# BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

## Paper Presentation

Isaac Kobby Anni

Computer Science
Bowling Green State University

November 18, 2024

# Prerequisite

- Knowledge of recurrent neural networks and how they work.

# Prerequisite

- Knowledge of recurrent neural networks and how they work.
- Attention is all you need paper.

# Introduction

- RNNs like LSTM and GRU were established as the state of the art for sequence modeling.

# Introduction

- RNNs like LSTM and GRU were established as the state of the art for sequence modeling.
- For problems like language modeling and machine translation.

# Introduction

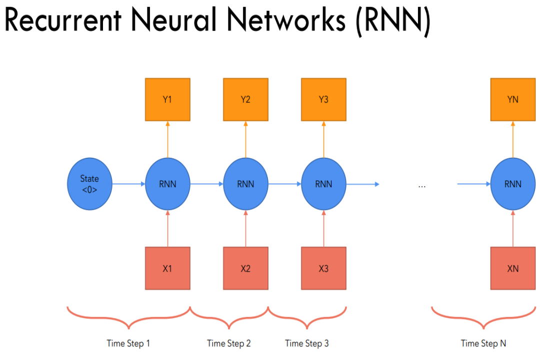



Figure 1: Workflow of RNN: *photo credit, Umar Jamil*

# Introduction

- Problem as sequence length increases;

# Introduction

- Problem as sequence length increases;

i Slow computation

# Introduction

- Problem as sequence length increases;

i Slow computation

ii Varnishing and Exploding gradients

# Introduction

- Problem as sequence length increases;

i Slow computation

ii Varnishing and Exploding gradients

iii Difficulty in accessing information from longer past time step

1. Background
2. Introducing Transformer
   - Attention is all you need.
3. BERT

1. Background
2. Introducing Transformer
   Attention is all you need.
3. BERT

# Transformer Architecture

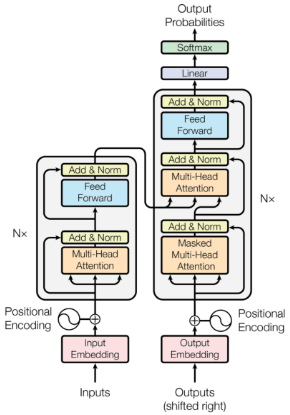



Figure 2: Transformer Model: *photo credit, Attention paper*

# The Encoder

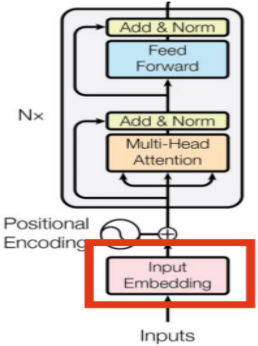



Figure 3: Transformer Model: *photo credit, Attention paper*

# Input Embedding


Figure 4: Document Representation: *photo credit, Umar Jamil*

# Input Embedding

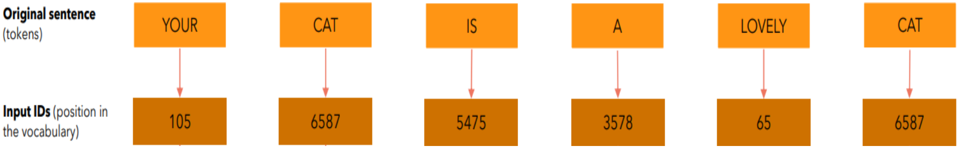



Figure 5: Document Representation

# Input Embedding

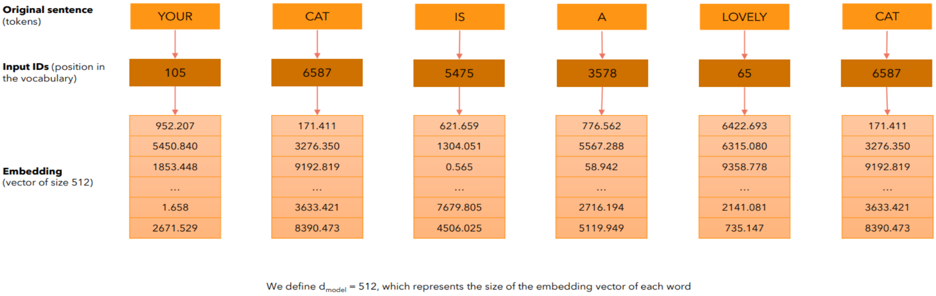
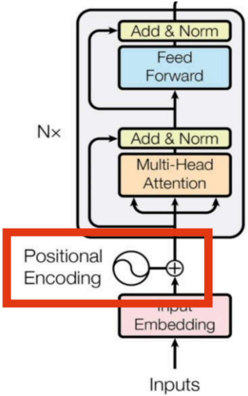



Figure 6: Document Representation

# Positional Embedding



# Positional Embedding

- Addressing the sequential encoding like in RNNs;

# Positional Embedding

- Addressing the sequential encoding like in RNNs;

i Position of each word in the document.

# Positional Embedding

- Addressing the sequential encoding like in RNNs;

i Position of each word in the document.

ii Treating words that are close as "close" and distant as "distant"

# Positional Embedding

- Addressing the sequential encoding like in RNNs;

i Position of each word in the document.

ii Treating words that are close as "close" and distant as "distant"

iii Position encoding portray patterns learned by the model.

# Positional Embedding

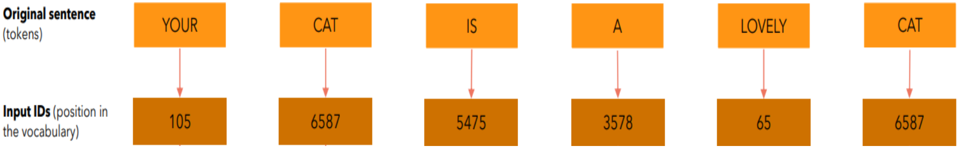



Figure 8: Document Representation

# Positional Embedding

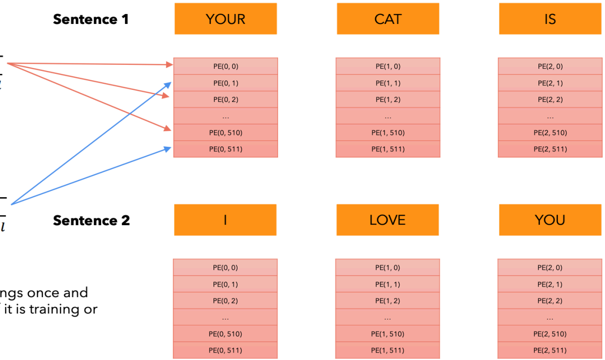
| Original sentence | YOUR | CAT | IS | A | LOVELY | CAT |
|---|---|---|---|---|---|---|
| **Embedding** (vector of size 512) | 952.207 | 171.411 | 621.659 | 776.562 | 6422.693 | 171.411 |
| | 5450.840 | 3276.350 | 1304.051 | 5567.288 | 6315.080 | 3276.350 |
| | 1853.448 | 9192.819 | 0.565 | 58.942 | 9358.778 | 9192.819 |
| | ... | ... | ... | ... | ... | ... |
| | 1.658 | 3633.421 | 7679.805 | 2716.194 | 2141.081 | 3633.421 |
| | 2671.529 | 8390.473 | 4506.025 | 5119.949 | 735.147 | 8390.473 |
| | + | + | + | + | + | + |
| **Position Embedding** (vector of size 512). Only computed once and reused for every sentence during training and inference. | ... | 1664.068 | ... | ... | ... | 1281.458 |
| | ... | 8080.133 | ... | ... | ... | 7902.890 |
| | ... | 2620.399 | ... | ... | ... | 912.970 |
| | ... | ... | ... | ... | ... | 3821.102 |
| | ... | 9386.405 | ... | ... | ... | 1659.217 |
| | ... | 3120.159 | ... | ... | ... | 7018.620 |
| | = | = | = | = | = | = |
| **Encoder Input** (vector of size 512) | ... | 1835.479 | ... | ... | ... | 1452.869 |
| | ... | 11356.483 | ... | ... | ... | 11179.24 |
| | ... | 11813.218 | ... | ... | ... | 10105.789 |
| | ... | ... | ... | ... | ... | ... |
| | ... | 13019.826 | ... | ... | ... | 5292.638 |
| | ... | 11510.632 | ... | ... | ... | 15409.093 |

Figure 9: Document Representation

# Positional Embedding Calculation

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

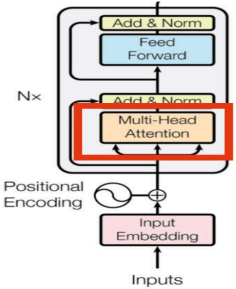
$$PE(pos, 2i+1) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$




We only need to compute the positional encodings once and then reuse them for every sentence, no matter if it is training or inference.

Figure 10: Document Representation

# Attention Mechanism



# Attention Mechanism

Self-Attention allows the model to relate words to each other.

In this simple case we consider the sequence length **seq** = 6 and $\mathbf{d_{model}} = \mathbf{d_k} = 512$.

The matrices **Q**, **K** and **V** are just the input sentence.

$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{softmax}\left(\frac{\underset{(6,\,512)}{\mathbf{Q}} \times \underset{(512,\,6)}{\mathbf{K^T}}}{\sqrt{512}}\right) =$$

| | YOUR | CAT | IS | A | LOVELY | CAT | Σ |
|---|---|---|---|---|---|---|---|
| YOUR | 0.268 | 0.119 | 0.134 | 0.148 | 0.179 | 0.152 | 1 |
| CAT | 0.124 | 0.278 | 0.201 | 0.128 | 0.154 | 0.115 | 1 |
| IS | 0.147 | 0.132 | 0.262 | 0.097 | 0.218 | 0.145 | 1 |
| A | 0.210 | 0.128 | 0.206 | 0.212 | 0.119 | 0.125 | 1 |
| LOVELY | 0.146 | 0.158 | 0.152 | 0.143 | 0.227 | 0.174 | 1 |
| CAT | 0.195 | 0.114 | 0.203 | 0.103 | 0.157 | 0.229 | 1 |

* all values are random.

Figure 12: Self Attention computation

# Attention Mechanism

$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

| | YOUR | CAT | IS | A | LOVELY | CAT |
|---|---|---|---|---|---|---|
| YOUR | 0.268 | 0.119 | 0.134 | 0.148 | 0.179 | 0.152 |
| CAT | 0.124 | 0.278 | 0.201 | 0.128 | 0.154 | 0.115 |
| IS | 0.147 | 0.132 | 0.262 | 0.097 | 0.218 | 0.145 |
| A | 0.210 | 0.128 | 0.206 | 0.212 | 0.119 | 0.125 |
| LOVELY | 0.146 | 0.158 | 0.152 | 0.143 | 0.227 | 0.174 |
| CAT | 0.195 | 0.114 | 0.203 | 0.103 | 0.157 | 0.229 |

(6, 6)

X

**V**

(6, 512)

=

**Attention**

(6, 512)

Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

Figure 13: Self Attention computation

# Properties of Self Atention

- No requires no parameter(s)

# Properties of Self Atention

- No requires no parameter(s)
- Values along diagonals are higher.

# Properties of Self Atention

- No requires no parameter(s)
- Values along diagonals are higher.
- Can switch off word interactions. etc

# Self-Attention Computation

$$softmax(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

| | YOUR | CAT | IS | A | LOVELY | CAT |
|---|---|---|---|---|---|---|
| YOUR | 0.268 | 0.119 | 0.134 | 0.148 | 0.179 | 0.152 |
| CAT | 0.124 | 0.278 | 0.201 | 0.128 | 0.154 | 0.115 |
| IS | 0.147 | 0.132 | 0.262 | 0.097 | 0.218 | 0.145 |
| A | 0.210 | 0.128 | 0.206 | 0.212 | 0.119 | 0.125 |
| LOVELY | 0.146 | 0.158 | 0.152 | 0.143 | 0.227 | 0.174 |
| CAT | 0.195 | 0.114 | 0.203 | 0.103 | 0.157 | 0.229 |

# Attention Mechanism

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$MultiHead(Q, K, V) = Concat(head_1 \ \ldots head_h)W^O$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Figure 14: Multi-Head Attention

# Multi-Head Attention Mechanism - DECODER

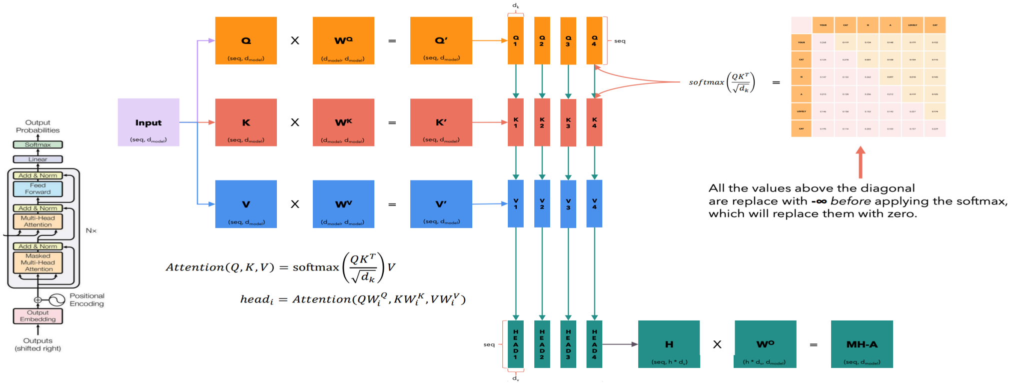



Figure 15: Masked Multi-head Attention computation

# Motivation

- Existing models like OpenAI GPT were unidirectional.

# Motivation

- Existing models like OpenAI GPT were unidirectional.
- Understanding context in both directions was a challenge.

# Goal

- To build a model that fully leverages context from both directions for improved language understanding.

# Model Architecture

- Architecture is made of layers of **ENCODER**'s of the transformer model.

## Model Architecture

- Architecture is made of layers of **ENCODER**'s of the transformer model.
- **BERT-base:**

# Model Architecture

- Architecture is made of layers of **ENCODER**'s of the transformer model.
- **BERT-base**:
    i 12 encoder layers

# Model Architecture

- Architecture is made of layers of **ENCODER**'s of the transformer model.
- **BERT-base**:
    - i 12 encoder layers
    - ii 12 attention heads

# Model Architecture

- Architecture is made of layers of **ENCODER**'s of the transformer model.
- **BERT-base**:
  - i 12 encoder layers
  - ii 12 attention heads
  - iii 3072 hidden size of feed-forward layer

# Model Architecture

- Architecture is made of layers of **ENCODER**'s of the transformer model.
- **BERT-base**:
    - i 12 encoder layers
    - ii 12 attention heads
    - iii 3072 hidden size of feed-forward layer
    - iv 768 embedding size

# Model Architecture

- **BERT-large:**

# Model Architecture

- **BERT-large:**

i 24 encoder layers

# Model Architecture

- **BERT-large:**

i 24 encoder layers

ii 16 attention heads

# Model Architecture

- **BERT-large:**

i 24 encoder layers

ii 16 attention heads

iii 4096 hidden size of feed-forward layer

# Model Architecture

- **BERT-large:**

i 24 encoder layers

ii 16 attention heads

iii 4096 hidden size of feed-forward layer

iv 1024 embedding size

# BERT Training

Two ways of training;

1. Pre-training

# BERT Training

Two ways of training;

1. Pre-training
2. Fine-tuning

# BERT Training

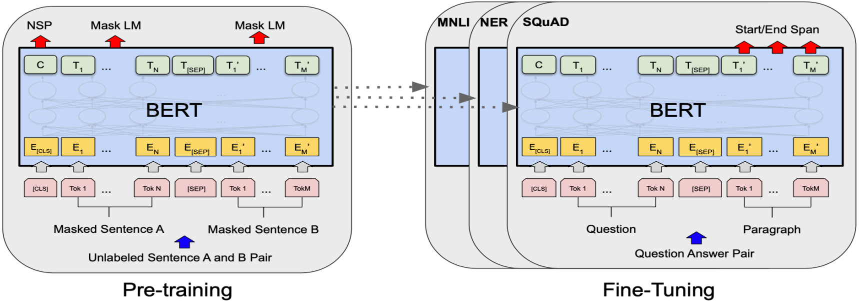



Figure 16: BERT Training

# BERT: Pre-training tasks

## Masked Language Modeling (MLM)

- Randomly masks 15% of the tokens.

# BERT: Pre-training tasks

## Masked Language Modeling (MLM)

- Randomly masks 15% of the tokens.
- Predicts the masked tokens using bidirectional context.

# Masked Language Modeling (MLM)

**SENTENCE: The cat sat on the mat**

**80% of the time [MASK] token —-> The [MASK] sat on the mat.**

**10% of the time random token —-> The ant sat on the mat.**

**10% of the time unchanged token —-> The cat sat on the mat.**

Figure 17: BERT: MLM Pre-training

# Masked Language Modeling (MLM)

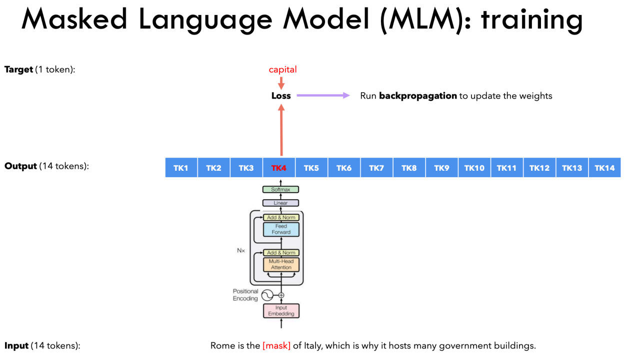



Figure 18: BERT: MLM Pre-training

# BERT: Pre-training tasks

## Next Sentence Prediction (NSP)

- Determines if a second sentence follows logically from the first.

# BERT: Pre-training tasks

## Next Sentence Prediction (NSP)

- Determines if a second sentence follows logically from the first.
- Sentence pair tasks; (sentence A, sentence B)

# BERT: Pre-training tasks

## Next Sentence Prediction (NSP)

- Determines if a second sentence follows logically from the first.
- Sentence pair tasks; (sentence A, sentence B)
- 50% of the B is the actual sentence, 50% is a random sentence.

# BERT: Pre-training tasks

## Next Sentence Prediction (NSP)

- Determines if a second sentence follows logically from the first.
- Sentence pair tasks; (sentence A, sentence B)
- 50% of the B is the actual sentence, 50% is a random sentence.
- Classification setup; *IsNEXT, NotNEXT*

# Next Sentence Prediction (NSP)

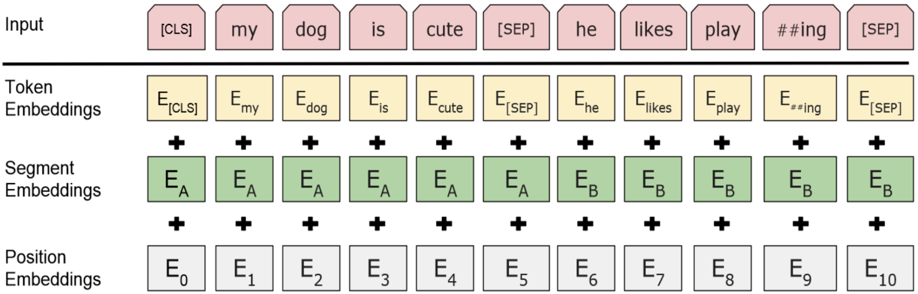



Figure 19: BERT: NSP Pre-training

# Next Sentence Prediction (NSP)

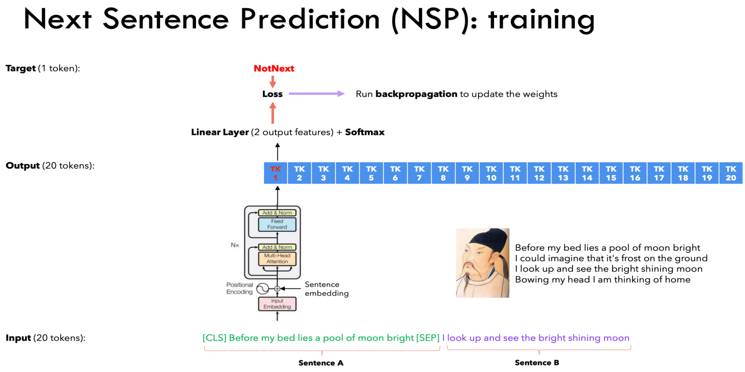



Figure 20: BERT: NSP Pre-training

# BERT: Fine-tuning tasks

## Text Classification Task

### Text Classification: training

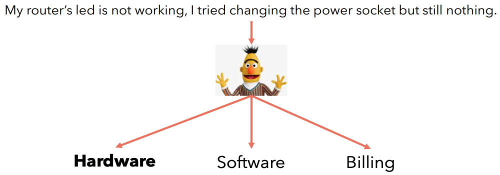



Figure 21: BERT: Fine-tuning for classification task

# Text Classification Task

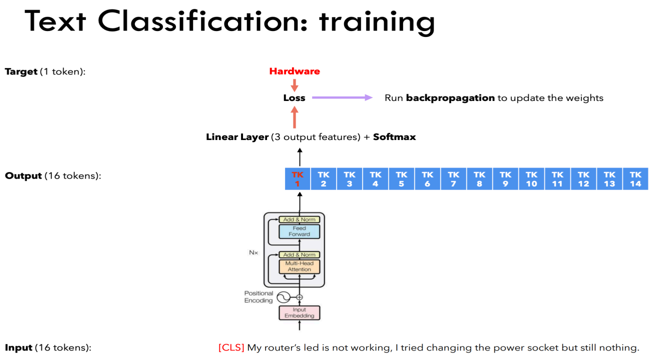



Figure 22: BERT: Fine-tuning for classification task

# BERT: Fine-tuning tasks

## Question Answering Task

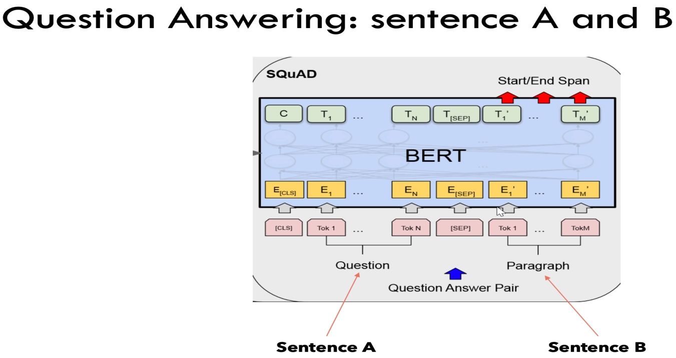



Figure 23: BERT: Fine-tuning for question answering task

# Question Answering Task

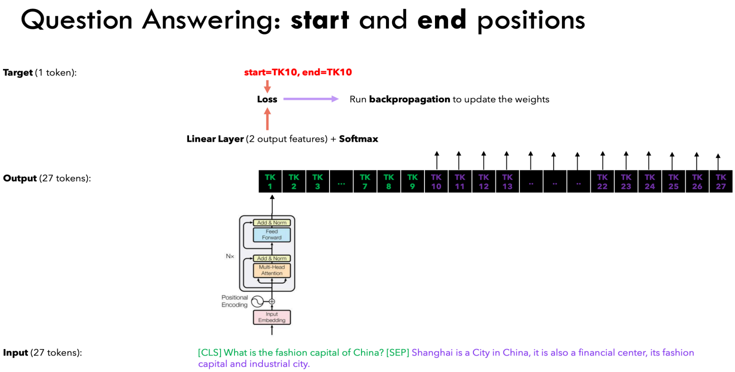



Figure 24: BERT: Fine-tuning for question answering task

# Experiment & Results

| System | MNLI-(m/mm) | QQP | QNLI | SST-2 | CoLA | STS-B | MRPC | RTE | **Average** |
|---|---|---|---|---|---|---|---|---|---|
| | 392k | 363k | 108k | 67k | 8.5k | 5.7k | 3.5k | 2.5k | - |
| Pre-OpenAI SOTA | 80.6/80.1 | 66.1 | 82.3 | 93.2 | 35.0 | 81.0 | 86.0 | 61.7 | 74.0 |
| BiLSTM+ELMo+Attn | 76.4/76.1 | 64.8 | 79.8 | 90.4 | 36.0 | 73.3 | 84.9 | 56.8 | 71.0 |
| OpenAI GPT | 82.1/81.4 | 70.3 | 87.4 | 91.3 | 45.4 | 80.0 | 82.3 | 56.0 | 75.1 |
| $\text{BERT}_{\text{BASE}}$ | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| $\text{BERT}_{\text{LARGE}}$ | **86.7/85.9** | **72.1** | **92.7** | **94.9** | **60.5** | **86.5** | **89.3** | **70.1** | **82.1** |

Figure 25: Experiment and results

# Limitations

- Computationally expensive.

# Limitations

- Computationally expensive.
- Requires large datasets.

# Limitations

- Computationally expensive.
- Requires large datasets.
- Limited understanding of rare or out-of-distribution words

# Conclusions

- BERT revolutionized NLP with bidirectional transformers.

# Conclusions

- BERT revolutionized NLP with bidirectional transformers.
- Future improvements focus on efficiency (e.g., DistilBERT) and domain-specific adaptations (e.g., BioBERT).

*Thank You*