Multi-label Document Classification with BERT

blog published September 14th 2019

all comments to contact@andriymulyar.com

accepted extended abstract at NeurIPS 2019 ML4Health (will be updated with arxiv link soon)

codebase: pytorch architectures and trained models

From rules and one-hot-vectors to embeddings and neural language models the natural language processing (NLP) community has indisputably witnessed a historic decade of growth. Now on the eve of a new decade, I am of the firm belief that language models will be cemented as the go-to recipe for high performance on downstream text processing tasks for many years to come. As engineering and research drives down model sizes, applying language model based systems at scale will surely gain feasibility. This post describes in precise terms a small contribution that allows neural language models with fixed context windows (such as BERT) to be applicable towards tasks that require signal that may possibly appear anywhere in a text document. The most straightforward example of such a task is document classification. This article is meant to be accessible and introductory yet precise; if you are confident feel free to skip to section Unrolling BERT.

Multi-label document classification

Given a set of text documents D the task of document classification is to find (estimate) a function $M : D \to \{0, 1\}^L$ that corresponds to a given document $d \in D$ a vector of labels $\vec{l} \in \{0, 1\}^L$ which dimension-wise serves as an indicator of various semantic information related to the contents of d.

Example.

If D is a set of news articles, it may be of practical interest to assign labels that indicate categories of news: for instance sports, politics and world. Here $L = 3$ hence the task of M is to in some way exploit the text in d to ascertain the correct indicator (1 or 0) for each label. Notice that a document can be given **multi**ple **label**s (e.g. politics and world with $M(d) = (0, 1, 1)$) hence it would intuitively be useful if M leveraged correlations between labels.

BERT

BERT is a textual language model that allows the embedding of a fixed context window of text into a Euclidean space while preserving properties of the text (syntactic, contextual, etc). It is trained over a large, un-supervised corpus in an encoder-decoder fashion with the final encoder parameters utilized to generate representations of new text. For the purpose of this write-up, we will refer to BERT as a black box $\mathcal{L} : (t_k) \to \mathbb{R}^n$ that takes as input a sequence of tokens and outputs a vector in $\mathbb{R}^n$ that represents a condesed form of the token sequence. In practice, BERT additionally produces per-token embeddings but the pooled sequence embedding will suffice for our task of interest. For the un-familiar reader, this blog post gives an illustrative overview of the original paper without it's page length limitation.

So what's the problem?

Why can't one directly utilize a language model such as BERT for document classification? The bottle-neck restriction is that the BERT encoder accepts a **fixed context window**. This means that the model can only encode and generate contextualized representations for a fixed, finite sequence of tokens. For instance, a well know implementation of BERT limits this context window size to $\ell = 512$. Such a context window size restriction is imposed for tractability - speeding up pre-training and reducing memory requirements during utilization.

If a tokenized document can fit inside the window size of $\sim \ell$ language model inputs, then one can proceed with normal utilization (finetuning/freezing) and expect decent performance. This is what the April 2019 article DocBert explored. Things get more interesting when documents are long. For our purposes a long document is one that **cannot** fit inside BERT's context window. Explicitly, the length of a WordPiece tokenization of the document - with respect to the BERT pre-training vocabulary - exceeds $\ell - 2 = 510$ tokens. The extra two tokens are special padding tokens ([CLS] and [SEP]) involved in the model pre-training objective that we must include during encoding.

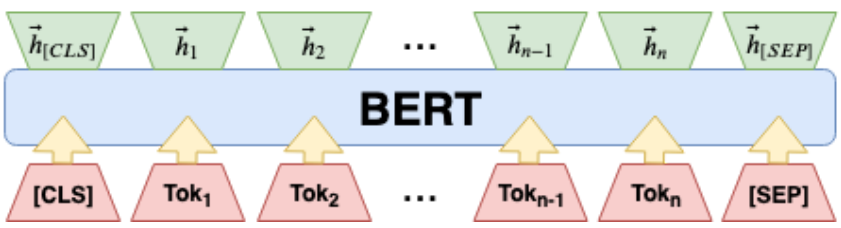

Figure 1: BERT. WordPiece tokenized text is embedded into Euclidean space. Recall our restriction that we are only concerned with the pooled sequence representions $\vec{h}_{CLS}$. Thus in our discussion, BERT is $\mathcal{L} : (t_k) \to \vec{h}_{CLS}$. This ignores the token representations $(\vec{h}_n)$ - otherwise the output is a real valued matrix.

Unrolling BERT

Notation

Define a document d as a finite sequence of tokens (t_k) with $k \in \{1, \dots Q\}$ where Q is the length of the document tokenization. We will work under the assumption that the document tokenizer in use is the WordPiece tokenizer with the vocabulary $\mathcal{L}$ was trained against. We say that d is a long document if $Q \gg \ell$ where ℓ is the encoding context window of $\mathcal{L}$. This means that the document token sequence (t_k) cannot fit inside the context window of $\mathcal{L}$.

Representing a long document

In order to represent a long document d for classification with BERT we "unroll" BERT over the token sequence (t_k) in fixed sized chunks of size ℓ. This allows us to generate a sequence of contextualized token sequence representations

$$(\vec{h}_p) \ : \ \vec{h}_p \ = \mathcal{L}(\ (t_k)_{k=p\cdot\ell}^{(p+1)\cdot\ell}\ ) \text{ for } p \in \{0, \dots, \lfloor \frac{Q}{\ell} \rfloor\}$$

that each encode a fixed length subsequence of (t_k). Intuitively [1] , each element of $(\vec{h}_p)$ encodes a sequential sub-chunk of d. So how do we compress this **sequence** of representations while retaining the label indicative signal that any given sequence element may be encoding? Naturally, a RNN is fit for this job. More generally, any algorithmic technique that falls under the class of seq2seq learning algorithms will do. In our experiments we find the best results from utilizing the last hidden state of an LSTM run over the sequence $(\vec{h}_p)$.

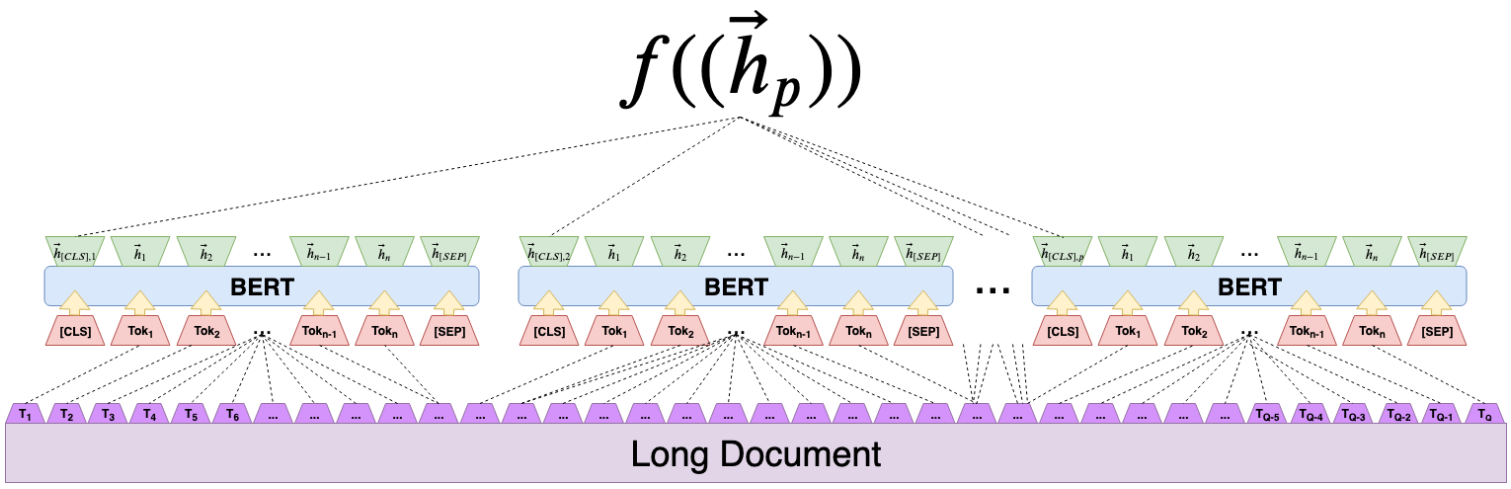

Figure 2: Document Bert. A function f parameterized by a seq2seq neural network condenses a sequence of BERT hidden states into a global document context vector. For instance, f could be an LSTM cell.

This process is visualized in Figure 2. Working bottom up, a long document is tokenized with respect to a word piece tokenization, the tokenization is partitioned in 510 token sequential token subchunks, each sub-chunk is contextually encoded by BERT to generate a sequence of contextualized subchunk representations and finally a seq2seq neural network condenses the sequence of subchunks into a global context vector. Our paper experiments with several condensation functions and finds that one parameterized by an LSTM works well.

Experiments and Results

Moving forward we will refer only to the case where $f((\vec{h}_p))$ is the last hidden state of an LSTM run over $(\vec{h}_p)$. For exploration of other condensation functions, please see our paper.

We evaluate this architecture on two EHR patient phenotyping tasks. Phenotyping is the task of extracting the presence of patient conditions from free clinical text. It is naturally represented as a document classification task. Suited to our problem of interest, clinical notes are often multi-page documents with tokens sampled from an ultra-specific domain lexicon (clinical terminology and non-intuitive abbreviations). This means that information must be synthesized from the global document context as one does not know a priori where label indicating signal may be present. For some perspective, most notes when tokenized consisted of over 4,000 individual word-piece tokens ($(\vec{h}_p)$ composed of ~ 8 BERT hidden state vectors). During training, we utilize a frozen BERT model fine-tuned over EHR records.

The evaluation focused on two phenotyping tasks - detection of smoking status via a four-label multi-class classification task and the detection of obesity and related co-morbidities via a 15-label multi-label classification task. Not only does this architecture beat previous state-of-the-art on both datasets, we achieve near perfect evaluation set generalization performance suggesting the retirement of these datasets as good measures of model competitiveness. For detailed results and numbers please see our paper or codebase.

Final Notes

- BERT does not just have to be utilized for encoding one or two sentences - the entire context window produces excellent representations.
- No hand engineered information is needed for patient phenotyping - heavily parameterized models can outperform human hand engineering (regex building and look up table) efforts.

Future work ideas

- While the architecture performs great, predictive accuracy is by no means the sole indicator of a models viability in a production setting. It would be useful to decrease the model parameter size via distillation. Alternatively, I believe a more thorough investigation could be useful to ascertaining the true necessity of such a large language model for this task. Would a distilled language model such as DistillBERT retain the same predictive performance?
- It is not unreasonable to assume that label indicating signal appears in only a small subset of the document. Can such training and inference in such an architecture be sped up by pruning out document sections ahead of time? How can one do this consistently?