BERT

&

Evaluation Metrics

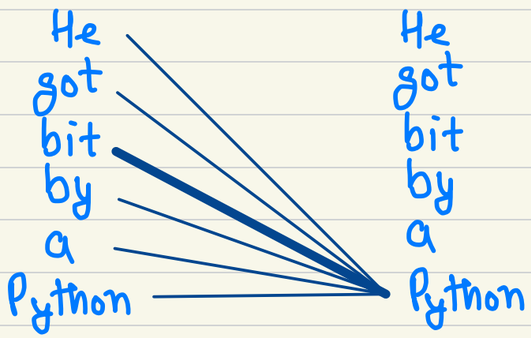
By: Kanav Bansal
(ThatAIGuy)

Bi-Directional Encoder Representation from Transformers (BERT)

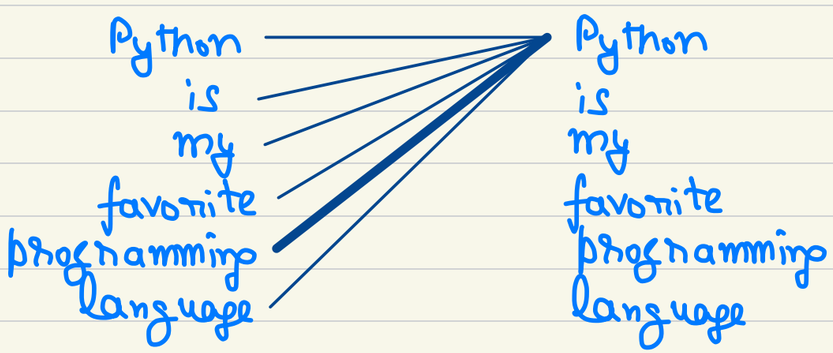
By: Kanav Bansal (That AI Guy)

* By Google in 2018
* Context Based Embedding Model

Ex: Sent A → He got bit by a Python



Sent B → Python is my favorite programming language



* BERT model is pretrained on a huge corpus using two interesting tasks, called Masked Language Modeling & Next Sentence Prediction.

* For any task, say question-answering, instead of training BERT from scratch, we will use pre-trained BERT model and adjust (fine-tune) its weights for the new task

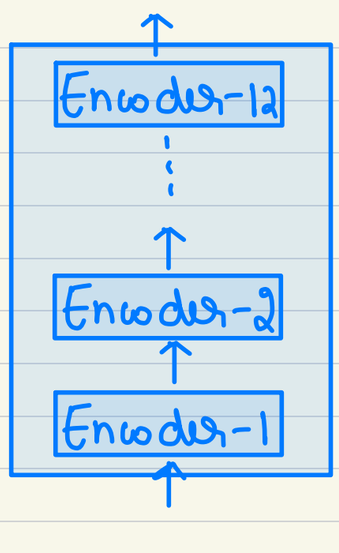
By: Kanav Bansal
(That AI Guy)

Various BERT Configurations

BERT-base vs BERT-large

1. BERT-base
   - 12 encoder layers, stacked on top of the other → L
   - Each encoder with 12 attention heads → A
   - 768 hidden units in FFNN. → H
   - Embedding size obtained is 768

∴ BERT-base :
L → 12 , A → 12 , H → 768
Parameters = 110 million



2. BERT-large :
L → 24, A → 16 , H → 1024
Parameters = 340 million

BERT - Uncased vs BERT-Cased

1. BERT-uncased → All tokens are lower-cased.

2. BERT-cased → Tokens are not lower-cased & used directly for training.

By: Kanav Bansal
(That AI Guy)

Input Data Representation

Before feeding the input to BERT, we convert the input into embeddings using the three embedding layers indicated by:

1. Token embedding
2. Segment embedding
3. Position embedding

Token Embedding

Sentence A: I am learning GenAI.
Sentence B: BERT is powerful.

Tokenize both the sentences & obtain tokens.
Note: BERT uses WordPiece Tokenizer. It helps with OOV.
tokens = [I, am, learning, GenAI, BERT, is, powerful]

Next add two special tokens:
[CLS] → Added at the beginning of first sentence. Represent classification task.
[SEP] → Added at the end of each sentence.

Input Tokens = [[CLS], I, am, learning, GenAI, [SEP], BERT, is, powerful, [SEP]]

Finally, convert each token to embeddings.
Token Embeddings = $E_{CLS}, E_I, E_{am}, E_{learning}$ ------ $E_{powerful}, E_{SEP}$

Segment Embedding

Apart from [SEP] token, we have to give some sort of indicator to our model to distinguish between the two sentences. For this input tokens are feed to the segment embedding layer.

The segment embedding returns only either of the two embeddings as an output:
E_A: If input token belongs to sentence A.
E_B: If input token belong to sentence B.

Input Tokens = [[CLS], I, am, learning, GenAI, [SEP], BERT, is, powerful, [SEP]]

Segment Embedding = [$E_A, E_A, E_A, E_A, E_A, E_A, E_B$ E_B, E_B, E_B]

By: Kanav Bansal
(That AI Guy)

Position Embedding

Since transformer doesnot use any recurrence mechanism & processes all the tokens in parallel, we need to provide some information relating to word order, so we use Positional Encoding.

Input Tokens = [[CLS], I, am, learning, GenAI, [SEP], BERT, is, powerful, [SEP]]

Position Embedding = [E_0, E_1, E_2, E_3, E_4, E_5, E_6 E_7, E_8, E_9]

Final Representation

Sum up all the embeddings & feed them as input to BERT.

Token Embedding + Segment Embedding + Position Embedding

WordPiece Tokenizer

* It follows the subword tokenization scheme.
* It helps handle OOV words.

Ex: Let us start pretraining the model.
tokens = [Let, us, start, pre, ##train, ##ing, the, model]
Adding [CLS] & [SEP] 7

Input Tokens = [[CLS], Let, us, start, pre, ##train, ##ing, the, model]

* How it works?
first it checks whether the word is present in BERT Vocabulary. If the word is present in the vocabulary, use it as it is. Otherwise, the word is splitted into subwords & we check if the subword is present in the vocabulary or not. If not the subword is again splitted.
In our example pretraining is splitted into the subwords → pre, ##train, ##ing
Note that hash sign before the tokens indicate that it is a subword.

* Size of BERT Vocabulary is 30K tokens.

BERT Pre-Training Strategies

By: Kanav Bansal
(That AI Guy)

Q: How is BERT trained from Scratch?

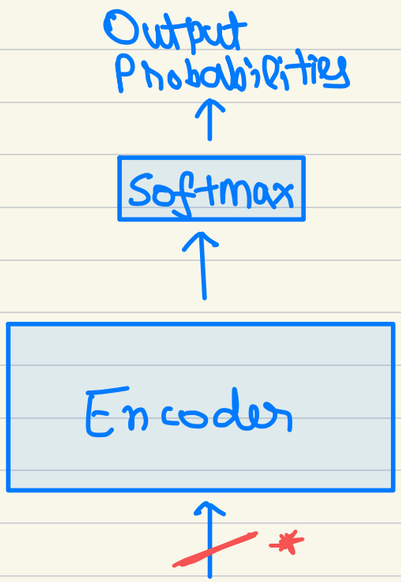
Wikipedia + & Google Books — Huge Data

Encoder only Transformer — Powerful Architecture

+

MLM & NSP — Auto-Encoding LM

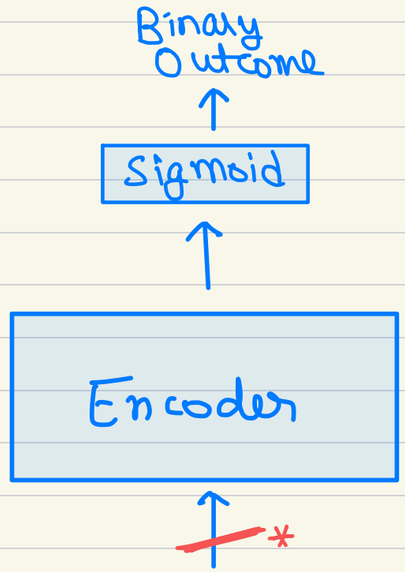
Q: What is Masked Language Modeling?



[CLS] I am learning [MASK] Modeling [SEP]

* Given input tokens, in MLM, we mask 15% of the tokens & train the network to predict the masked tokens.
* Encoder reads the sentence in both directions & tries to predict the masked tokens.

Q: What is Next Sentence Prediction?



[CLS] GPT is a LLM. [SEP] Full form is Bidirectional Encoder Rep from Transformer [SEP]

* Input:

Word Piece Tokenization + Segment Encoding + Positional Encoding = Input Encoding

Q: What activation function does BERT use?

Gaussian Error Linear Unit (GELU)

It is smoother & results in stable training when compared to ReLU.

The 80-10-10 % rule ↓

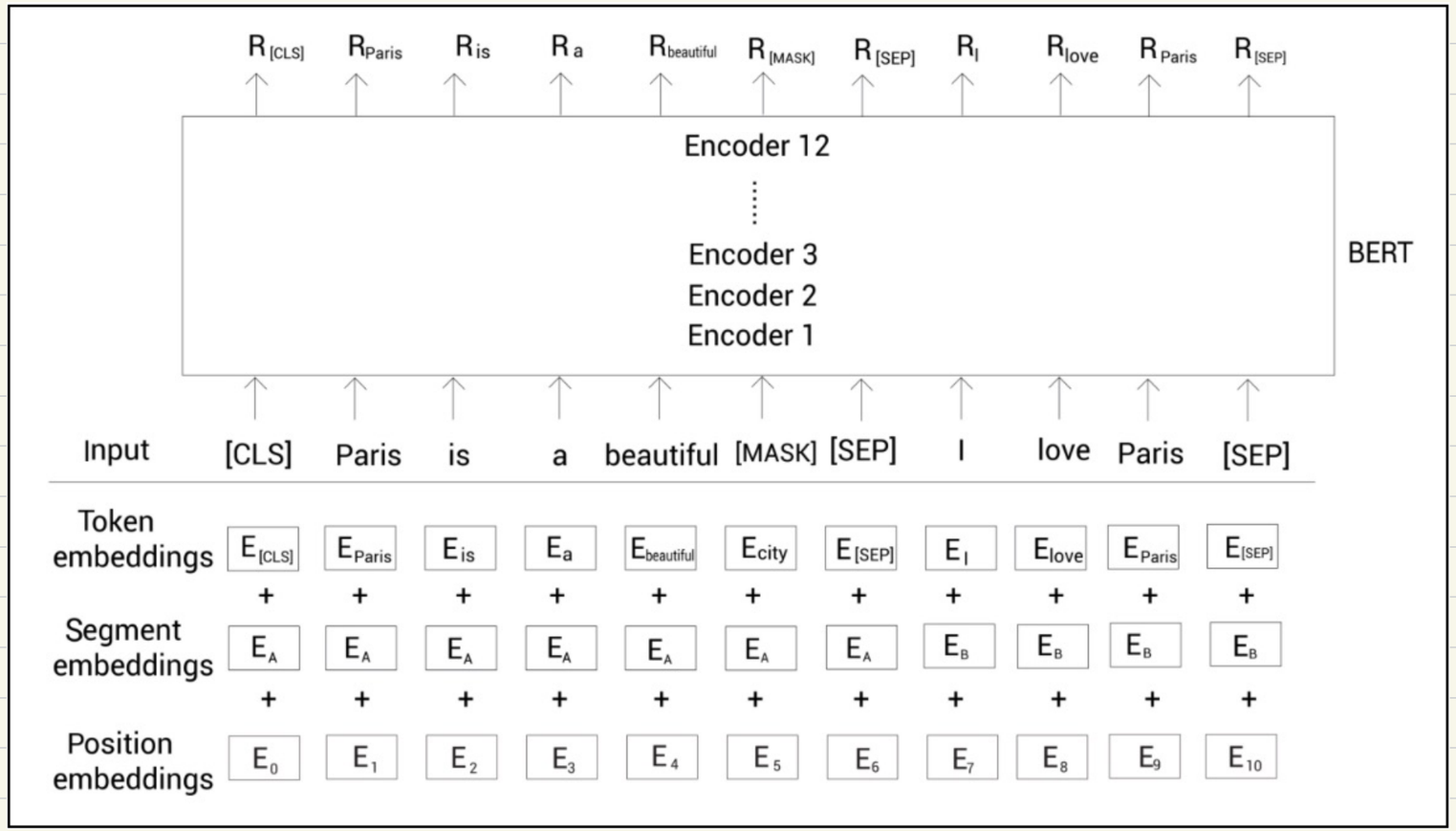
By: Kanav Bansal
(That AI Guy)

In MLM, we mask 15% of the tokens.
We apply 80-10-10% on top of these 15% tokens.

80%: We replace the token with the [MASK] token.

10%: We replace the token with a random token.

10%: We don't make any changes.



Conclusion ↴

By: Kanav Bansal
(That AI Guy)

* BERT is context based model unlike w2v which is context free.

* There can be different BERT configurations BERT-base & BERT-large.

* BERT is pre-trained using two interesting tasks → Masked Language Modelling
  → Next Sentence Prediction

* In MLM, we mask 15% of the tokens.

* BERT use sub-word tokenization algorithm → WordPiece.

Evaluation Metrics

By: Kanav Bansal
(That AI Guy)

1. Perplexity
It is used to evaluate the performance of Language Models for quality of text generation. Lower perplexity indicates that the model is better at predicting the sequence of words.

2. Bilingual Evaluation Understudy (BLEU)
BLEU is a metric used to evaluate the quality of machine-generated translation by comparing with one or more reference translation. It try measuring the overlap of n-grams. High BLEU score is close to 1 & Low BLEU score is close to 0.

3. Metric for Evaluation of Translation with Explicit ORdering (METEOR)
In addition to exact matches, it also considers synonymy, stemming & paraphrasing to evaluate Machine Translation task.

4. Recall-Oriented Understudy for Gisting Evaluation (ROUGE)
Used to evaluate text summarization by comparing the overlap of n-grams, word sequences & word pairs b/w system-generated & a reference summary

5. General Language Understanding Evaluation (GLUE)
GLUE is a benchmark used to evaluate NLP models performance on variety of Language Understanding tasks. It uses set of diverse tasks like sentiment analysis, sentence similarity etc...
Model is evaluated on each task & the performance is aggregated to get overall score.

GLUE

By: Kanav Bansal
(That AI Guy)

GLUE includes a variety of tasks that tests different language understanding abilities:

1. CoLA (Corpus of Linguistic Acceptability)
   Task: Determines grammatical correctness

2. SST-2 (Stanford Sentiment Treebank)
   Task: Binary sentiment classification

3. MRPC (Microsoft Research Paraphrase Corpus)
   Task: Determines whether two sentences are paraphrases of each other

4. STS-B (Semantic Textual Similarity Benchmark)
   Task: Measures semantic similarity between two sentences

5. QQP (Quora Question Pairing)
   Task: Measures semantic similarity of two questions

6. MNLI (Multi-Genre Natural Language Inference)
   Task: Classifies Premise & Hypothesis as entailed, contradicted or neutral.

7. QNLI (Question Natural Language Inference)
   Task: Determines if a sentence contains answer to given question

8. RTE (Recognizing Textual Entailment)
   Task: Binary classification of whether a given hypothesis follow logically from a given premise

9. WNLI (Winograd Schema Challenge)
   Task: Reasoning about pronoun reference in sentences.

Note:

- Paraphrase: Two sentences with different words but similar meaning.
- Premise: Initial statement which is assumed to be true.
- Hypothesis: Sentence which is evaluated against the premise
- Entailed: Hypothesis logically follows premise
- Contradicted: Hypothesis is logically inconsistent with premise.

By: Kanav Bansal
(That AI Guy)