BERT Algorithm

Chavi
& Rukmini

What is BERT?

- Bidirectional Encoder Representations from Transformers
- Machine Learning framework for Natural Language Processing (NLP)
- Helps to learn contextual relationships between words

HOW DOES BERT WORK?

- Tokenisation
- Bi-directional Transformer Encoder
- Pre Training:
    a. Masked Model Language
    b. Next Sentence Prediction
- Fine-Tuning on Tasks

Tokenisation

Input sentence:

"I loved the movie, it was amazing!"

Tokenised sentence:

["I", "loved", "the", "movie", ",", "it", "was", "amazing", "!"]

The WordPiece algorithm further splits each word in the input text into subwords. For instance, the word "loved" may be split into "love" and "##d", where "##d" represents a suffix.

```
Token: I
Embedding: [0.12, 0.45, -0.63, ...]

Token: loved
Embedding: [0.53, -0.23, 0.17, ...]

Token: the
Embedding: [0.04, -0.56, 0.89, ...]

Token: movie
Embedding: [0.87, 0.23, -0.42, ...]
```

Word Embedding

Transformer Encoder

```
      I  loved  the  movie,  it  was  amazing!
      |      |    |      |    |     |      |
(Token Embedding layer)
      |      |    |      |    |     |      |
 Embedding1 Embedding2 Embedding3 ... EmbeddingN
      |      |    |      |    |     |      |
      |      |    |      |    |     |      |
(Transformer layer 1)
      |      |    |      |    |     |      |
Embedding1' Embedding2' Embedding3' ... EmbeddingN'
      |      |    |      |    |     |      |
      |      |    |      |    |     |      |
(Transformer layer 2)
      |      |    |      |    |     |      |
Embedding1'' Embedding2'' Embedding3'' ... EmbeddingN''
      |      |    |      |    |     |      |
```

```
             +----------+
             | Input    |
             | Embedding |
             +----------+
                  |
                  |
             +----------+
             | Transformer|
             | Encoder   |
             +----------+
              /          \
             /            \
  +----------+        +----------+
  | Self-     |        | Feed-     |
  | Attention |        | Forward   |
  +----------+        +----------+
        \                  /
         \                /
          +------------+
          | Contextual  |
          | Embeddings |
          +------------+
```

Pre-Training

- MLM: Masked Model Language

  Original sentence: I loved the movie, it was amazing!
  Masked sentence: [MASK] loved the movie, it was amazing!

- NSP: Next Sentence Prediction

  I love Pizza. Pizza is my favorite food. (correct sentence pair)
  The cat sat on the mat. It was a sunny day (incorrect sentence pair)

Fine-Tuning

- Fine-tuning involves training the model on a specific task with task-specific labeled data.
- To fine-tune the model, we first add a classification layer on top of the pre-trained BERT model.

Summary

```
Input Sentence: "I loved the movie, it was amazing!"

+------------------+
| Tokenization     |
+------------------+
         |
         v
Tokenized Sequence: ["[CLS]", "I", "loved", "the", "movie", ",", "it", "was", "amazing", "!", "[SEP]"]

+------------------+
| BERT Encoding    |
+------------------+
         |
         v
Contextualized Embeddings: [CLS] embedding, I embedding, loved embedding, the embedding, movie embedding, , embedding, it embedding, was embedding, amazing embedding, ! embedding [SEP]

+------------------+
| Classification   |
+------------------+
         |
         v
Prediction: Positive
```

BERT MODEL SIZE & ARCHITECTURE

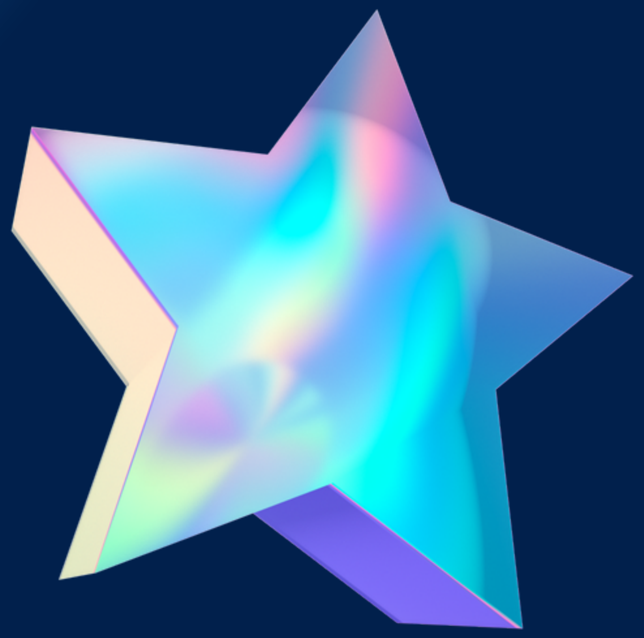
BERT-Base

- 12 transformer layers
- 110 million parameters
- can process input sequences of 512 tokens

BERT-Large

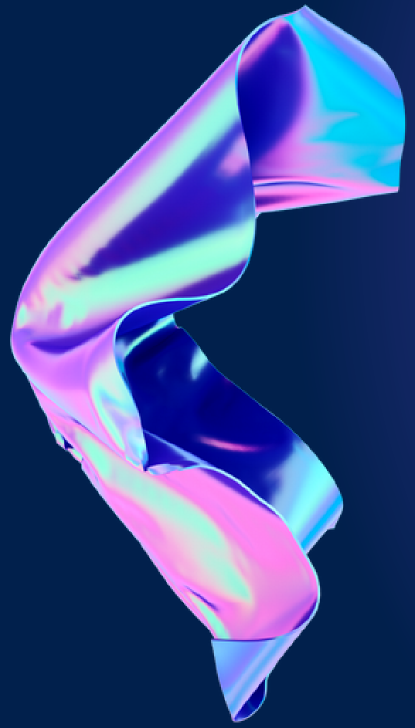
- 24 transformer layers
- 340 million parameters
- can process input sequences of 1024 tokens

BERT'S PERFORMANCE



93.1
SQuAD v1.1 & v2.0

86.3
SWAG



82.1
GLUE

BERT'S PERFORMANCE

BERT’s Performance - SQuAD1.1 Leaderboard

Rank	Model	EM	F1
1 **[Oct 05, 2018]**	**BERT (ensemble)** Google AI Language **arrive.org/abs/180.04805**	**87.433**	**93.16**
-	**Human Performance** Stanford University (Rajpurkar et al. ’16)	82.304	91.221
2 **[Sep 09, 2018]**	**nlnet (ensemble)** Microsoft Research Asia	85.356	91.202
3 **[Jul 11, 2018]**	**QANet (ensemble)** Google Brain & SMU	84.454	90.490

BERT'S PERFORMANCE

BERT's Performance - SWAG (Situations With Adversarial Generations)

System	Dev	Test
$BERT_{LARGE}$	**86.6**	**86.3**
Human (expert)	-	85.0
OpenAI GPT	-	78
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2

BERT'S PERFORMANCE

BERT's Performance on GLUE:

Task	Average	Grammatical	Sentiment Analysis	Similarity	Paraphrase	Question Similarity	Contradiction	Answerable	Entail
BERTLARGE	**82.1**	**60.5**	**94.9**	**86.5**	**89.3**	**72.1**	**86.7/85.9**	**92.7**	**70.1**
BERTBASE	79.6	52.1	93.5	85.8	88.9	71.2	84.6/83.4	90.5	66.4
OpenAI GPT	75.1	45.4	91.3	80.0	82.3	70.3	82.1/81.4	87.4	56.0
Pre-OpenAI SOTA	74.0	35.0	93.2	81.0	86.0	66.1	80.6/80.1	82.3	61.7
BiLSTM+ELMo+Attn	71.0	36.0	90.4	73.3	84.9	64.8	76.4/76.1	79.8	56.8

ENVIRONMENTAL IMPACT

Energy usage

- Bert requires large amount of computational resources like GPU's and TPU's which consume a significant amount of energy.

Indirect impacts

- The requirement of data centers and cloud computing infrastructure contribute to the global carbon footprint.

Open Source Power of BERT

- BERT's source code is publicly accessible that allows BERT to be more widely used all around the world.
- Thousands of open-source and free, pre-trained BERT models are currently available for specific use cases if you don't want to fine-tune BERT

How to use BERT

- **Prepare the Data**
  - Data Collection
  - Text Cleaning
  - Text Preprocessing
  - Labeling
  - Splitting the Data

```
# Pre-Processing --------------------------------------------------------
# Import Data from Kaggle
data1 <- read.csv("train.csv")
data2 <- read.csv("test.csv")
data <- bind_rows(data1,data2)

# Data Pre-Processing
data$tweet <- gsub("[^[:alpha:]]", " ", data$tweet) #Remove non-alphabetic c
data$tweet  <- gsub("[[:space:]]+", " ", data$tweet) #Remove spaces
data$tweet <- trimws(data$tweet) #Remove trailing whitespaces
```

How to use BERT

- **Fine-tune BERT**
  - Load the pre-trained BERT model
  - Prepare the data
  - Define the fine-tuning task
  - Initialize the classification layer
  - Train and evaluate the model
  - Predict on new data

How to use BERT

- **Prediction**
  - After the model is trained, we can input text data into the BERT model and this will display an output which is the probability distribution over the sentiment labels

Common BERT Queries

What makes Bert different

Time taken to fine-tune Bert

Can BERT be used with Tensorflow?

conclusion

BERT takes into account the entire context of a sentence using the bidirectional training

Uses MLM modeling, hence it is trained to predict based on context

The performance of this algorithm can be made better through fine tuning

BERT has had a big impact on NLP and continues to be a big area of research as it is still improved upon