BERT 101 - State Of The Art NLP Model Explained (huggingface.co)

# BERT 101 🤗 State Of The Art NLP Model Explained

Published March 2, 2022Update on GitHub

britneymullerBritney Muller

## What is BERT?

BERT, short for Bidirectional Encoder Representations from Transformers, is a Machine Learning (ML) model for natural language processing. It was developed in 2018 by researchers at Google AI Language and serves as a swiss army knife solution to 11+ of the most common language tasks, such as sentiment analysis and named entity recognition.

Language has historically been difficult for computers to 'understand'. Sure, computers can collect, store, and read text inputs but they lack basic language *context*.

So, along came Natural Language Processing (NLP): the field of artificial intelligence aiming for computers to read, analyze, interpret and derive meaning from text and spoken words. This practice combines linguistics, statistics, and Machine Learning to assist computers in 'understanding' human language.

Individual NLP tasks have traditionally been solved by individual models created for each specific task. That is, until— BERT!

BERT revolutionized the NLP space by solving for 11+ of the most common NLP tasks (and better than previous models) making it the jack of all NLP trades.

In this guide, you'll learn what BERT is, why it's different, and how to get started using BERT:

Let's get started! 🚀

# 1. What is BERT used for?

BERT can be used on a wide variety of language tasks:

- Can determine how positive or negative a movie's reviews are. (Sentiment Analysis)
- Helps chatbots answer your questions. (Question answering)
- Predicts your text when writing an email (Gmail). (Text prediction)
- Can write an article about any topic with just a few sentence inputs. (Text generation)
- Can quickly summarize long legal contracts. (Summarization)
- Can differentiate words that have multiple meanings (like 'bank') based on the surrounding text. (Polysemy resolution)

**There are many more language/NLP tasks + more detail behind each of these.**

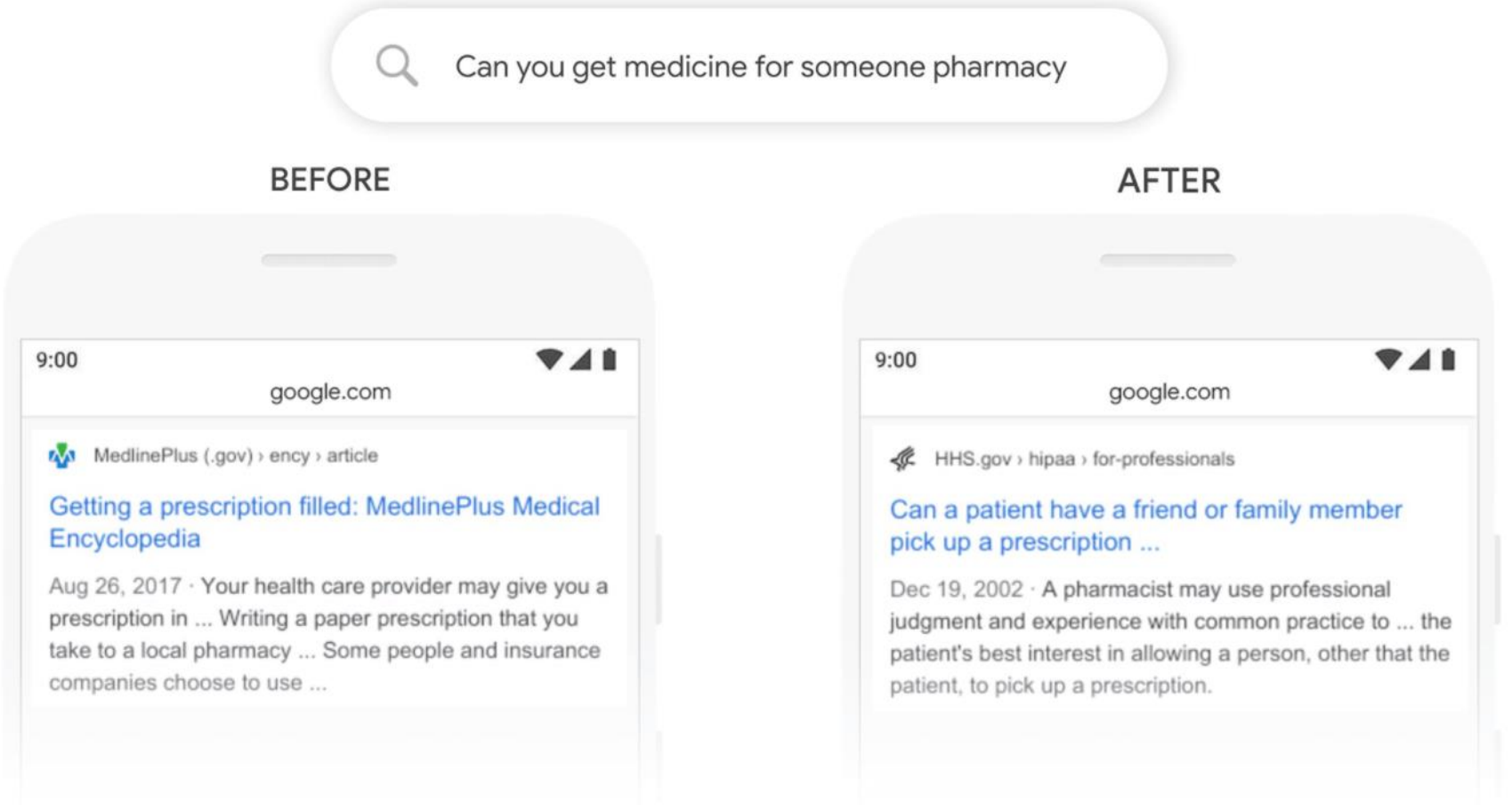
***Fun Fact:*** You interact with NLP (and likely BERT) almost every single day!

NLP is behind Google Translate, voice assistants (Alexa, Siri, etc.), chatbots, Google searches, voice-operated GPS, and more.

## 1.1 Example of BERT

BERT helps Google better surface (English) results for nearly all searches since November of 2020.

Here's an example of how BERT helps Google better understand specific searches like:




Source

Pre-BERT Google surfaced information about getting a prescription filled.

Post-BERT Google understands that “for someone” relates to picking up a prescription for someone else and the search results now help to answer that.

# 2. How does BERT Work?

BERT works by leveraging the following:

## 2.1 Large amounts of training data

A massive dataset of 3.3 Billion words has contributed to BERT's continued success.

BERT was specifically trained on Wikipedia (~2.5B words) and Google's BooksCorpus (~800M words). These large informational datasets contributed to BERT's deep knowledge not only of the English language but also of our world! 🚀

Training on a dataset this large takes a long time. BERT's training was made possible thanks to the novel Transformer architecture and sped up by using TPUs (Tensor Processing Units - Google's custom circuit built specifically for large ML models). —64 TPUs trained BERT over the course of 4 days.

**Note:** Demand for smaller BERT models is increasing in order to use BERT within smaller computational environments (like cell phones and personal computers). 23 smaller BERT models were released in March 2020. DistilBERT offers a lighter version of BERT; runs 60% faster while maintaining over 95% of BERT's performance.

## 2.2 What is a Masked Language Model?

MLM enables/enforces bidirectional learning from text by masking (hiding) a word in a sentence and forcing BERT to bidirectionally use the words on either side of the covered word to predict the masked word. This had never been done before!

**Fun Fact:** We naturally do this as humans!

### Masked Language Model Example:

Imagine your friend calls you while camping in Glacier National Park and their service begins to cut out. The last thing you hear before the call drops is:

Friend: "Dang! I'm out fishing and a huge trout just [blank] my line!"

Can you guess what your friend said??

You're naturally able to predict the missing word by considering the words bidirectionally before and after the missing word as context clues (in addition to your historical knowledge of how fishing works). Did you guess that your friend said, 'broke'? That's what we predicted as well but even we humans are error-prone to some of these methods.

**Note:** This is why you'll often see a "Human Performance" comparison to a language model's performance scores. And yes, newer models like BERT can be more accurate than humans!

The bidirectional methodology you did to fill in the [blank] word above is similar to how BERT attains state-of-the-art accuracy. A random 15% of tokenized words are hidden during training and BERT's job is to correctly predict the hidden words. Thus, directly teaching the model about the English language (and the words we use). Isn't that neat?

Play around with BERT's masking predictions:

**Hosted inference API**

Fill-Mask

Examples
Mask token: `[MASK]`
The goal of life is [MASK].Compute
The model is loaded and running on Intel Xeon 3rd Gen Scalable CPU
JSON OutputMaximize

**Fun Fact:** Masking has been around a long time - 1953 Paper on Cloze procedure (or 'Masking').

### 2.3 What is Next Sentence Prediction?

NSP (Next Sentence Prediction) is used to help BERT learn about relationships between sentences by predicting if a given sentence follows the previous sentence or not.

**Next Sentence Prediction Example:**

1. Paul went shopping. He bought a new shirt. (correct sentence pair)

2. Ramona made coffee. Vanilla ice cream cones for sale. (incorrect sentence pair)

In training, 50% correct sentence pairs are mixed in with 50% random sentence pairs to help BERT increase next sentence prediction accuracy.

**Fun Fact:** BERT is trained on both MLM (50%) and NSP (50%) at the same time.

### 2.4 Transformers

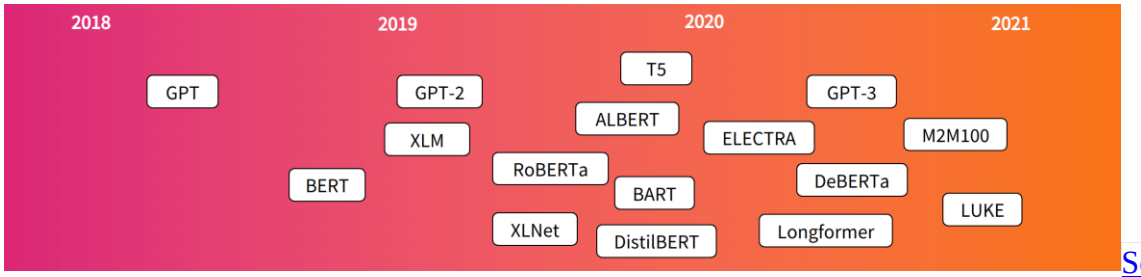
The Transformer architecture makes it possible to parallelize ML training extremely efficiently. Massive parallelization thus makes it feasible to train BERT on large amounts of data in a relatively short period of time.

Transformers use an attention mechanism to observe relationships between words. A concept originally proposed in the popular 2017 Attention Is All You Need paper sparked the use of Transformers in NLP models all around the world.

*Since their introduction in 2017, Transformers have rapidly become the state-of-the-art approach to tackle tasks in many domains such as natural language processing, speech recognition, and computer vision. In short, if you're doing deep learning, then you need Transformers!*

Lewis Tunstall, Hugging Face ML Engineer & Author of Natural Language Processing with Transformers

Timeline of popular Transformer model releases:



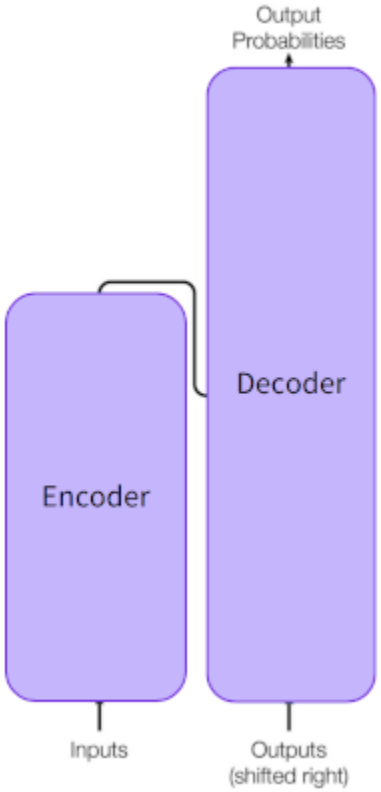
Source

#### 2.4.1 How do Transformers work?

Transformers work by leveraging attention, a powerful deep-learning algorithm, first seen in computer vision models.

—Not all that different from how we humans process information through attention. We are incredibly good at forgetting/ignoring mundane daily inputs that don't pose a threat or require a response from us. For example, can you remember everything you saw and heard coming home last Tuesday? Of course not! Our brain's memory is limited and valuable. Our recall is aided by our ability to forget trivial inputs.

Similarly, Machine Learning models need to learn how to pay attention only to the things that matter and not waste computational resources processing irrelevant information. Transformers create differential weights signaling which words in a sentence are the most critical to further process.



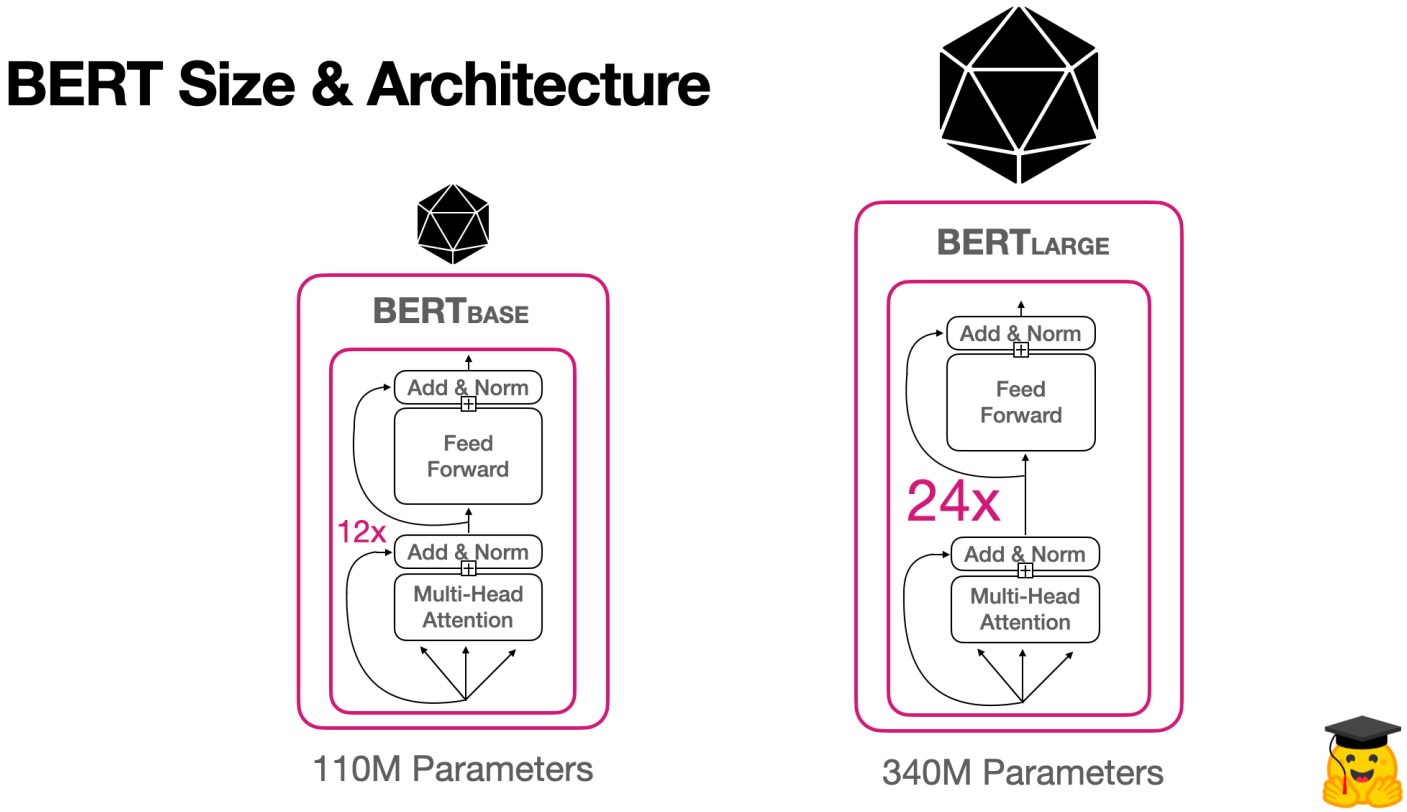

A transformer does this by successively processing an input through a stack of transformer layers, usually called the encoder. If necessary, another stack of transformer layers - the decoder - can be used to predict a target output. —BERT however, doesn't use a decoder. Transformers are uniquely suited for unsupervised learning because they can efficiently process millions of data points.

Fun Fact: Google has been using your reCAPTCHA selections to label training data since 2011. The entire Google Books archive and 13 million articles from the New York Times catalog have been transcribed/digitized via people entering reCAPTCHA text. Now, reCAPTCHA is asking us to label Google Street View images, vehicles, stoplights, airplanes, etc. Would be neat if Google made us aware of our participation in this effort (as the training data likely has future commercial intent) but I digress..

To learn more about Transformers check out our Hugging Face Transformers Course.

## 3. BERT model size & architecture

Let's break down the architecture for the two original BERT models:




ML Architecture Glossary:

| ML Architecture Parts | Definition |
| --- | --- |
| Parameters: | Number of learnable variables/values available for the model. |
| Transformer Layers: | Number of Transformer blocks. A transformer block transforms a sequence of word representations to a sequence of contextualized words (numbered representations). |
| Hidden Size: | Layers of mathematical functions, located between the input and output, that assign weights (to words) to produce a desired result. |
| Attention Heads: | The size of a Transformer block. |
| Processing: | Type of processing unit used to train the model. |

| ML Architecture Parts | Definition |
|---|---|
| Length of Training: | Time it took to train the model. |

Here’s how many of the above ML architecture parts BERTbase and BERTlarge has:

| | Transformer Layers | Hidden Size | Attention Heads | Parameters | Processing | Length of Training |
|---|---|---|---|---|---|---|
| BERTbase | 12 | 768 | 12 | 110M | 4 TPUs | 4 days |
| BERTlarge | 24 | 1024 | 16 | 340M | 16 TPUs | 4 days |

Let’s take a look at how BERTlarge’s additional layers, attention heads, and parameters have increased its performance across NLP tasks.

## 4. BERT's performance on common language tasks

BERT has successfully achieved state-of-the-art accuracy on 11 common NLP tasks, outperforming previous top NLP models, and is the first to outperform humans! But, how are these achievements measured?

## NLP Evaluation Methods:

### 4.1 SQuAD v1.1 & v2.0

SQuAD (Stanford Question Answering Dataset) is a reading comprehension dataset of around 108k questions that can be answered via a corresponding paragraph of Wikipedia text. BERT's performance on this evaluation method was a big achievement beating previous state-of-the-art models and human-level 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 |

### 4.2 SWAG

SWAG (Situations With Adversarial Generations) is an interesting evaluation in that it detects a model's ability to infer commonsense! It does this through a large-scale dataset of 113k multiple choice questions about common sense situations. These

questions are transcribed from a video scene/situation and SWAG provides the model with four possible outcomes in the next scene. The model then does its' best at predicting the correct answer.

BERT out outperformed top previous top models including human-level 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 |

### 4.3 GLUE Benchmark

GLUE (General Language Understanding Evaluation) benchmark is a group of resources for training, measuring, and analyzing language models comparatively to one another. These resources consist of nine "difficult" tasks designed to test an NLP model's understanding. Here's a summary of each of those tasks:

# GLUE (General Language Understanding Evaluation) Benchmark Tasks:

| Task | Example | Dataset | Metric |
|---|---|---|---|
| **Grammatical** | "This toast is than that one."<br>**= Ungrammatical** | CoLA | Matthews |
| **Sentiment Analysis** | "Toy Story 2 was okay."<br>**= .543291 (neutral)** | SST-2 | Accuracy |
| **Similarity** | a.) A pride of lions surrounded a monkey.<br>b.) Lions encompassed a monkey.<br>**= 4.7 (Very Similar)** | STS-B | Person / Spearman |
| **Paraphrase** | A. Last week, Seattle reported 12 new earthquakes.<br>B. Seattle reported another 12 earthquakes yesterday.<br>**= A Paraphrase** | MRPC | Accuracy / F1 |
| **Question Similarity** | a.) How can I cook noodles over a campfire?<br>b.) How do you make Mac & Cheese?<br>**= Not Similar** | QQP | Accuracy / F1 |
| **Contradiction** | a.) Glossier products are the best!<br>b.) Glossier products are overpriced.<br>**= Contradiction** | MNLI-mm | Accuracy |
| **Answerable** | a.) How does the Dyson Airwrap work?<br>b.) The Airwarp uses the Coanda effect to create a vortex pulling the hair towards the attachments.<br>**= Answerable** | QNLI | Accuracy |
| **Entail** | a.) In 2006, Paul David bought a Microprocessing center to create 30,000 jobs in Northern Minnesota.<br>b.) Paul David created 30,000 jobs in MN.<br>**= Entail** | RTE | Accuracy |
| **Ambiguous pronouns** | a.) Federico spoke to Marie, breaking her focus.<br>b.) Federico spoke to Marie, breaking Federico's focus.<br>**= Incorrect Referent** | WNLI | Accuracy |

## 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 |

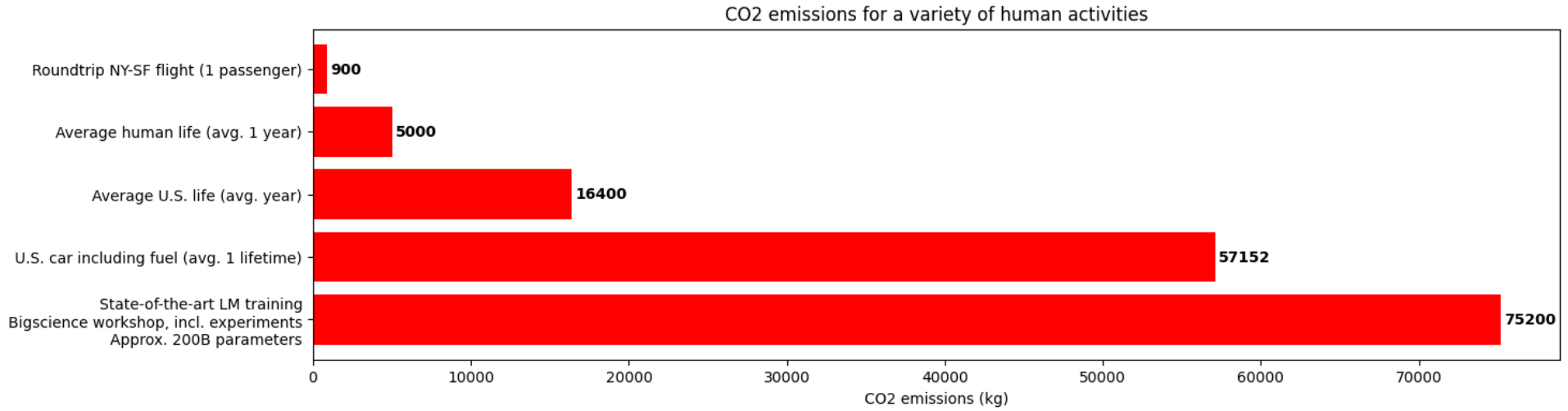
While some of these tasks may seem irrelevant and banal, it's important to note that these evaluation methods are *incredibly* powerful in indicating which models are best suited for your next NLP application.

Attaining performance of this caliber isn't without consequences. Next up, let's learn about Machine Learning's impact on the environment.

## 5. Environmental impact of deep learning

Large Machine Learning models require massive amounts of data which is expensive in both time and compute resources.

These models also have an environmental impact:




Source

Machine Learning's environmental impact is one of the many reasons we believe in democratizing the world of Machine Learning through open source! Sharing large pre-trained language models is essential in reducing the overall compute cost and carbon footprint of our community-driven efforts.

## 6. The open source power of BERT

Unlike other large learning models like GPT-3, BERT's source code is publicly accessible (view BERT's code on Github) allowing BERT to be more widely used all around the world. This is a game-changer!

Developers are now able to get a state-of-the-art model like BERT up and running quickly without spending large amounts of time and money.

Developers can instead focus their efforts on fine-tuning BERT to customize the model's performance to their unique tasks.

It's important to note that 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.

BERT models pre-trained for specific tasks:

- Twitter sentiment analysis
- Analysis of Japanese text
- Emotion categorizer (English - anger, fear, joy, etc.)
- Clinical Notes analysis
- Speech to text translation
- Toxic comment detection

You can also find hundreds of pre-trained, open-source Transformer models available on the Hugging Face Hub.

# 7. How to get started using BERT

We've created this notebook so you can try BERT through this easy tutorial in Google Colab. Open the notebook or add the following code to your own. Pro Tip: Use (Shift + Click) to run a code cell.

Note: Hugging Face's pipeline class makes it incredibly easy to pull in open source ML models like transformers with just a single line of code.

### 7.1 Install Transformers

First, let's install Transformers via the following code:

```
!pip install transformers
```

### 7.2 Try out BERT

Feel free to swap out the sentence below for one of your own. However, leave [MASK] in somewhere to allow BERT to predict the missing word

```
from transformers import pipeline

unmasker = pipeline('fill-mask', model='bert-base-uncased')

unmasker("Artificial Intelligence [MASK] take over the world.")
```

When you run the above code you should see an output like this:

```
[{'score': 0.3182411789894104,

 'sequence': 'artificial intelligence can take over the world.',

 'token': 2064,

 'token_str': 'can'},

{'score': 0.18299679458141327,

 'sequence': 'artificial intelligence will take over the world.',

 'token': 2097,

 'token_str': 'will'},

{'score': 0.05600147321820259,

 'sequence': 'artificial intelligence to take over the world.',
```

```
 'token': 2000,

 'token_str': 'to'},

{'score': 0.04519503191113472,

 'sequence': 'artificial intelligences take over the world.',

 'token': 2015,

 'token_str': '##s'},

{'score': 0.045153118669986725,

 'sequence': 'artificial intelligence would take over the world.',

 'token': 2052,

 'token_str': 'would'}]
```

Kind of frightening right? 🙃

### 7.3 Be aware of model bias

Let's see what jobs BERT suggests for a "man":

```
unmasker("The man worked as a [MASK].")
```

When you run the above code you should see an output that looks something like:


```
[{'score': 0.09747546911239624,
 'sequence': 'the man worked as a carpenter.',
 'token': 10533,
 'token_str': 'carpenter'},
{'score': 0.052383411675691605,
 'sequence': 'the man worked as a waiter.',
 'token': 15610,
 'token_str': 'waiter'},
{'score': 0.04962698742747307,
 'sequence': 'the man worked as a barber.',
 'token': 13362,
 'token_str': 'barber'},
{'score': 0.037886083126068115,
 'sequence': 'the man worked as a mechanic.',
```

```
 'token': 15893,

 'token_str': 'mechanic'},

{'score': 0.037680838257074356,

 'sequence': 'the man worked as a salesman.',

 'token': 18968,

 'token_str': 'salesman'}]
```

BERT predicted the man's job to be a Carpenter, Waiter, Barber, Mechanic, or Salesman

Now let's see what jobs BERT suggesst for "woman"

```
unmasker("The woman worked as a [MASK].")
```

You should see an output that looks something like:

```
[{'score': 0.21981535851955414,

 'sequence': 'the woman worked as a nurse.',

 'token': 6821,

 'token_str': 'nurse'},
```

{'score': 0.1597413569688797,
'sequence': 'the woman worked as a waitress.',
'token': 13877,
'token_str': 'waitress'},
{'score': 0.11547300964593887,
'sequence': 'the woman worked as a maid.',
'token': 10850,
'token_str': 'maid'},
{'score': 0.037968792021274 57,
'sequence': 'the woman worked as a prostitute.',
'token': 19215,
'token_str': 'prostitute'},
{'score': 0.030423851683735847,
'sequence': 'the woman worked as a cook.',
'token': 5660,

```
 'token_str': 'cook'}]
```

BERT predicted the woman's job to be a Nurse, Waitress, Maid, Prostitute, or Cook displaying a clear gender bias in professional roles.

### 7.4 Some other BERT Notebooks you might enjoy:

A Visual Notebook to BERT for the First Time

Train your tokenizer

+Don't forget to checkout the Hugging Face Transformers Course to learn more 🍾

## 8. BERT FAQs

### Can BERT be used with PyTorch?

Yes! Our experts at Hugging Face have open-sourced the PyTorch transformers repository on GitHub.

Pro Tip: Lewis Tunstall, Leandro von Werra, and Thomas Wolf also wrote a book to help people build language applications with Hugging Face called, 'Natural Language Processing with Transformers'.

### Can BERT be used with Tensorflow?

Yes! You can use Tensorflow as the backend of Transformers.

### How long does it take to pre-train BERT?

The 2 original BERT models were trained on 4(BERTbase) and 16(BERTlarge) Cloud TPUs for 4 days.

### How long does it take to fine-tune BERT?

For common NLP tasks discussed above, BERT takes between 1-25mins on a single Cloud TPU or between 1-130mins on a single GPU.

**What makes BERT different?**

BERT was one of the first models in NLP that was trained in a two-step way:

1. 1. BERT was trained on massive amounts of unlabeled data (no human annotation) in an unsupervised fashion.
2. 2. BERT was then trained on small amounts of human-annotated data starting from the previous pre-trained model resulting in state-of-the-art performance.

## 9. Conclusion

BERT is a highly complex and advanced language model that helps people automate language understanding. Its ability to accomplish state-of-the-art performance is supported by training on massive amounts of data and leveraging Transformers architecture to revolutionize the field of NLP.

Thanks to BERT's open-source library, and the incredible AI community's efforts to continue to improve and share new BERT models, the future of untouched NLP milestones looks bright.

What will you create with BERT?

Learn how to fine-tune BERT for your particular use case 🤗