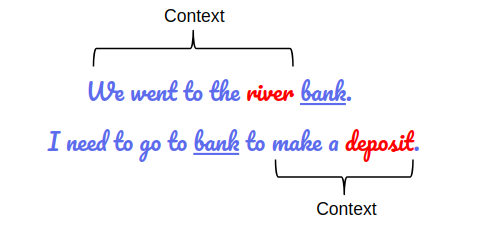
BERT

**What is BERT?**

* **BERT (Bidirectional Encoder Representations from Transformers) is an open-sourced NLP pre-training model developed by researchers at Google in 2018. A direct descendant to GPT (Generalized Language Models), BERT has outperformed several models in NLP and provided top results in Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and other frameworks.**
* **“BERT stands for**B**idirectional**E**ncoder**R**epresentations from**T**ransformers. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.”**
* **First, it’s easy to get that BERT stands for**B**idirectional**E**ncoder**R**epresentations from**T**ransformers. Each word here has a meaning to it and we will encounter that one by one in this article. For now, the key takeaway from this line is –**BERT is based on the Transformer architecture.
* **Second, BERT is pre-trained on a large corpus of unlabelled text including the entire Wikipedia(that’s 2,500 million words!) and Book Corpus (800 million words).**
* **This pre-training step is half the magic behind BERT’s success. This is because as we train a model on a large text corpus, our model starts to pick up the deeper and intimate understandings of how the language works. This knowledge is the swiss army knife that is useful for almost any NLP task.**
* **Third, BERT is a “deeply bidirectional” model. Bidirectional means that BERT learns information from both the left and the right side of a token’s context during the training phase.**
* **The bidirectionality of a model is important for truly understanding the meaning of a language. Let’s see an example to illustrate this. There are two sentences in this example and both of them involve the word “bank”:**

****

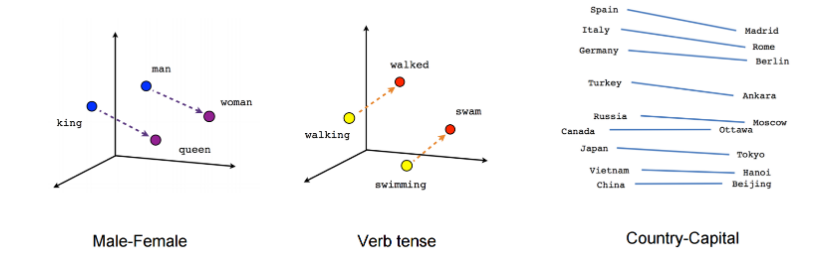
* **What makes it’s unique from the rest of the model is that it’s is the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus. Since it’s open-sourced, anyone with machine learning knowledge can easily build an NLP model without the need for sourcing massive datasets for training the model thus saving time, energy, knowledge and resources.**
* **Finally, BERT is pre-trained on a large corpus of unlabelled text which includes the entire Wikipedia (that’s about 2,500 million words) and a book corpus (800 million words).**

## From Word2Vec to BERT: NLP’s Quest for Learning Language Representations

**“One of the biggest challenges in natural language processing is the shortage of training data. Because NLP is a diversified field with many distinct tasks, most task-specific datasets contain only a few thousand or a few hundred thousand human-labelled training examples.” – Google AI**

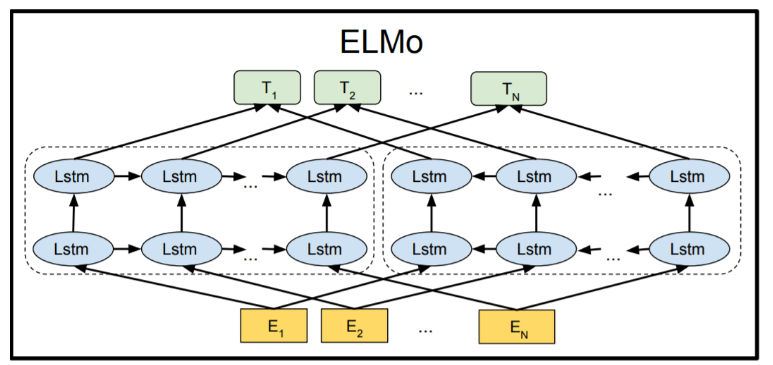
### Word2Vec and GloVe

**The quest for learning language representations by pre-training models on large unlabelled text data started from word embeddings like**[**Word2Vec and GloVe**](https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/?utm_source=blog&utm_medium=demystifying-bert-groundbreaking-nlp-framework)**. These embeddings changed the way we performed NLP tasks. We now had embeddings that could capture contextual relationships among words.**

****

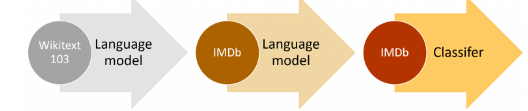
* **These embeddings were used to train models on downstream NLP tasks and make better predictions. This could be done even with less task-specific data by utilizing the additional information from the embeddings itself.**
* One limitation of these embeddings was the use of very shallow [Language Models](https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-language-model-nlp-python-code/?utm_source=blog&utm_medium=demystifying-bert-groundbreaking-nlp-framework).**This meant there was a limit to the amount of information they could capture and this motivated the use of deeper and more complex language models (layers of**[**LSTMs**](https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/?utm_source=blog&utm_medium=demystifying-bert-groundbreaking-nlp-framework)**and GRUs).**
* Another key limitation was that these models did not take the context of the word into account. **Let’s take the above “bank” example. The same word has different meanings in different contexts, right? However, an embedding like Word2Vec will give the same vector for “bank” in both the contexts.**

### Enter ELMO and ULMFiT

****

[**ELMo**](https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/?utm_source=blog&utm_medium=demystifying-bert-groundbreaking-nlp-framework)**was the NLP community’s response to the problem of**Polysemy **– same words having different meanings based on their context. From training shallow feed-forward networks (Word2vec), we graduated to training word embeddings using layers of complex Bi-directional LSTM architectures. This meant that the same word can have multiple ELMO embeddings based on the context it is in.**

**That’s when we started seeing the advantage of**pre-training **as a training mechanism for NLP.**

****

[**ULMFiT**](https://www.analyticsvidhya.com/blog/2018/11/tutorial-text-classification-ulmfit-fastai-library/)**took this a step further. This framework could train language models that could be fine-tuned to provide excellent results even with fewer data (less than 100 examples) on a variety of document classification tasks. It is safe to say that ULMFiT cracked the code to**[**transfer learning**](https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/?utm_source=blog&utm_medium=demystifying-bert-groundbreaking-nlp-framework)**in NLP.**

**Transfer Learning in NLP = Pre-Training and Fine-Tuning**

**Most of the NLP breakthroughs that followed ULMFIT tweaked components of the above equation and gained state-of-the-art benchmarks.**

### OpenAI’s GPT

[**OpenAI’s GPT**](https://www.analyticsvidhya.com/blog/2019/07/openai-gpt2-text-generator-python/?utm_source=blog&utm_medium=demystifying-bert-groundbreaking-nlp-framework)**extended the methods of pre-training and fine-tuning that were introduced by ULMFiT and ELMo. GPT essentially replaced the LSTM-based architecture for Language Modeling with a Transformer-based architecture.**

**The GPT model could be fine-tuned to multiple NLP tasks beyond document classification, such as common sense reasoning, semantic similarity, and reading comprehension.**

**GPT also emphasized the importance of the Transformer framework, which has a simpler architecture and can train faster than an LSTM-based model. It is also able to learn complex patterns in the data by using the Attention mechanism.**

**OpenAI’s GPT validated the robustness and usefulness of the Transformer architecture by achieving multiple State-of-the-Arts.**

### Moving onto BERT

**So, the new approach to solving NLP tasks became a 2-step process:**

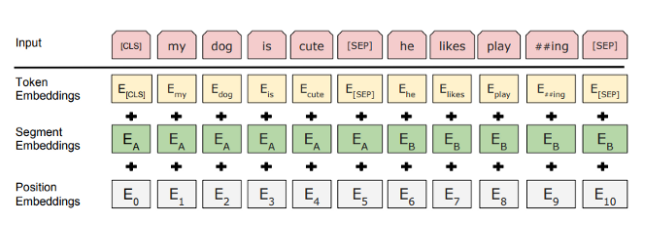
1. **Train a language model on a large unlabelled text corpus (unsupervised or semi-supervised)**
2. **Fine-tune this large model to specific NLP tasks to utilize the large repository of knowledge this model has gained (supervised)**

**With that context, let’s understand how BERT takes over from here to build a model that will become a benchmark of excellence in NLP for a long time.**

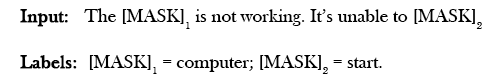
**How does it work?**

**Traditional context-free models (like word2vec or GloVe) generate a single word embedding representation for each word in the vocabulary which means the word**“right” **would have the same context-free representation in “*I’m sure I’m right*” and “*Take a right turn.*” However, BERT would represent based on both previous and next context making it bidirectional. While the concept of bidirectional was around for a long time, BERT was first on its kind to successfully pre-train bidirectional in a deep neural network.**

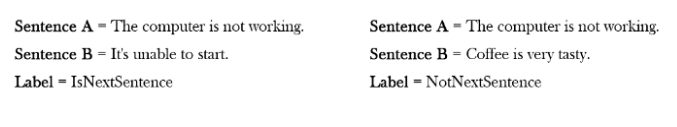
**How did they achieve this?**



**They use two strategies — Mask Language Model (MLM) — by Masking out some of the words in the input and then condition each word bidirectionally to predict the masked words. Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.**

****

**The second technique is the**Next Sentence Prediction (NSP)**, where BERT learns to model relationships between sentences. In the training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. Let’s consider two sentences A and B, is B the actual next sentence that comes after A in the corpus, or just a random sentence? For example:**



**When training the BERT model, both the techniques are trained together, thus minimizing the combined loss function of the two strategies.**

**The BERT architecture builds on top of Transformer. There are two variants available:**

**· BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million parameters**

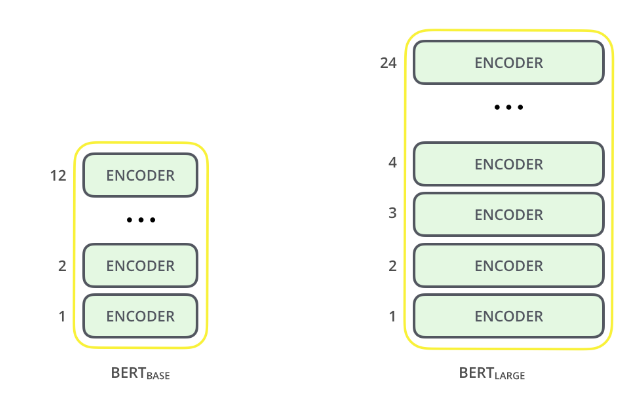
**· BERT Large: 24 layers (transformer blocks), 16 attention heads and, 340 million parameters**

## How Does BERT Work? A Look Under the Hood

### 1. BERT’s Architecture

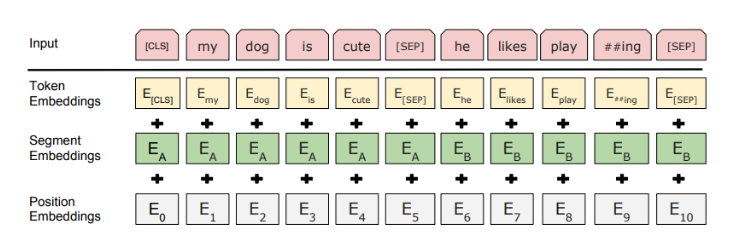
**The BERT architecture builds on top of Transformer. We currently have two variants available:**

* **BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million parameters**
* **BERT Large: 24 layers (transformer blocks), 16 attention heads and, 340 million parameters**

****

**The BERT Base architecture has the same model size as OpenAI’s GPT for comparison purposes. All of these Transformer layers are**Encoder**-only blocks.**

### 2. Text Preprocessing



**The developers behind BERT have added a specific set of rules to represent the input text for the model. Many of these are creative design choices that make the model even better.**

**For starters, every input embedding is a combination of 3 embeddings:**

1. **Position Embeddings: BERT learns and uses positional embeddings to express the position of words in a sentence. These are added to overcome the limitation of Transformer which, unlike an RNN, is not able to capture “sequence” or “order” information**
2. **Segment Embeddings: BERT can also take sentence pairs as inputs for tasks (Question-Answering). That’s why it learns a unique embedding for the first and the second sentences to help the model distinguish between them. In the above example, all the tokens marked as EA belong to sentence A (and similarly for EB)**
3. **Token Embeddings: These are the embeddings learned for the specific token from the WordPiece token vocabulary**

**For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings.**

**Such a comprehensive embedding scheme contains a lot of useful information for the model.**

**These combinations of preprocessing steps make BERT so versatile. This implies that without making any major change in the model’s architecture, we can easily train it on multiple kinds of NLP tasks.**

**3. Pre-training Tasks**

**BERT is pre-trained on two NLP tasks:**

**1.Masked Language Modeling**

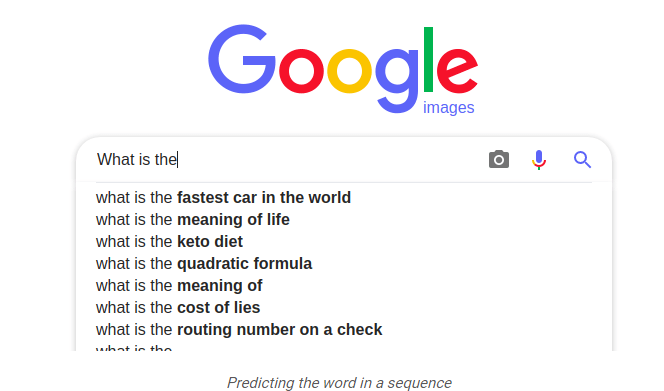
**2.Next Sentence Prediction**

### a. Masked Language Modeling (Bi-directionality)

Need for Bi-directionality

**BERT is designed as a**deeply bidirectional**model. The network effectively captures information from both the right and left context of a token from the first layer itself and all the way through to the last layer.**

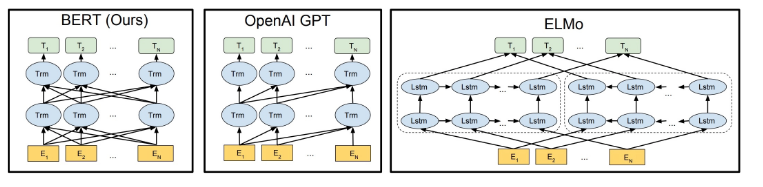
**Traditionally, we had language models either trained to predict the next word in a sentence (right-to-left context used in GPT) or language models that were trained on a left-to-right context. This made our models susceptible to errors due to loss in information.**

****

**ELMo tried to deal with this problem by training two LSTM language models on left-to-right and right-to-left contexts and shallowly concatenating them. Even though it greatly improved upon existing techniques, it wasn’t enough.**

**“Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-left model.” – BERT**

**That’s where BERT greatly improves upon both GPT and ELMo. Look at the below image:**

****

**The arrows indicate the information flow from one layer to the next. The green boxes at the top indicate the final contextualized representation of each input word.**

**It’s evident from the above image: BERT is bi-directional, GPT is unidirectional (information flows only from left-to-right), and ELMO is shallowly bidirectional.**

**This is where the**Masked Language Model**comes into the picture.**

**About Masked Language Models**

**Let’s say we have a sentence – “I love to read data science blogs on Analytics Vidhya”. We want to train a bi-directional language model. Instead of trying to predict the next word in the sequence, we can build a model to predict a missing word from within the sequence itself.**

**Let’s replace “Analytics” with “[MASK]”. This is a token to denote that the token is missing. We’ll then train the model in such a way that it should be able to predict “Analytics” as the missing token: “I love to read data science blogs on [MASK] Vidhya.”**

**This is the crux of a Masked Language Model. The authors of BERT also include some caveats to further improve this technique:**

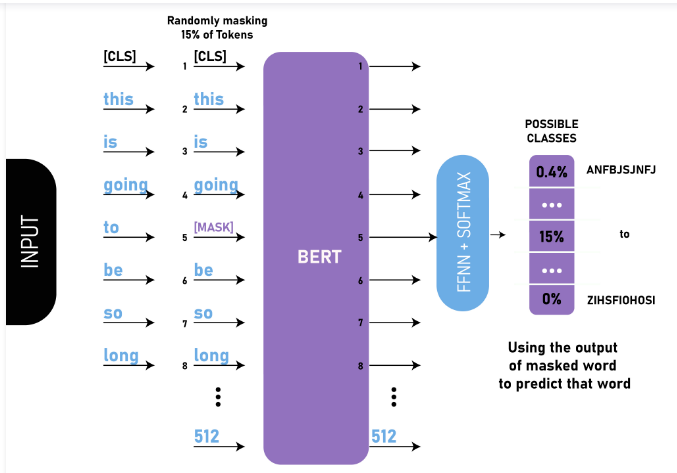
* **To prevent the model from focusing too much on a particular position or tokens that are masked, the researchers randomly masked 15% of the words**
* **The masked words were not always replaced by the masked tokens [MASK] because the [MASK] token would never appear during fine-tuning**
* **So, the researchers used the below technique:**

**80% of the time the words were replaced with the masked token [MASK]**

**10% of the time the words were replaced with random words**

**10% of the time the words were left unchanged**

Masked Language Model: **In this NLP task, we replace 15% of words in the text with the [MASK] token. The model then predicts the original words that are replaced by [MASK] token. Beyond masking, the masking also mixes things a bit in order to improve how the model later for fine-tuning because [MASK] token created a mismatch between training and fine-tuning. In this model, we add a classification layer at the top of the encoder input. We also calculate the probability of the output using a fully connected and a softmax layer.**

****

**Masked Language Model:  
The BERT loss function while calculating it considers only the prediction of masked values and ignores the prediction of the non-masked values. This helps in calculating loss for only those 15% masked words.**

### b. Next Sentence Prediction

**Masked Language Models (MLMs) learn to understand the relationship between words. Additionally,**BERT is also trained on the task of Next Sentence Prediction for tasks that require an understanding of the relationship between sentences.

**A good example of such a task would be**[**question answering systems**](https://www.analyticsvidhya.com/blog/2017/12/introduction-computational-linguistics-dependency-trees/?utm_source=blog&utm_medium=demystifying-bert-groundbreaking-nlp-framework)**.**

**The task is simple. Given two sentences – A and B, is B the actual next sentence that comes after A in the corpus, or just a random sentence?**

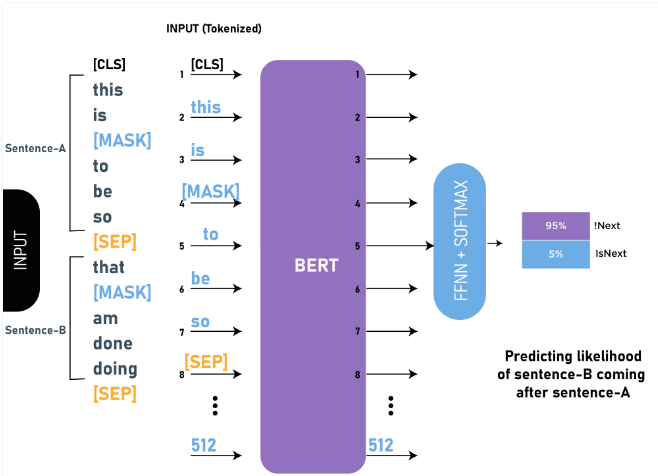
**Since it is a binary classification task, the data can be easily generated from any corpus by splitting it into sentence pairs. Just like MLMs, the authors have added some caveats here too. Let’s take this with an example:**

**Consider that we have a text dataset of 100,000 sentences. So, there will be 50,000 training examples or pairs of sentences as the training data.**

* **For 50% of the pairs, the second sentence would actually be the next sentence to the first sentence**
* **For the remaining 50% of the pairs, the second sentence would be a random sentence from the corpus**
* **The labels for the first case would be ‘IsNext’ and ‘NotNext’ for the second case**

**And this is how BERT is able to become a true task-agnostic model. It combines both the Masked Language Model (MLM) and the Next Sentence Prediction (NSP) pre-training tasks.**

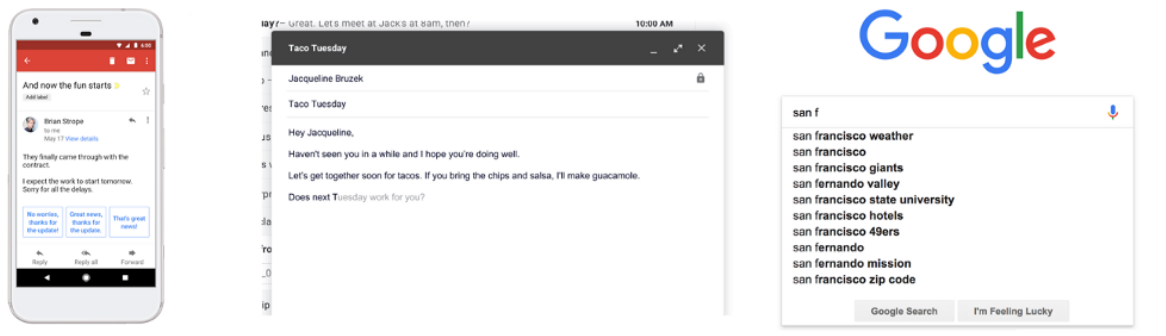
**In this NLP task, we are provided two sentences, our goal is to predict whether the second sentence is the next subsequent sentence of the first sentence in the original text. During training the BERT, we take 50% of the data that is the next subsequent sentence (labelled as isNext) from the original sentence and 50% of the time we take the random sentence that is not the next sentence in the original text (labelled as NotNext). Since this is a classification task so we the first token is the [CLS] token. This model also uses a [SEP] token to separate the two sentences that we passed into the model.**

****

**The BERT model obtained an accuracy of 97%-98% on this task. The advantage of training the model with the task is that it helps the model understand the relationship between sentences.**

**BERT is here — But is it ready for the real world?**

* **BERT is undoubtedly a milestone in the use of Machine Learning for Natural Language Processing. But we need to introspect on how BERT can be used in various NLP scenarios.**
* **Text Classification and Categorization has been one of the prime applications of NLP. E.g. the concept has been used in ticket tools to classify tickets based on the short description/email and categorize/route the ticket to the right team for resolution. Similarly, it can also be used to classify if an email is a spam or not.**
* **You can find some applications of it already used in your daily life.**

****

* **Chatbots are disrupting the messaging industry with its ability to answer user queries and handle a variety of tasks. However, one of the biggest limitations has been in intent recognition and capturing entities from sentences.**
* **Question Answering (QnA) model is one of the very basic systems of Natural Language Processing. In QnA, the Machine Learning based system generates answers from the knowledge base or text paragraphs for the questions posed as input.**Can BERT be used in a chatbot?**Certainly yes. BERT is now being utilized in many conversational AI applications. So, your chatbots should be getting smarter.**
* **However, BERT can be used only for answering questions from very short paragraphs and a lot of key issues need to be addressed. NLP as a general task is way too complex and has many more meanings and subtleties. BERT solves only a part of it but is certainly going to change entity Recognition models soon.**
* **BERT today can address only a limited class of problems. However, there are many other tasks such as sentiment detection, classification, machine translation, named entity recognition, summarization and question answering that need to build upon. A common criticism now is that such tasks are based on the manipulation of representations without any kind of understanding and adding simple adversarial content that modifies original content confuses it.**
* **The true benefits of BERT in NLP will only be realized when there is broader adoption in operations and improvement in live scenarios thus supporting a wide range of applications across organizations and users.**
* **However, things are changing quickly with a wave of transformer-based methods (GPT-2, RoBERTa, XLNet) that keeps raising the bar by demonstrating better performance or easier training or some other specific benefit.**

**Other developments which came after BERT’s**

## RoBERTa

**Developed by Facebook, RoBERTa is built on BERT’s language masking strategy and modifies some of the key hyperparameters in BERT. To improve the training procedure, RoBERTa removes the Next Sentence Prediction (NSP) task from BERT’s pre-training and introduces dynamic masking so that the masked token changes during the training epochs. It was also trained on an order of magnitude more data than BERT, for a longer amount of time.**

## DistilBERT

**Developed by HuggingFace, DistilBERT learns a distilled (approximate) version of BERT, retaining 95% performance on GLUE but using only half the number of parameters (only 66 million parameters, instead of 110 million). The concept is that once a large neural network has been trained, its full output distributions can be approximated using a smaller network (like posterior approximation).**

## XLM/mBERT

**Developed by Facebook, XLM uses a known pre-processing technique (BPE) and a dual-language training mechanism with BERT in order to learn relations between words in different languages. The model outperforms other models in a multi-lingual classification task and significantly improves machine translation when a pre-trained model is used for the initialization of the translation model.**

## ALBERT

**Jointly developed by Google Research and Toyota Technological Institute,**ALBERT (A Lite BERT for Self-Supervised Learning of Language Representations) **is primed to be the successor to BERT which is much smaller and lighter and smarter to BERT. Two key architecture changes allow ALBERT to both outperform and dramatically reduce the model size. The first one is the number of parameters. It improves parameter efficiency by sharing all parameters, across all layers. That means Feed Forward Network parameters and Attention parameters are all shared.**

**Researchers also isolated the size of the hidden layers from the size of vocabulary embeddings. This was done by projecting one-hot vectors into a lower-dimensional embedding space and then to the hidden space, which made it easier to increase the hidden layer size without significantly increasing the parameter size of the vocabulary embeddings.**

**When it comes to pre-train, ALBERT has it’s own training method called**Sentence-Order Prediction (SOP)**as opposed to NSP. The problem with NSP as theorized by the authors was that it conflates *topic* prediction with *coherence* prediction.**

**ALBERT represents a new state of the art for NLP on several benchmarks and a new state of the art for parameter efficiency. It’s an amazing breakthrough that builds on the great work done by BERT one year ago and advances NLP in multiple aspects.**

**BERT and models like it are certainly game-changers in NLP. Machines can now better understand speech and respond intelligently in real-time. Many BERT based models are being developed including VideoBERT, ViLBERT (Vision-and-Language BERT), PatentBERT, DocBERT, etc.**

**Implementing BERT for Text Classification in Python**

**Your mind must be whirling with the possibilities BERT has opened up. There are many ways we can take advantage of BERT’s large repository of knowledge for our NLP applications.**

**One of the most potent ways would be fine-tuning it on your own task and task-specific data. We can then use the embeddings from BERT as embeddings for our text documents.**

**In this section, we will learn how to use BERT’s embeddings for our NLP task. We’ll take up the concept of fine-tuning an entire BERT model in one of the future articles.**

**For extracting embeddings from BERT, we will use a really useful open source project called**[**Bert-as-Service**](https://github.com/hanxiao/bert-as-service)**:**

**Running BERT can be a painstaking process since it requires a lot of code and installing multiple packages. That’s why this open-source project is so helpful because it lets us** use BERT to extract encodings for each sentence in just two lines of code.

### Installing BERT-As-Service

**BERT-As-Service works in a simple way. It creates a BERT server which we can access using the Python code in our notebook. Every time we send it a sentence as a list, it will send the embeddings for all the sentences.**

**We can install the server and client via pip. They can be installed separately or even on *different* machines:**

**pip install bert-serving-server # server**

**pip install bert-serving-client # client, independent of `bert-serving-server`**

**Note that the server MUST be running on**Python >= 3.5**with**TensorFlow >= 1.10**(*one-point-ten*).**

Also, since running BERT is a GPU intensive task, I’d suggest installing the bert-serving-server on a cloud-based GPU or some other machine that has high compute capacity.

**Now, go back to your terminal and download a model listed below. Then, uncompress the zip file into some folder, say */tmp/english\_L-12\_H-768\_A-12/*.**

**Here’s a list of the released pre-trained BERT models:**

|  |  |
| --- | --- |
| [**BERT-Base, Uncased**](https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip) | **12-layer, 768-hidden, 12-heads, 110M parameters** |
| [**BERT-Large, Uncased**](https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-24_H-1024_A-16.zip) | **24-layer, 1024-hidden, 16-heads, 340M parameters** |
| [**BERT-Base, Cased**](https://storage.googleapis.com/bert_models/2018_10_18/cased_L-12_H-768_A-12.zip) | **12-layer, 768-hidden, 12-heads, 110M parameters** |
| [**BERT-Large, Cased**](https://storage.googleapis.com/bert_models/2018_10_18/cased_L-24_H-1024_A-16.zip) | **24-layer, 1024-hidden, 16-heads, 340M parameters** |
| [**BERT-Base, Multilingual Cased (New)**](https://storage.googleapis.com/bert_models/2018_11_23/multi_cased_L-12_H-768_A-12.zip) | **104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters** |
| [**BERT-Base, Multilingual Cased (Old)**](https://storage.googleapis.com/bert_models/2018_11_03/multilingual_L-12_H-768_A-12.zip) | **102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters** |
| [**BERT-Base, Chinese**](https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip) | **Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-heads, 110M parameters** |

**We’ll download BERT Uncased and then decompress the zip file:**

**wget https://storage.googleapis.com/bert\_models/2018\_10\_18/uncased\_L-12\_H-768\_A-12.zip && unzip uncased\_L-12\_H-768\_A-12.zip**

**Once we have all the files extracted in a folder, it’s time to start the BERT service:**

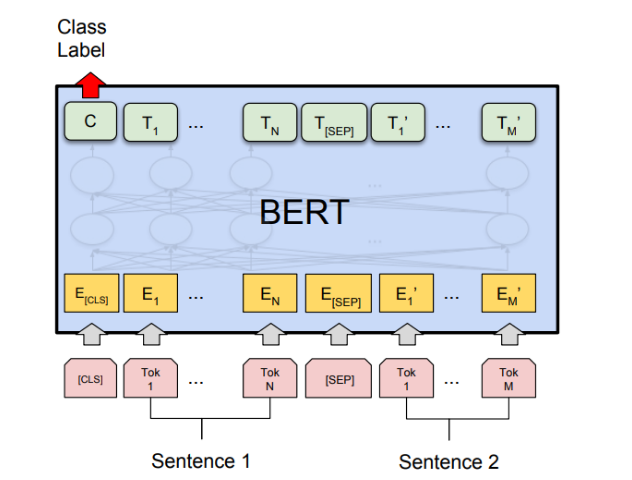
**bert-serving-start -model\_dir uncased\_L-12\_H-768\_A-12/ -num\_worker=2 -max\_seq\_len 50**

**You can now simply call the BERT-As-Service from your Python code (using the client library). Let’s just jump into code!**

**Fine Tune BERT for Different Tasks –**

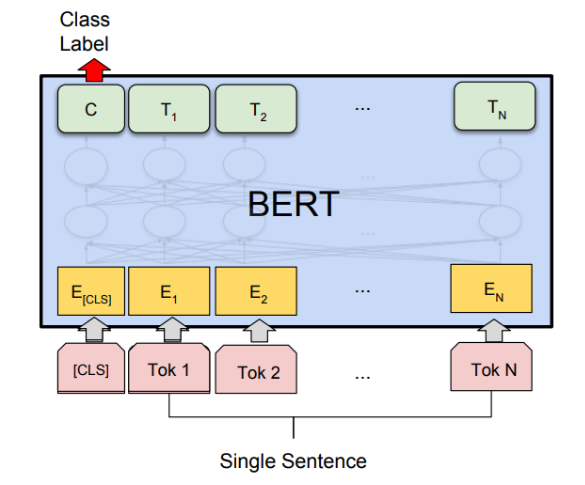
**BERT for Sentence Pair Classification Task:  
BERT has fine-tuned its architecture for a number of sentence pair classification tasks such as:**

* **MNLI: Multi-Genre Natural Language Inference is a large-scale classification task. In this task, we have given a pair of the sentence. The goal is to identify whether the second sentence is entailment, contradiction or neutral with respect to the first sentence.**
* **QQP: Quora Question Pairs, In this dataset, the goal is to determine whether two questions are semantically equal.**
* **QNLI: Question Natural Language Inference, In this task the model needs to determine whether the second sentence is the answer to the question asked in the first sentence.**
* **SWAG: Situations With Adversarial Generations dataset contains 113k sentence classifications. The task is to determine whether the second sentence is the continuation of first or not.**

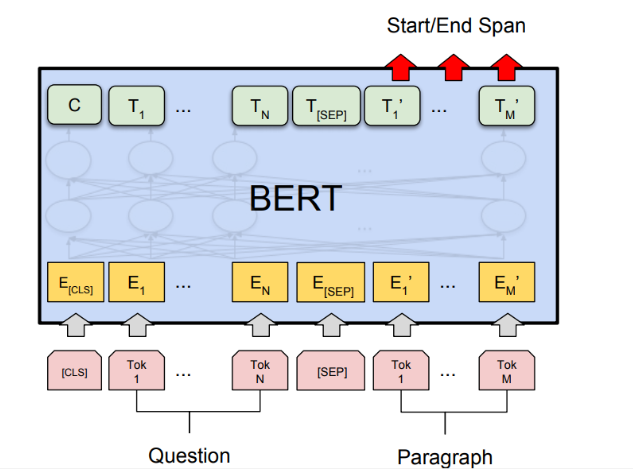
****

**Single Sentence Classification Task :**

* **SST-2: The Stanford Sentiment Treebank is a binary sentence classification task consisting of sentences extracted from movie reviews with annotations of their sentiment representing in the sentence. BERT generated state-of-the-art results on SST-2.**
* **CoLA:The Corpus of Linguistic Acceptability is the binary classification task. The goal of this task to predict whether an English sentence that is provided is linguistically acceptable or not.**

****

Question Answer Task: **BERT has also generated state-of-the-art results Question Answering Tasks such as Stanford Question Answer Datasets (SQuAD v1.1 and SQuAD v2.0). In these Question Answering task, the model takes a question and passage. The goal is to mark the answer text span in the question**



BERT for Google Search: **As we discussed above that BERT is trained and generated state-of-the-art results on Question Answers task. This was the result of particularly due to transformers models that we used in BERT architecture. These models take full sentences as inputs instead of word by word input. This helps in generating full contextual embeddings of a word and helps to understand the language better. This method is very useful in understanding the real intent behind the search query in order to serve the best results.**

**BERT Search Query From the above image, we can see that after applying the BERT model the google understands search query better, therefore, produced a more accurate result.**

Conclusion: **BERT has proved to be a breakthrough in Natural Language Processing and Language Understanding field similar to that AlexNet has provided in the Computer Vision field. It has achieved state-of-the-art results in different task thus can be used for many NLP task**

<https://mccormickml.com/2019/07/22/BERT-fine-tuning/>

<https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/>

<https://studymachinelearning.com/introduction-to-bert/>

<http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/>

<https://www.youtube.com/watch?v=s3LBdmZb00g>

<https://www.youtube.com/watch?v=Osj0Z6rwJB4>

<https://github.com/curiousily/Getting-Things-Done-with-Pytorch/blob/master/08.sentiment-analysis-with-bert.ipynb>

<https://github.com/curiousily/Getting-Things-Done-with-Pytorch>

<https://www.youtube.com/watch?v=mw7ay38--ak>

https://www.youtube.com/watch?v=OR0wfP2FD3c