BERT MultiLabel Classifier with low-latency

Saleem Ansari (@tuxdna) at DevConf.IN/2019

Objectives

- Introducing BERT model
- Embedding representation
- Single Label Classifier
- Low Latency Architecture
- Demo

Google BERT Model

- Some background on vector space / word-embeddings
- Contextual Embeddings and Self Attention i.e. Transformer Architecture
- BERT is the Encoder only part from Transformer
- BERT is bidirectional due to the two tasks it pre-trains on i.e MLM and NSP

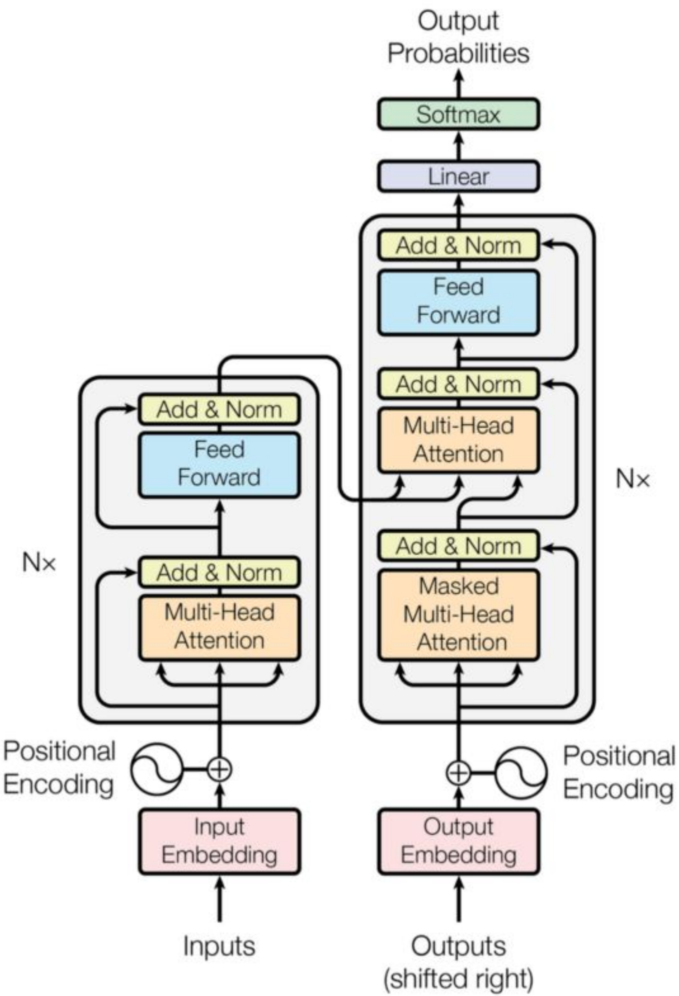
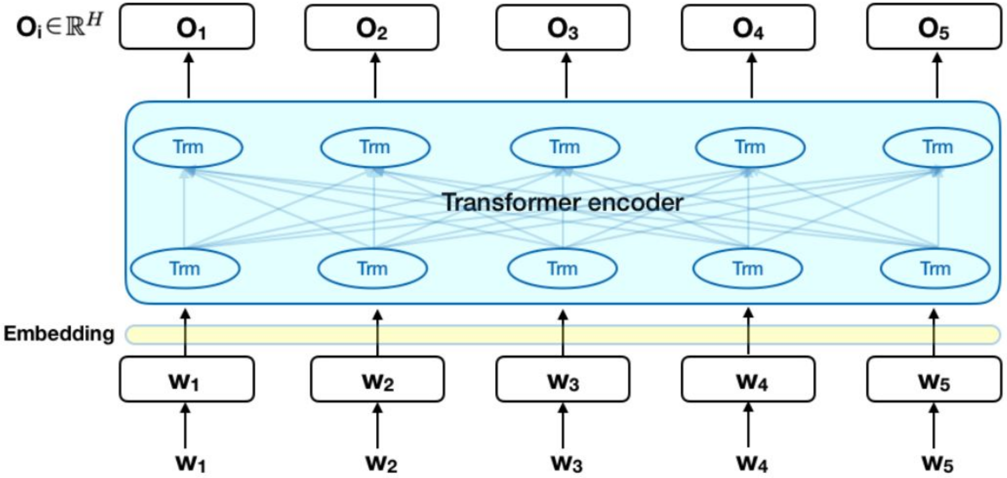


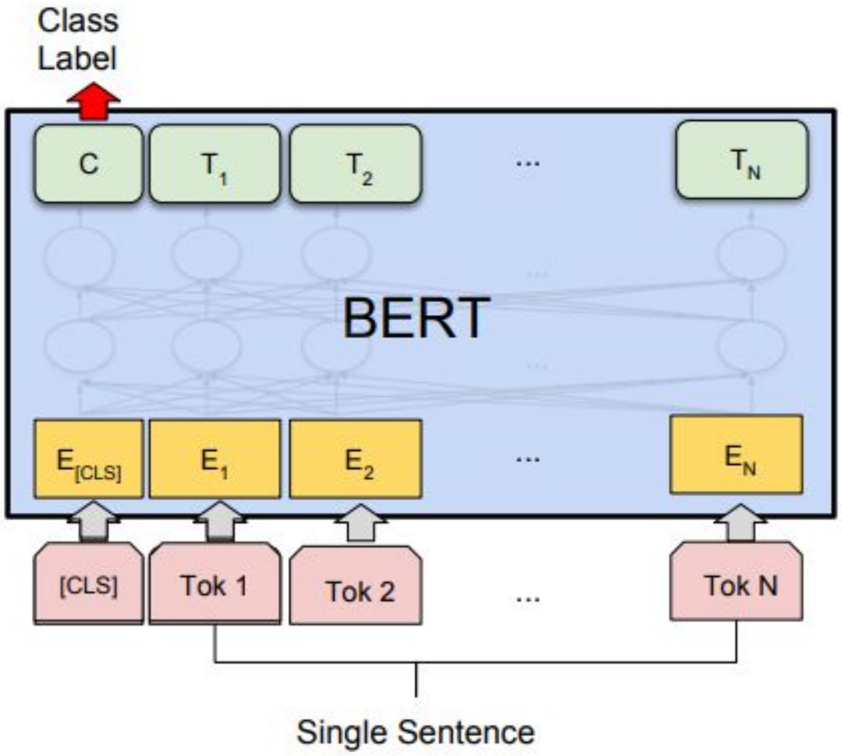
Figure 1: The Transformer - model architecture.



BERT Encoder from [BERT – State of the Art Language Model for NLP]

Embedding Representation

- BERT Serving Project
- CLS token
- Fine-tuning vs Feature-based

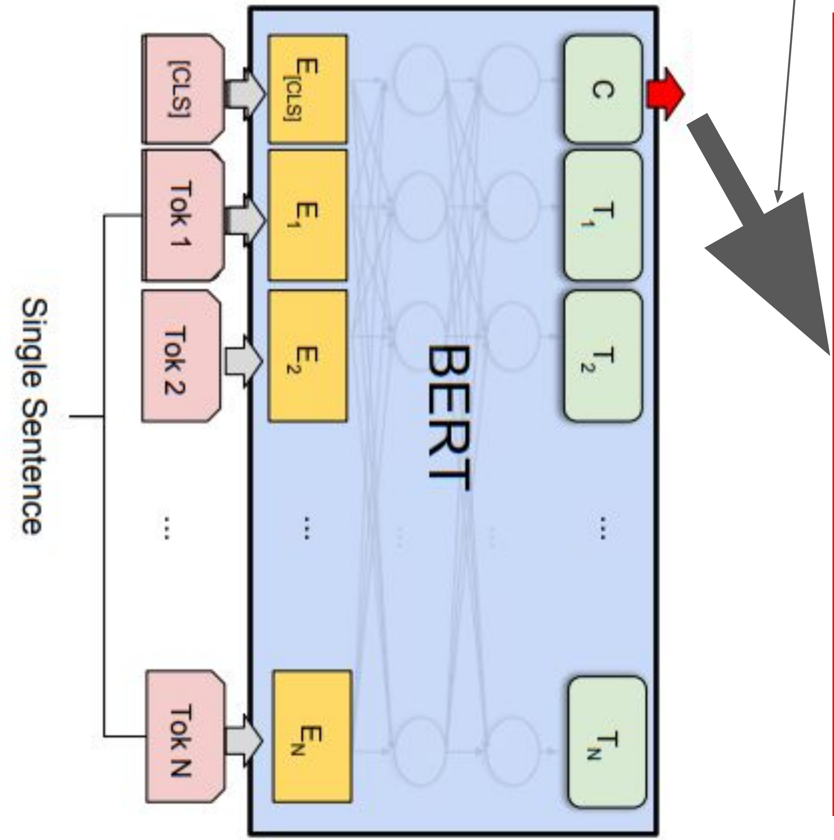


Single Label Classifier

- Obtain embeddings from BERT Server (defaults to **[CLS]** i.e. the first token)
- Create Keras based classifier using embeddings obtained above as input features
- Evaluation Metrics - RMSE, False Positive Rate, Accuracy

BERT Embeddings

Single Label Classifier



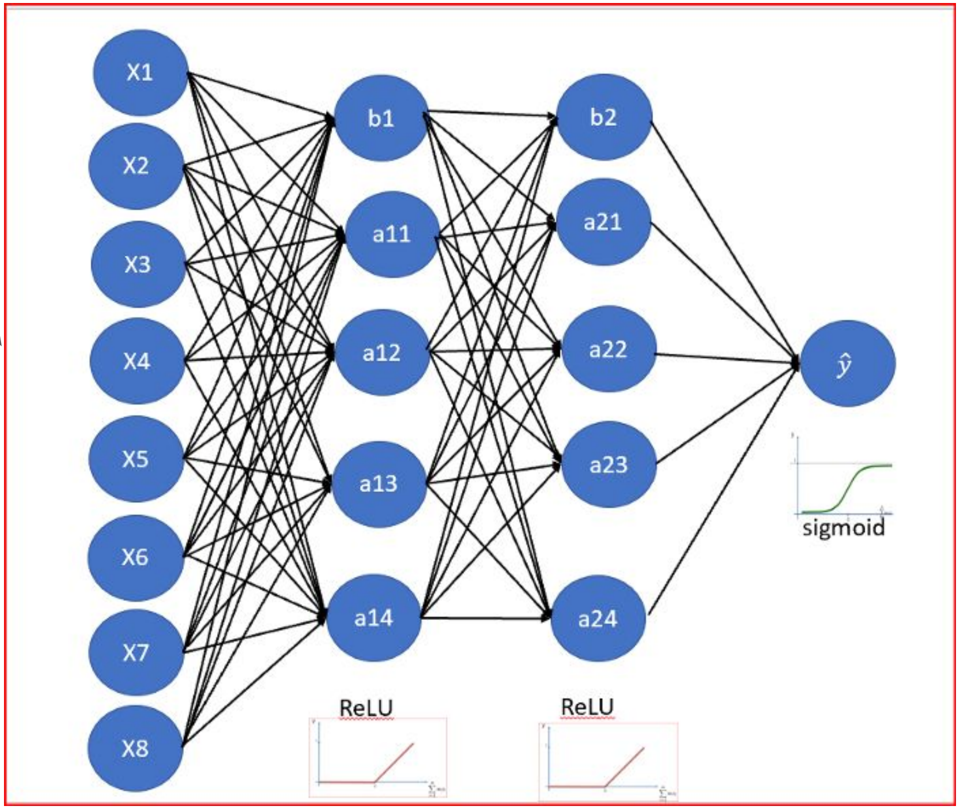


Image Source

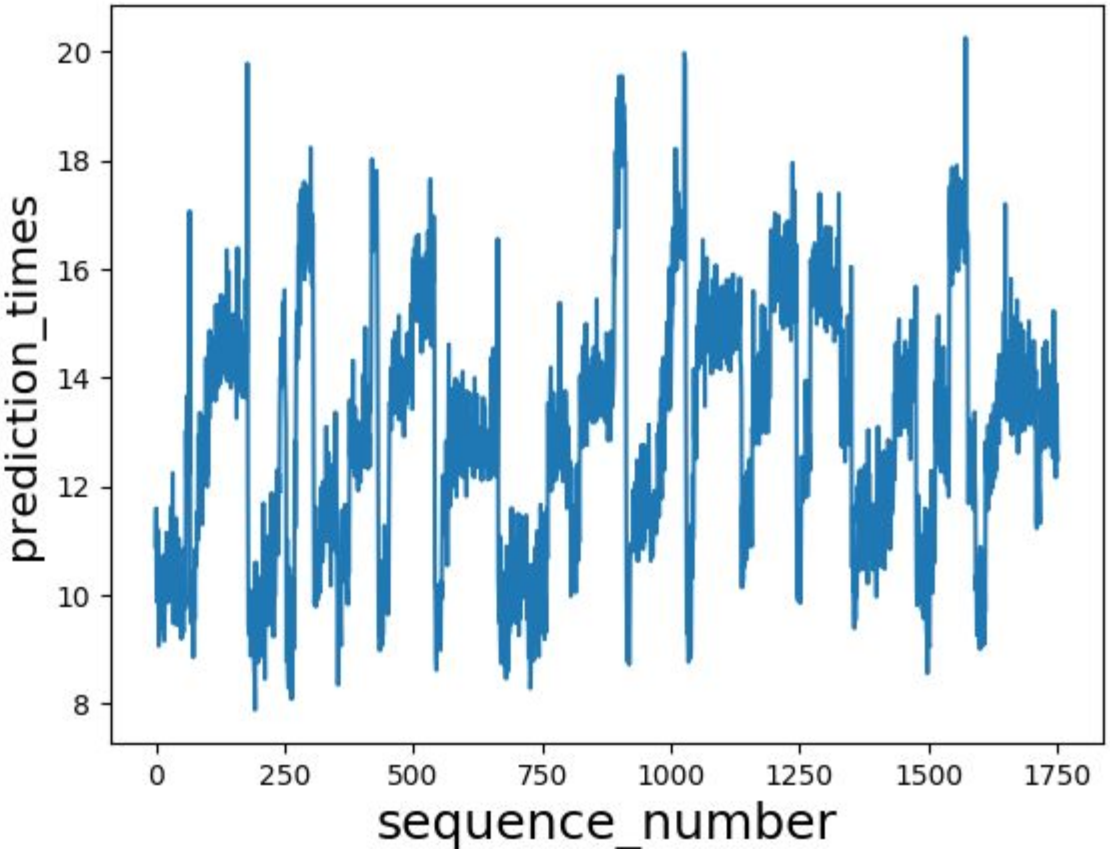
Low Latency Architecture

- Use same embeddings for different single-label classifiers (*then these can run in parallel*)
- Use Keras (Python API) for training the single-label classifier models
- Time for fetching BERT embeddings dominates evaluating a single-label classifier layer
- IMPORTANT: Use **FP16** for reducing prediction time and model size both.
- IMPORTANT: Use ***max_seq_len=NONE*** when running BERT server
- We can run BERT Server in parallel on multiple GPUs.
- (Optional) Use DL4J for serving the Keras models (for Java projects)

Demo

- WARNING: This is a demo of toxic comments which may contain some obscene language (part of the dataset).
- What do the metrics say?
- Evaluation on Test data.
- Evaluation on some custom text.
- Evaluation on sentences from a movie script.
- Average latency is about 12ms overall, with a single machine / single GPU setup. More hardware can help increase throughput. Better hardware can help reduce latency.
- The predictions also look quite sensible!

Prediction time over different predict() calls



minmax=(7.88, 20.25)

mean=12.92

variance=5.29

skewness=0.17

Additional Remarks

- BERT can and does work on very less number of labeled samples.
- This demo only used ***BERT base uncased*** model. A larger model will increase accuracy but will not be as a fast. **Time vs Accuracy tradeoff.**
- After pretraining on domain-specific data, it generally improves accuracy of the models.
- Chances of out-of-vocabulary tokens is reduced considerably by WordPiece (BPE) tokenizer.
- Also try pooling of different layers (combined with MEAN, MAX operators).
- BERT works on many other NLP Tasks (please do read the paper).
- (My Experience) Pytorch implementation of BERT takes around 40ms.

Questions?

Thank you!

References below:

- https://github.com/hanxiao/bert-as-service
- https://www.lyrn.ai/2018/11/07/explained-bert-state-of-the-art-language-model-for-nlp/
- http://jalammar.github.io/illustrated-bert/
- https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
- https://towardsdatascience.com/deconstructing-bert-distilling-6-patterns-from-100-million-parameters-b49113672f77
- https://towardsdatascience.com/deconstructing-bert-part-2-visualizing-the-inner-workings-of-attention-60a16d86b5c1
- http://www.valkaama.com/index.php
- https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data
- https://arxiv.org/abs/1706.03762
- https://arxiv.org/abs/1810.04805