Review of Deep Learning Techniques in NLP (Natural Language Processing)

November 2019| Tao Li | taol4@illinois.edu

1. Introduction

Nature Language Processing is the technology used for Computer to automatically analyze and understand meaning of human's natural language. The development of NLP technologies has enabled a wide range of intelligent applications such as search engine of Google, Bing and Yahoo, recommendation system of Amazon, Netflix. There are also other popular NLP tasks:

Text Mining	*Text mining is the automatic process of analyzing unstructured text and identify useful information or construct structured knowledge data. For example, business people can leverage text mining application to quickly mine the most critical information and turn them into actionable insights.*
Text Classification	*Text classification is the process of assigning categories, segments, or tags to unstructured text data, which can help organize and enrich the complex text into meaningful data. For example, sentiment analysis is an especially important approach by analyzing the social media post and customer review to provide sentiment feedback (Positive, Negative)*
Documentation Summarization	*Documentation summarization is the problem of summarizing a short, accurate, and readable highlights of a longer text documentation. This need to address the large amount of text data to discover relevant information and provide valuable information faster.*
Machine Translation	*Machine translation is the task of automatically converting on natural language into another, preserving the meaning of the input text, and producing fluent text in the output. For example, Language translator between Chinese and English.*
Question Answering	*Question answering is the tasks to translate sentence into an internal representation, so the system generates valid answers to questions asked by and user. Mostly, this need to leverage the massive online web data and knowledge-based graph to provide more accurate and relevant answers.*

For a long time, most methods used to study NLP problem using the statistical methods, to traditional machine learning models. However, these approaches cannot perform well for complicated problems and are usual time-consuming.

In the recent years, the field of NLP has been shifting to deep learning approaches. In both the research and industry, Deep Learning methods are achieving state-of-the-art results on various NLP tasks as compared to traditional machine learning models.

Deep Learning is an advanced machine learning method that takes the input X, and uses it to predict an output Y. Leveraging large dataset of input and output pairs, a Deep Learning model need to be smart and robust to learn from the input/output data to summarize the association and patterns in a much more advanced approach. Deep Learning models are using the neural network algorithm which is composed of input, hidden and output layers learning the parameter to optimize the loss function. The “deep” part of Deep Learning refers to creating a deep neural network which contains a large number of layers, more complicated weights, parameters, which improves its ability to formulate and present more complex functions.

2. Distributed Feature Representation

2.1 Word Embedding

Word embedding is one of the most popular representations of document/text vocabulary. It’s capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. Simply speaking, word embeddings are vector representations of a particular word.

2.1.1 Word2Vec

Word2Vec has been one of the most popular technique to learn word embeddings using the shallow neural network. Which was developed by Tomas Mikolov in 2013 at Google [1]. Word2Vec first starts with one-hot encoded vector to represent each word in an exhaustive vocabulary. (A vector of zeros except for the element at the index representing the corresponding word in the vocabulary.[2]) Here’s one example for the words Rome, Paris, Italy, France.

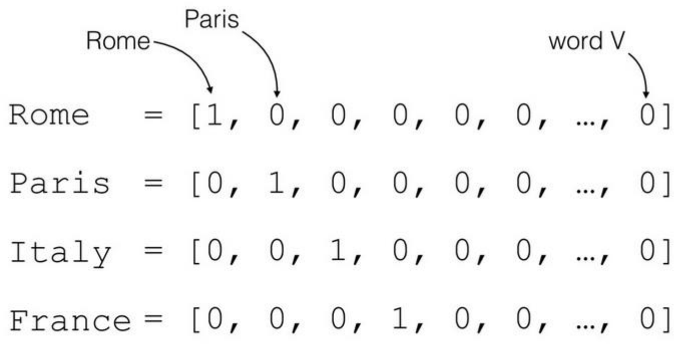
Rome
Paris
word V
Rome = [1, 0, 0, 0, 0, 0, …, 0]
Paris = [0, 1, 0, 0, 0, 0, …, 0]
Italy = [0, 0, 1, 0, 0, 0, …, 0]
France = [0, 0, 0, 1, 0, 0, …, 0]

Figure 2.1: One-hot encoded vector representation. Source :(Marco Bonzanini, 2017)

To generate distributed feature representations, firstly, let’s introduce some dependence of one word on the other words. The words in the context would get a greater share of this dependence.

In one hot encoding representations, all the words are independent of each other. This technique the advantage of simplest methods to implement, however, it’s un-ordered, therefore the context of the words is lost. What’s more, the vector representation is in binary form, therefore no frequency information is taken into account.

Word2Vec is a method to construct such and embedding. It can be obtained using two methods (Both involving Neural Networks): Skip Gram and Common Bag of Words (CBOW). The difference is that the CBOW architecture predict the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

Common Bag of Words(CBOW):

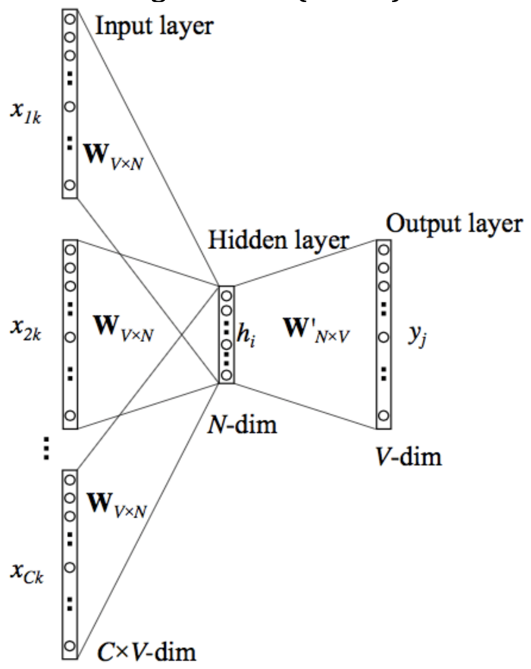

Figure 2.2: A CBOW model

CBOW takes the context of each word as the input and tries to predict the word corresponding to the context.

The process of predicting the target word, we need to use the one hot encoding to measure the output error and learn the vector representations.

Skip-Gram:

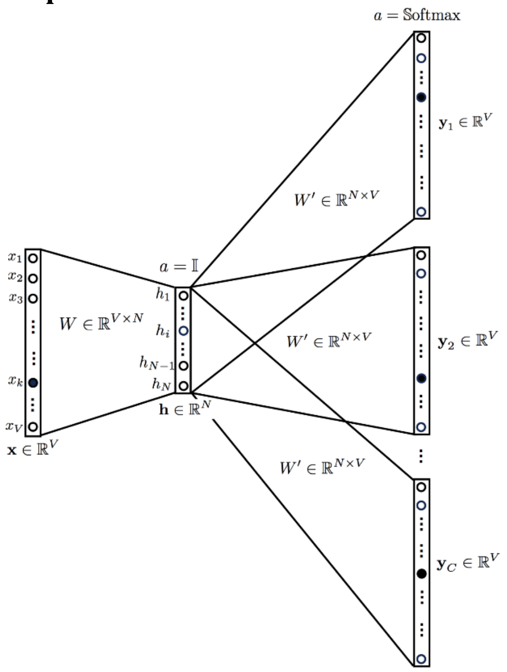
$a = \text{Softmax}$
$\mathbf{y}_1 \in \mathbb{R}^V$
$W' \in \mathbb{R}^{N\times V}$
x_1
x_2
x_3
x_k
x_V
$\mathbf{x} \in \mathbb{R}^V$
$W \in \mathbb{R}^{V\times N}$
$a = \mathbb{I}$
h_1
h_i
h_{N-1}
h_N
$\mathbf{h} \in \mathbb{R}^N$
$W' \in \mathbb{R}^{N\times V}$
$\mathbf{y}_2 \in \mathbb{R}^V$
$W' \in \mathbb{R}^{N\times V}$
$\mathbf{y}_C \in \mathbb{R}^V$

Figure 2.3: A Skip-Gram model

Skip-Gram model uses the target word (which is the target word for prediction in the CBOW model) to predict the context and produce the representations.

2.1.2 GloVe

GloVe is a word vector technique which is very popular in research and industry among all NLP tasks. The advantage of Glove is that, GloVe does not rely just on local statistics (local context information of words), but incorporates global statistics (word co-occurrence) to obtain word vectors.

GloVe stands for "Global Vectors", which can capture both global statistics and local statistics of a corpus, in order to come up with word vectors [4].

	the	cat	sat	on
the	0	1	0	1
cat	1	0	1	0
sat	0	1	0	1
on	1	0	1	0
mat	1	0	0	0

Probability and Ratio	$k = solid$	$k = gas$	$k = water$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36

Figure 2.4: Word co-occurrence and the probability

Given a corpus having V words, the co-occurrence matrix X will be a V*V matrix, where the row i and column j of X means how many times that word i has co-occurred with word j. Based on this, it compute the probability of seeing word i and k together, which is computed by diving the number of times i and k appeared together by the total number of times word i appeared in the corpus.

The GloVe has a process to form a metric to word vectors by measuring the distance between different words following with matrix transpose and dot produce to converting this toa scalar. The method leverages both global and local statistics of a corpus in order to come up with a principled loss function which uses both.

2.2 Convolutional Neural Networks (CNN)

CNN in Image Processing

A Convolutional Neural Networks (CNN) is basically a neural-based approach which represents a feature function that is applied to constituting words or the n-grams to extract higher level or dimensions features. The resulting abstract features have been effectively used for sentiment analysis, machine translation and question answering, among other tasks.

CNN model first comes from image processing and recognitions problem[5], which take each image through a series of convolution layers with filters, pooling, fully connected layers and apply Softmax function to classify an object with probabilistic values between 0 and 1. Here's one process of CNN.

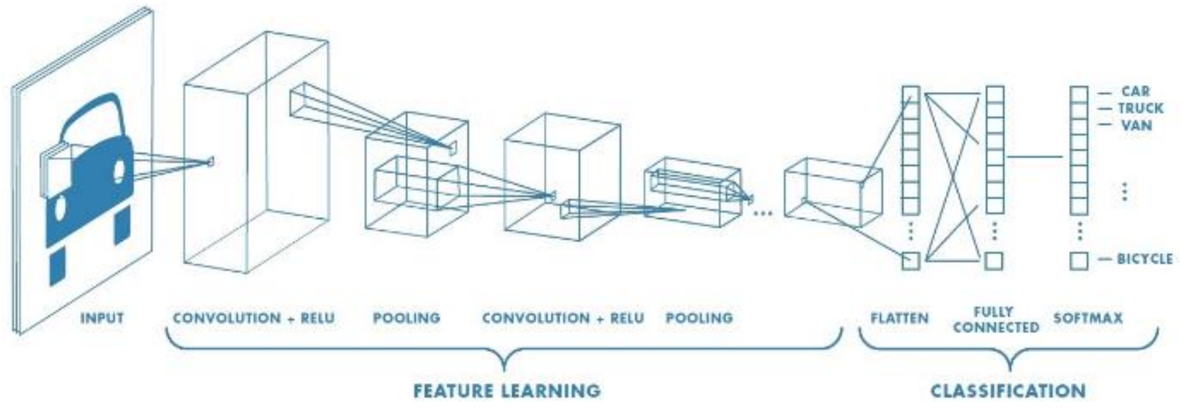

Figure 2.5: Neural network with many convolutional layers

CNN in NLP

Instead of image pixels, the input to most NLP tasks are sentences or documents represented as a matrix.

- *Each row of the matrix corresponds to one token, typically a word, but it could be a character. That is, each row is vector that represents a word.*
- *Typically, these vectors are word embeddings like word2vec or GloVe, but they could also be one-hot vectors that index the word into a vocabulary.*

CNN can also benefit the NLP tasks taking the advantage of context information. For example, in order to perform sentence modeling with a basic CNN, sentences are first tokenized into words, which are further transformed into a word embedding matrix (input embedding layer) of d dimension. Then, convolutional filters are applied on this input embedding layer which consists of applying a filter of all possible window sizes to produce what's called a feature map. This then followed by a max-polling operation which applied a max operation on each filter to obtain a fixed length output and reduce the dimensionality of the output. And that procedure produces the final sentence representation.

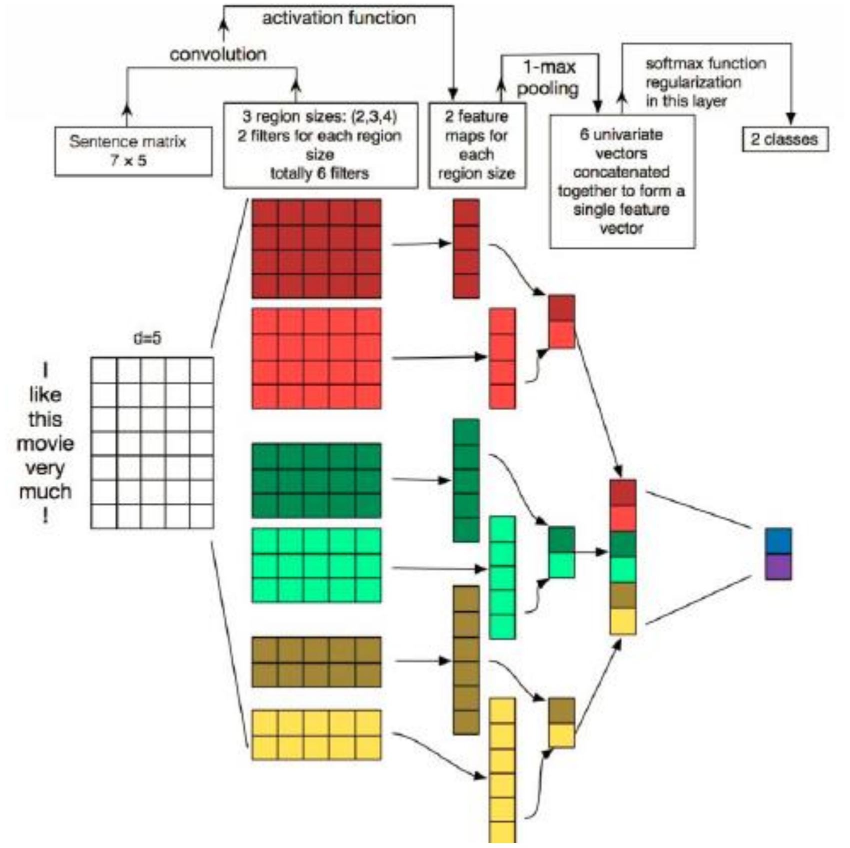
activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Figure 2.5: CNN in NLP (Sentence Task)

By increasing the complexity of basic CNN and adapting it to perform word-based prediction and other NLP tasks. Overall, CNNs are effective because they can mine semantic clues in the contextual windows, but they struggle to preserve sequential order and model long-distance contextual information.

2.3 Recurrent Neural networks (RNN)

Traditional neural networks (CNN) have not ability to remember longer memory and benefit the task prediction, which is a major shortcoming. Recurrent neural networks (RNNs) can address this pain point. They are advanced networks (with loops) in them, allowing information to persist.

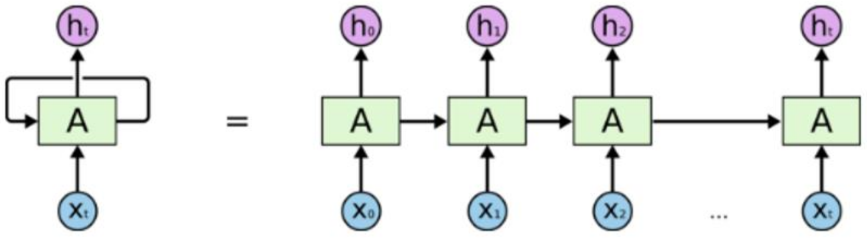

Figure 2.6: Recurrent Neural Networks

RNNs can be thought of as multiple copies of the same network (each passing a message to a successor). Its chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists of features.

The major reason that recurrent nets are more significant allowing to operate over sequences of vectors: Sequences in the input[6], the output, or in the most general case both. As it compares with CNN, RNN model can be similarly effective or even better at specific natural language tasks but not necessarily superior. One reason it they model very different aspects of the data, which only makes them effective depending on the semantics required by the task in real world application.

2.4 Long Short Term Memory (LSTM) Networks

With the recent breakthroughs that have been happening in Deep Learning, it is found that for almost all of these sequence prediction problems, it has been found and proved that Long short Term Memory networks (LSTMs) as the most effective approach [7] for NLP tasks.

To be more advanced over conventional CNN and RNN in many ways, LSTM has the ability to remember patterns for long duration. For example, LSTMs make small but significant modifications to the information by multiplications and additions. Using LSTMs, the information can flow through a mechanism called cell states. Benefited by this technique, LSTMs enable selective **remember** or **forget** information.

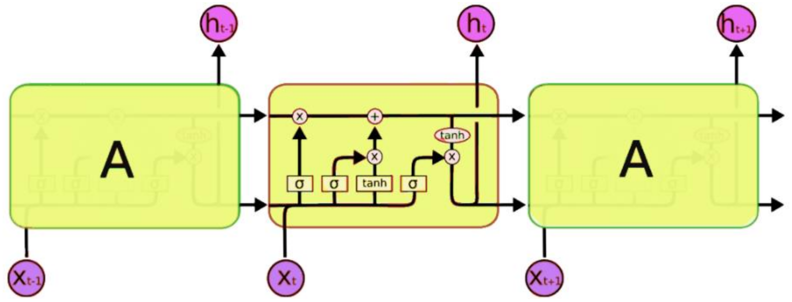

Figure 2.6: LSTM architecture

A typical LSTM network is comprised of different memory blocks called cells (the rectangles in above chart). There are two states that are being transferred to the next cell: cell state and hidden state. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, known as gates.

LSTMs are a very promising solutions to sequence and time-series related tasks. However, the cost to train the model is higher than simple models.

3. New Model Architecture

3.1 Transformer

To better solve the sequence-to-sequence NLP tasks, the Transformer becomes to be a novel architecture handling long-range dependencies with ease [8]. The Transformer is the first invented transduction model depending entirely on self-attention to compute the representations of input and output without using sequence aligned CNNs and RNNs[10].

In transformer, "transduction" is the conversion of input sequences into output sequences. The technique behind Transformer is to handle and solve the dependencies between input and output with attention and recurrence completely.

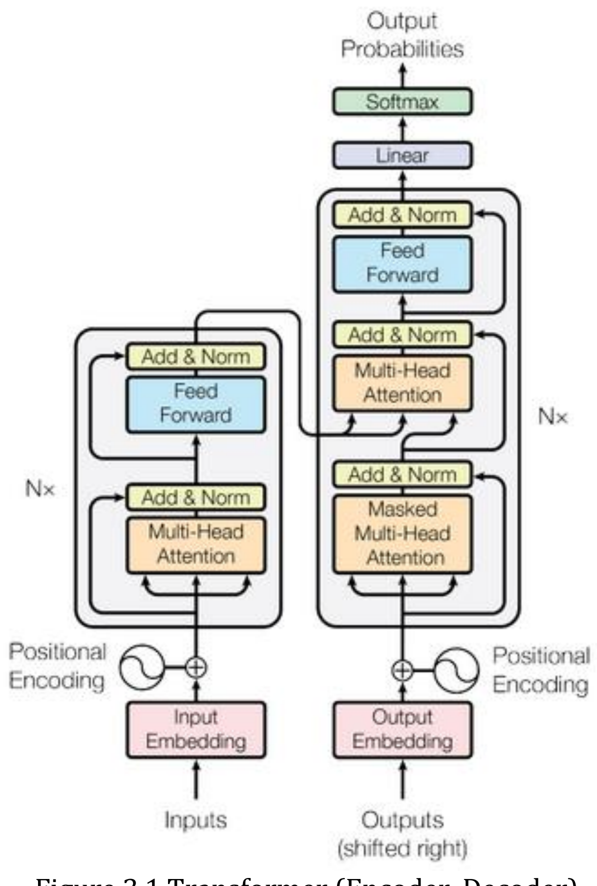

Figure 3.1 Transformer (Encoder, Decoder)

The Left chart is a superb illustration of Transformer's architecture. The most important parts are "Encoder" and "Decoder". ***Encoder****: 1 layer of a Multi-Head Attention followed by another layer of Feed Forward Neural Network.*
Decoder*: an extra Masked Multi-Head Attention.*
The encoder and decoder blocks are actually multiple identical encoders and decoders stacked on top of each other. Both the encoder stack and the decoder stack have the same number of units.

1. *The input sequences are passed to the 1st encoder using word embeddings*
2. *Using transformed and propagated to the next encoder*
3. *The output from the last encoder in the encoder is passed to all the decoders in the decoder-stack as shown in the figure below:*

Beyond the self-attention and feed-forward layers, the decoders also have one more layer of Encoder-Decoder Attention layer. Which enables the decoder focusing on the appropriate parts of the input sequence.

3.2 Google's BERT

The BERT framework is a new language representation model from Google AI, based on pre-training and fine-tuning to create state-of-the-art models for a wide range of different tasks (For example, question answering systems, sentiment analysis, and language inference tasks [11]).

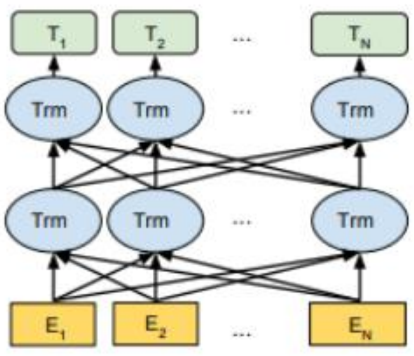

Figure 3.2 Google BERT architecture

BERT uses bidirectionality by pre-training on a couple of tasks — Masked Language Model and Next Sentence Prediction.

1. **Masked Language Modeling (MLM)**

The masked language model randomly masks some of the tokens from the input, and the objective or goal is to predict the original vocabulary id of the masked word based only on its context. Unlike left-to-right language model pre-training, the MLM objective allows the representation to leverage the left and the right context, which allows to pre-train a deep bidirectional Transformer.

2. **Next Sentence Prediction**

BERT was pre-trained on these common tasks as well, using pairs of sentences as its training input. To let the model distinguish between the two sentences in training, here's the pre-process:

1. *Insert [CLS] token at beginning of the first sentence and a [SEP] token at the end of sentence.*
2. *A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.*
3. *Add a positional embedding to each token indicating its position in the sequence.*

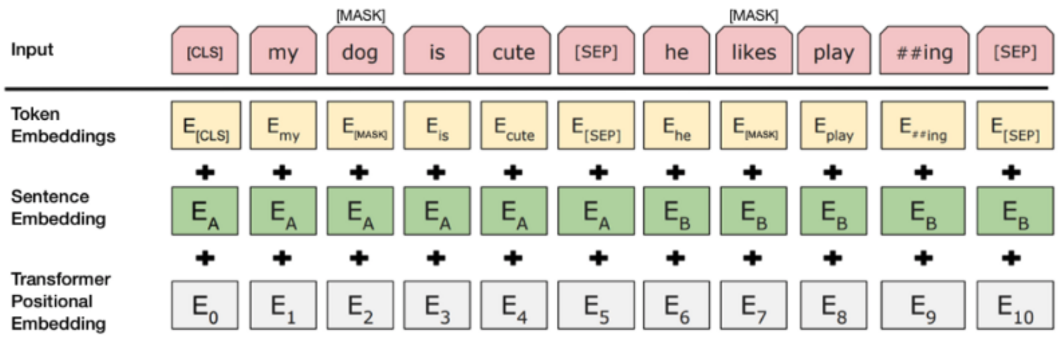
Input
[MASK]
[MASK]
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 3.3 Google BERT architecture

To predict if the second sentence is indeed connected to the first, here's the following steps:

1. *Entire input sequence goes through the Transformer model.*
2. *The output of the [CLS] token is transformed into a vector (2×1 shaped), using a simple classification layer.*
3. *Calculating the probability of IsNextSequence with Softmax function.*

With the development of BERT and other Pre-Trained deep learning transformer models. It's now super easier to leverage the Pre-Trained models to solve real world problems with a high performance. Transfer learning will enable firms to use pre-trained models to create applications to perform tasks such as text classifications, sentiment analysis and so on.

4. Summary

A lot of research and industry applications have proved that Deep Learning as the state-of-the-art approaches in NLP tasks. Understanding the fundamental techniques of Machine Learning and Deep Learning is super important as data scientist to solve real world NLP problem. To follow and understand which way NLP research trending is important as well to apply the cutting edge of techniques in your own business. It's clear to see that:

- CNN and RNN are no longer an NLP standard architecture
- The transformer will become the dominant NLP deep learning architecture
- BERR among with more PreTrained model will transform the NLP application landscape
- Fine-tuning models will get easier and more useful in the real world application

REFERENCES

1. Distributed Representations of Words and Phrases and their Compositionality, 2013
2. https://www.pycon.it/media/conference/slides/word-embeddings-for-natural-language-processing-in-python.pdf
3. Efficient Estimation of Word Representations in Vector Space 2013
4. GloVe: Global Vectors for Word Representation
5. ImageNetClassificationwithDeepConvolutional NeuralNetworks
6. Comparative Study of CNN and RNN for Natural Language Processing
7. Contextual LSTM (CLSTM) models for Large scale NLP tasks
8. Attention Is All You Need
9. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
10. Reinforcement Learning for NLP
11. https://github.com/google-research/bert