the original implementation described in Vaswani et al. (2017)
- The paper presents two models BERT$_{BASE}$ AND BERT $_{LARGE}$, the BERT$_{BASE}$ model is similar in size to GPT and is used to make comparisons while the BERT $_{LARGE}$ model is used to achieve the state of the art results presented in the paper.

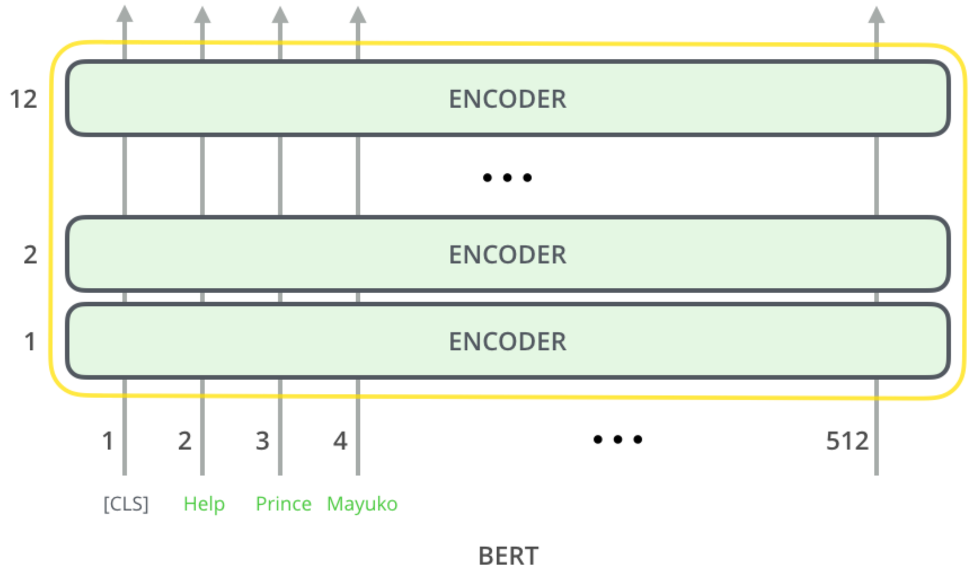


- Both BERT model sizes have a large number of encoder layers (which the paper calls Transformer Blocks) – twelve for the Base version, and twenty four for the Large version.
- These also have larger feedforward-networks (768 and 1024 hidden units respectively), and more attention heads (12 and 16 respectively) than the default configuration in the reference implementation of the Transformer in the initial paper (6 encoder layers, 512 hidden units, and 8 attention heads).

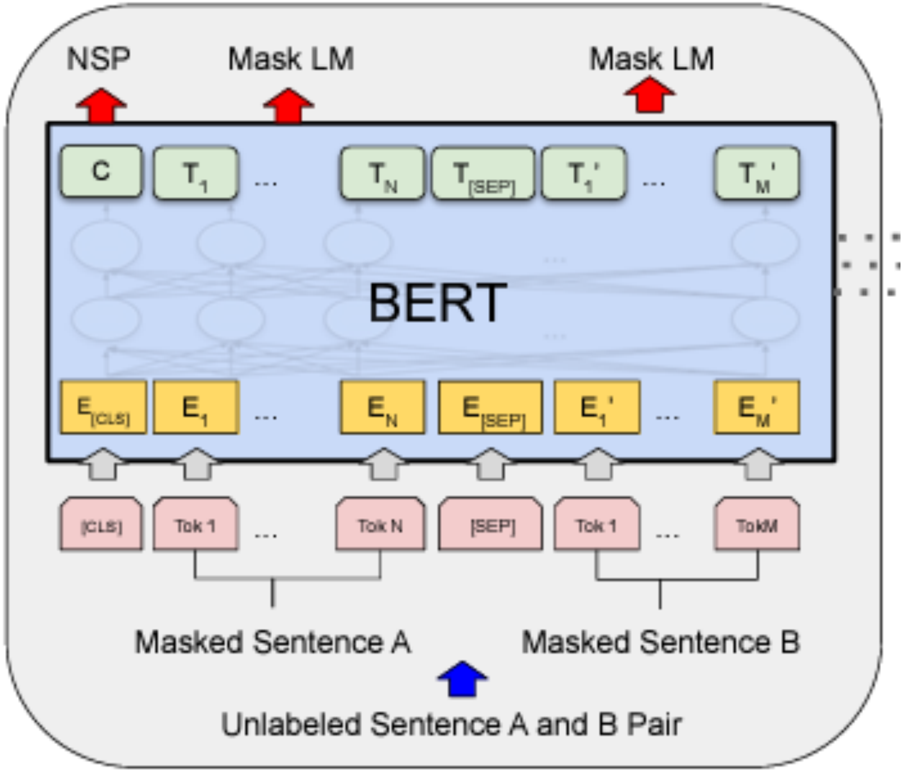
BERT Model inputs

- Such that the model can handle a variety of downstreamed tasks its input representation is such that which can unambiguously represent a single sentence or a pair of sentences.

- The first token of every sequence is classed a [CLS] token or a classification token.The final hidden state wrt to the [CLS] token is is used as the aggregate sequence representation for classification tasks.



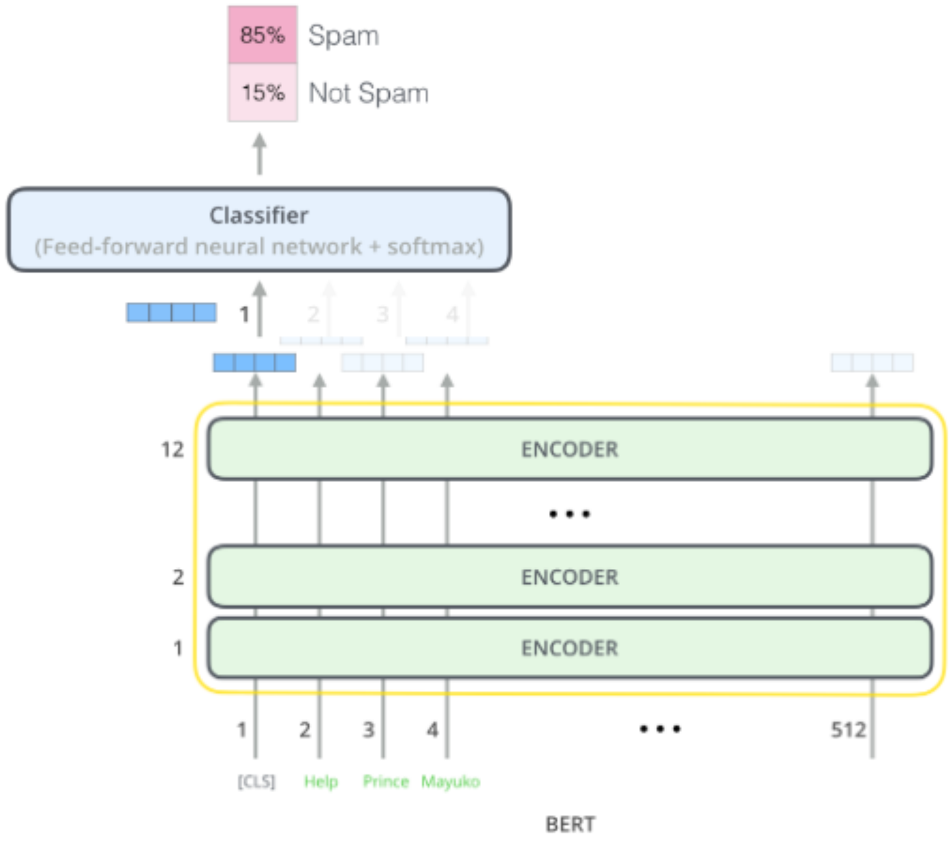
BERT Model inputs



Pre-training

- Now while training BERT we many use a single sentence of sentence pairs packed together into a single sequence . But we must figure out a way to differentiate two different sentences. This can be done using two methods:
    - Sentences separated by a special token called [SEP] token.
    - We could add a learned embedding to every token indicating which sentence it belongs to.

BERT Model Outputs



Each position outputs a vector of size *hidden_size* (768 in BERT Base).

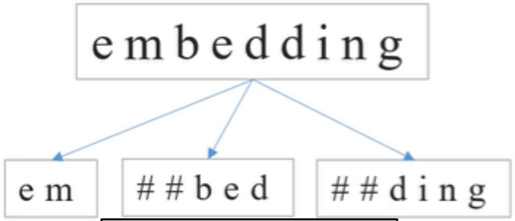
For the sentence classification example we've looked at above, we focus on the output of only the first position (that we passed the special [CLS] token to).

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.

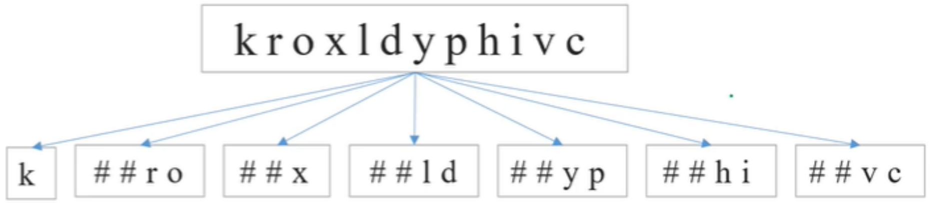
If you have more labels (for example if you're an email service that tags emails with “spam”, “not spam”, “social”, and “promotion”), you just tweak the classifier network to have more output neurons that then pass through softmax.

Word Piece Embeddings

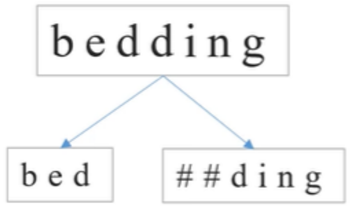
- The BERT model has a fixed vocabulary , so how does the model deal with words which are not present in the vocabulary.
- One of the approaches used in other language models is to use a [UNK] or unknown token to represent out of vocabulary words . But what BERT does is use the Word piece model and break down an unknown word into multiple known subwords. Ex1
- All subwords other the to first are preceded by ## .
- In a case where breaking down a word into meaningful subwords is not possible the word can be broken down into individual characters and some kind of a representation can be created for it. Ex2
- However this turns out to be very useful when we are able to break an unknown word into meaningful sub tokens. Ex 3



Example 1



Example 2



Example 3

BERT Model Pre-training Masked Language Modelling

Unfortunately, standard conditional language models can only be trained left-to-right or right-to-left, since bidirectional conditioning would allow each word to indirectly “see itself”, and the model could trivially predict the target word in a multi-layered context.

In order to train a deep bi directional representation, some percentage of the inputs are masked at random and then the masked tokens are to be predicted. The final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM.

In all of their experiments, they mask 15% of all Word Piece tokens in each sequence at random.

Although this allows them to obtain a bidirectional pre-trained model, a downside is that they are creating a mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning. To mitigate this, they do not always replace “masked” words with the actual [MASK] token.

BERT Model Pre-training Masked Language Modelling

Assuming the unlabeled sentence is *“ my dog is hairy”* and during the random masking procedure chose the 4th token (which corresponds to hairy) , our masking procedure can be further illustrated by:

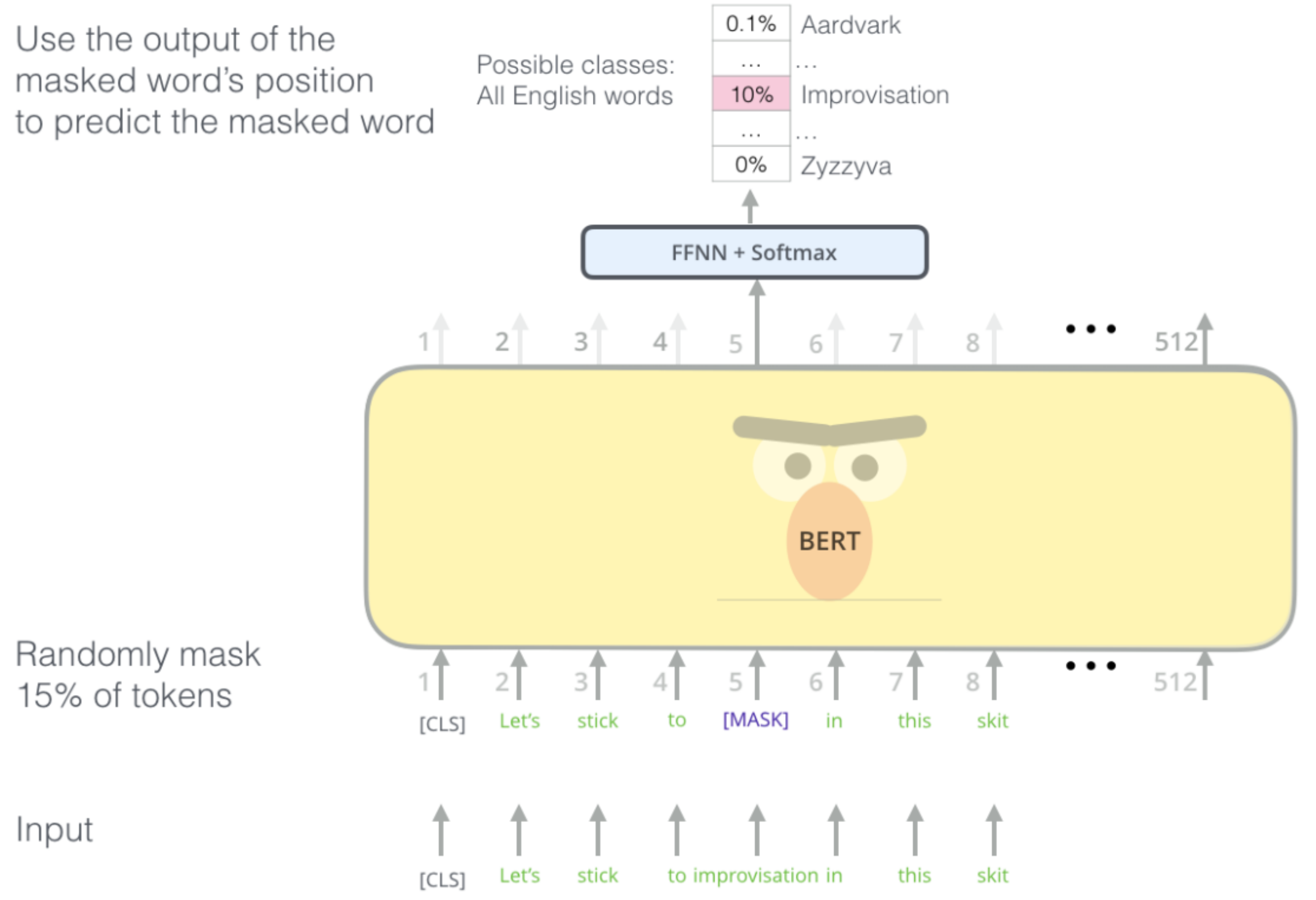
1. 80% of the time : Replace the word with the [MASK] token,
   E.g., my dog is hairy→ my dog is [MASK]

1. 10% of the time : Replace the word with a random word,
   E.g.,my dog is hairy→my dog is apple

1. 10% of the time: Keep the word unchanged,
   E.g.,my dog is hairy→ my dog is hairy.
   (The purpose of this is to bias the representation towards the actual observed word.)

The advantage of this procedure is that the Transformer encoder does not know which words it will be asked to predict or which have been replaced by random words, so it is forced to keep a distributional contextual representation of every input token. Additionally, because random replacement only occurs for 1.5%of all tokens (i.e., 10% of 15%), this does not seem to harm the model’s language understanding capability.

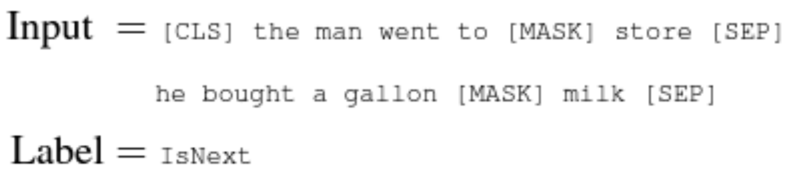
Masked Language Modelling


Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
... ...
10% Improvisation
... ...
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
Randomly mask 15% of tokens
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

BERT's clever language modeling task masks 15% of words in the input and asks the model to predict the missing word.

BERT Model Pre-training Next Sentence Prediction

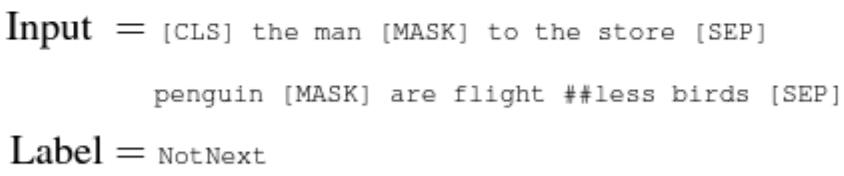
- Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two sentences.
- This is not directly captured by masked language modeling.
- Inorder to train a model that understands sentence relationships, we pre-train for a binarized next sentence prediction task that can be trivially generated from any monolingual corpus.
- Specifically, when choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as Not Next).



Input = [CLS] the man went to [MASK] store [SEP]

he bought a gallon [MASK] milk [SEP]

Label = IsNext



Input = [CLS] the man [MASK] to the store [SEP]

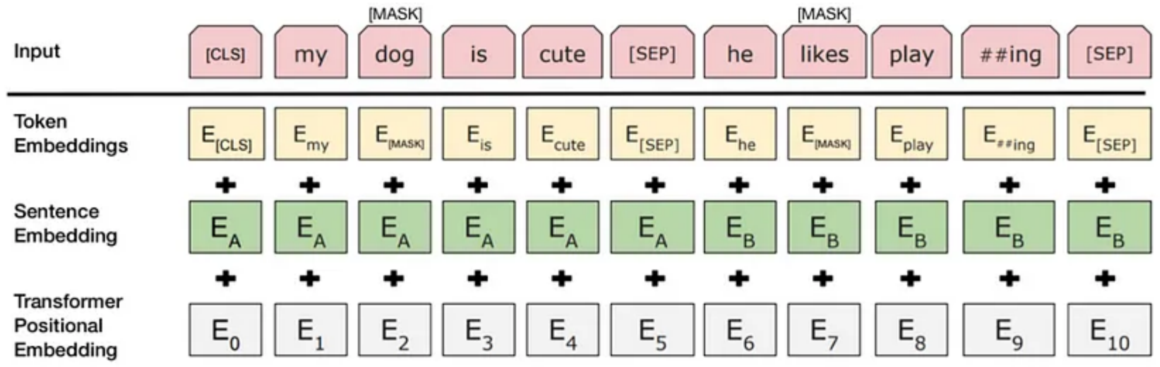
penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

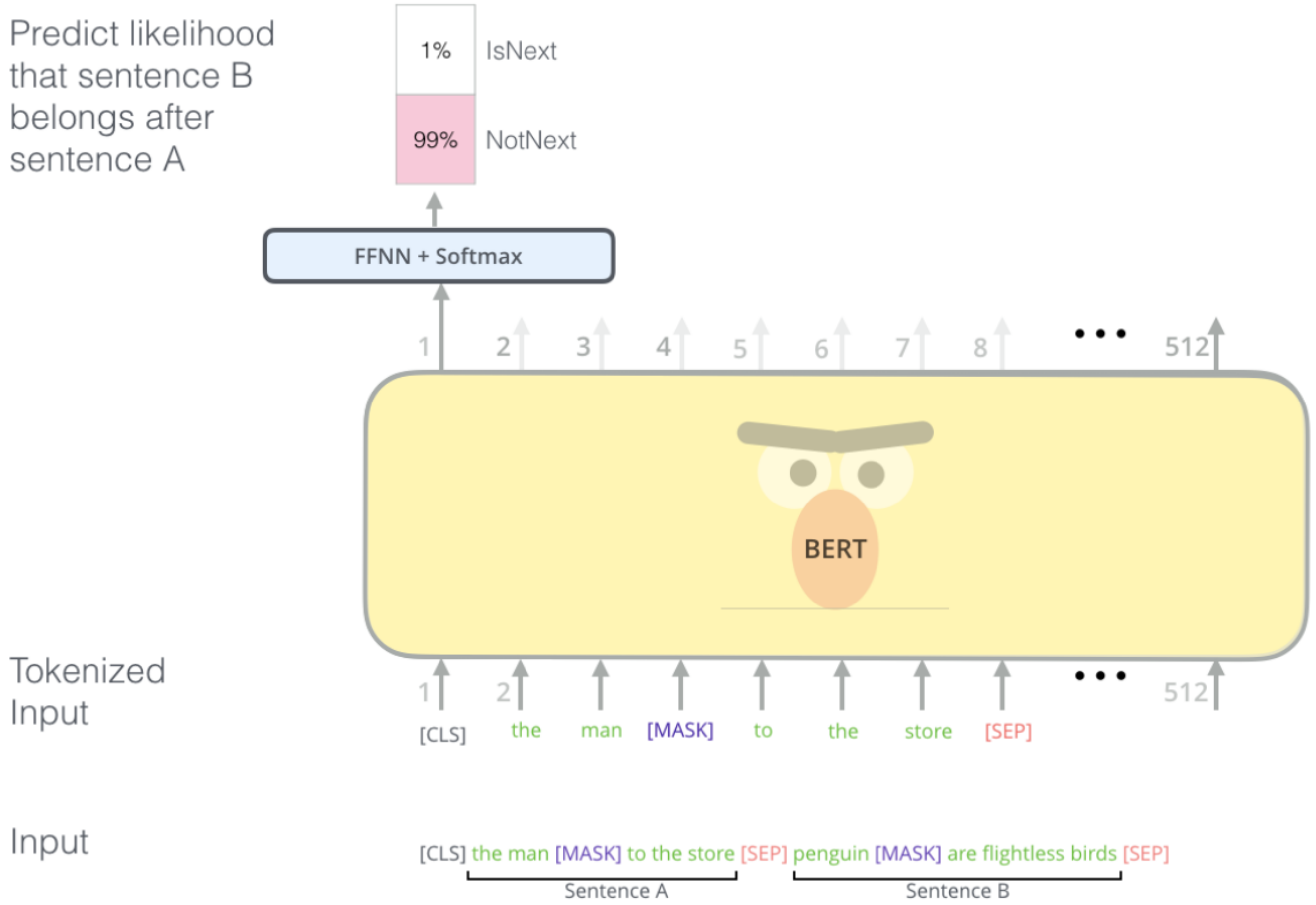
BERT Model Pre-training Next Sentence Prediction

To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:

1. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
3. A positional embedding is added to each token to indicate its position in the sequence. The concept and implementation of positional embedding are presented in the Transformer paper.


Input
[MASK]
[MASK]
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

To predict if the second sentence is indeed connected to the first, the following steps are performed:


Predict likelihood that sentence B belongs after sentence A
1%
IsNext
99%
NotNext
FFNN + Softmax
1
2
3
4
5
6
7
8
512
BERT
Tokenized Input
1
2
512
[CLS]
the
man
[MASK]
to
the
store
[SEP]
Input
[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flightless birds [SEP]
Sentence A
Sentence B

1. The entire input sequence goes through the Transformer model.
2. The output of the [CLS] token is transformed into a 2×1 shaped vector, using a simple classification layer (learned matrices of weights and biases).
3. Calculating the probability of Is Next Sequence with softmax.

BERT Model Pre-training details

- The pre-training procedure largely follows the existing literature on language model pre-training.
  - They train with batch size of 256 sequences for about 40 epochs on the 2.3 billion word corpus.
  - They use Adam with a learning rate of 1e-4, β1=0.9, β2=0.999, L2 weight decay of 0.01, learning rate warmup over the first 10,000 steps,and linear decay of the learning rate.
  - They also use a dropout probability of 0.1 on all layers.

- For the pre-training corpus we use theBooksCorpus (800 Words) (Zhu Et al., 2015) and the English Wikipedia (2,500Mwords).

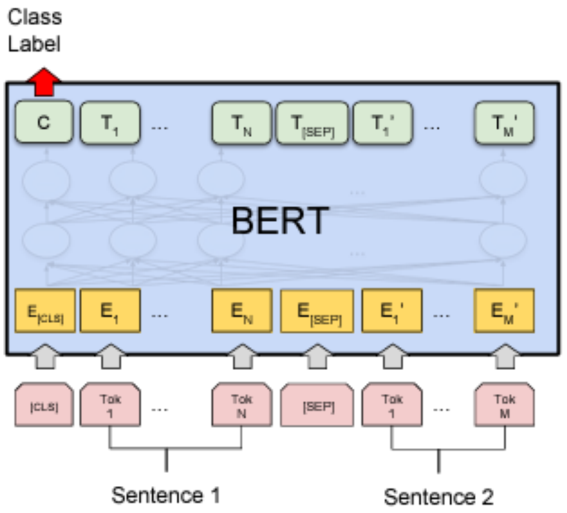
- To generate each training input sequence, they sample two spans of text from the corpus,which they refer to as “sentences” even though they are typically much longer than single sentences (but can be shorter also).
- The first sentence receives the A embedding and the second receives the B embedding. They are sampled such that the combined length is ≤ 512 tokens. The LM masking is applied after Word Piece tokenization with a uniform masking rate of 15%.

- Training of BERTBASE was performed on 4 Cloud TPUs in Pod configuration(16 TPU chips total). 13 Training of BERTLARGE was performed on 16 Cloud TPUs (64 TPU chips total). Each pre training took 4 days to complete.

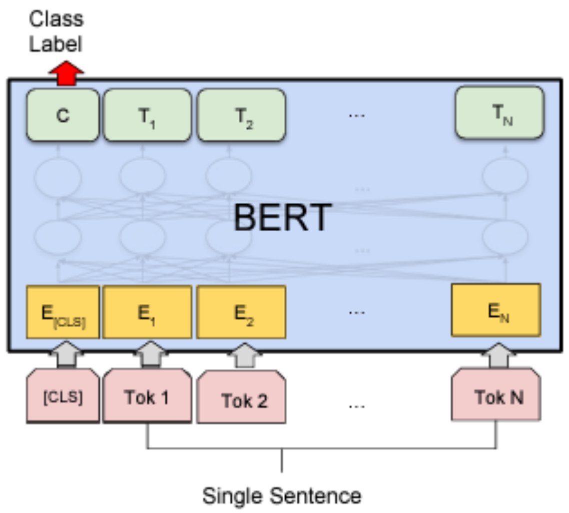
BERT Fine Tuning

BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model:

1. Classification tasks such as sentiment analysis are done similarly to Next Sentence classification, by adding a classification layer on top of the Transformer output for the [CLS] token.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC,

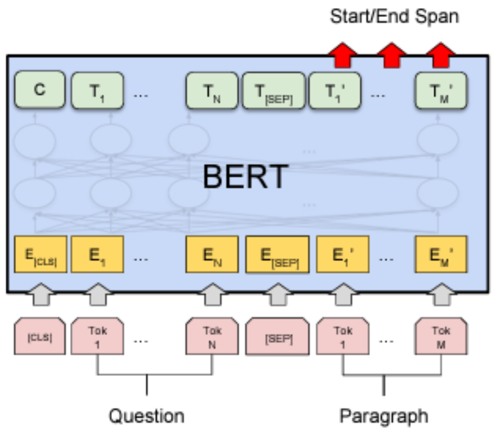


(b) Single Sentence Classification Tasks: SST-2, CoLA

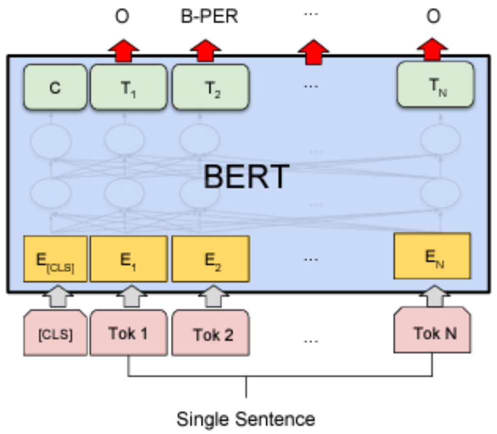
BERT Fine Tuning

2. In Question Answering tasks (e.g. SQuAD v1.1), the software receives a question regarding a text sequence and is required to mark the answer in the sequence. Using BERT, a Q&A model can be trained by learning two extra vectors that mark the beginning and the end of the answer.

3. In Named Entity Recognition (NER), the software receives a text sequence and is required to mark the various types of entities (Person, Organization, Date, etc) that appear in the text. Using BERT, a NER model can be trained by feeding the output vector of each token into a classification layer that predicts the NER label.



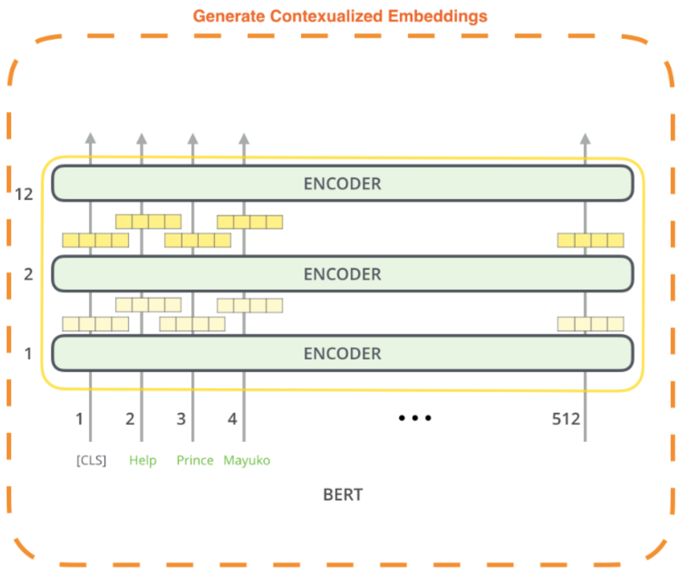
(c) Question Answering Tasks: SQuAD v1.1

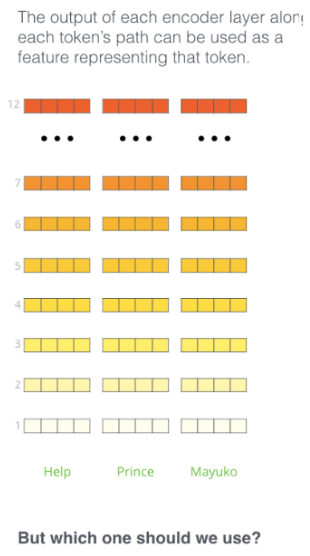


(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Feature based approach with BERT

The fine-tuning approach isn't the only way to use BERT. Just like ELMo, you can use the pre-trained BERT to create contextualized word embeddings. Then you can feed these embeddings to your existing model – a process the paper shows yield results not far behind fine-tuning BERT on a task such as named-entity recognition.




The output of each encoder layer along each token's path can be used as a feature representing that token.
12
7
6
5
4
3
2
1
Help
Prince
Mayuko
But which one should we use?

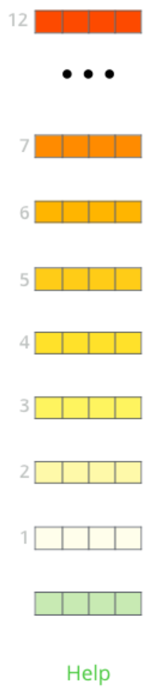
Which vector works best as a contextualized embedding? I would think it depends on the task. The paper examines six choices (Compared to the fine-tuned model which achieved a score of 96.4):

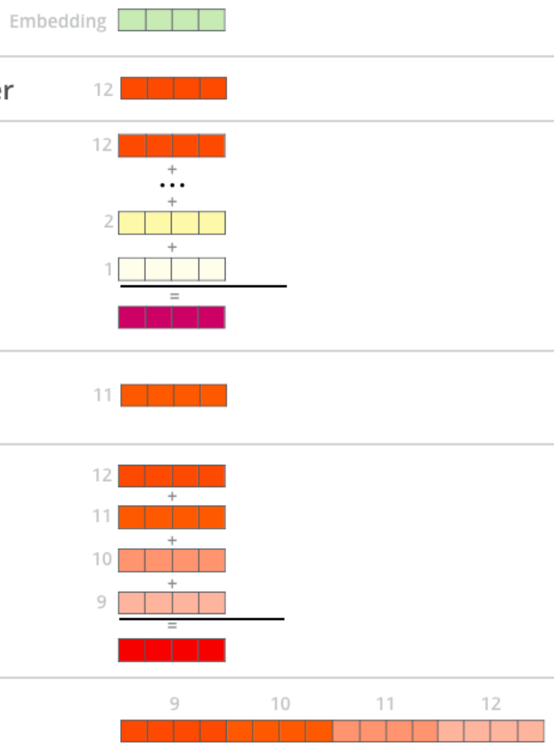
The feature-based approach here comprised of extracting the activations (or contextual embeddings or token representations or features) from one or more of the 12 layers without fine-tuning any parameters of BERT.

These embeddings are then used as input to a BiLSTM followed by the classification layer for NER. The authors report that when they concatenate the token representations from the top four hidden layers of the pre-trained Transformer and use that directly in the downstream task, the performance achieved is comparable to fine-tuning the entire model (including the parameters of BERT).

What is the best contextualized embedding for “Help” in that context?

For named-entity recognition task CoNLL-2003 NER





	Dev F1 Score
First Layer	91.0
Last Hidden Layer	94.9
Sum All 12 Layers	95.5
Second-to-Last Hidden Layer	95.6
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1

Conclusion

The best way to try out BERT is through the BERT Fine Tuning with Cloud TPUs notebook hosted on Google Colab. If you've never used Cloud TPUs before, this is also a good starting point to try them as well as the BERT code works on TPUs, CPUs and GPUs as well.

We hope this preliminary understanding of BERT was useful we would encourage students to read the paper[1] cited in the references and also refer to the blog posts and videos linked in the same.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’, *arXiv [cs.CL]*. 2019.

[2] https://www.youtube.com/playlist?list=PLam9sigHPGwOBuH4_4fr-XvDbe5uneaf6

[3] https://www.youtube.com/watch?v=-9evrZnBorM

[4] https://trishalaneeraj.github.io/2020-04-04/feature-based-approach-with-bert

[5] https://jalammar.github.io/illustrated-bert/

UE21CS343BB2
Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Anashua Dastidar, Teaching Assistant