UIUC CS 410: Text Information Systems
Fall 2022
Stuart Jaffe (ID: sijaffe2)

BERT Versus SBERT and Its Augmented Extensions

BERT Overview

BERT, an acronym for Bidirectional Encoder Representations from Transformers, is a natural language processing ("NLP") model developed by researchers at Google in 2018. BERT's primary innovation is in the "Bidirectional" portion of its name; previous cutting edge language models were trained on left-to-right and right-to-left contexts independently, thus denying the language model the deep bidirectional semantic context that human interpretation takes for granted. BERT achieves the bidirectional context utilizing two training strategies: the Masked Language Model ("MLM") and Next Sentence Prediction ("NSP"), which will be discussed below. Building off of OpenAI's Generative Pre-trained Transformer ("GPT") model, BERT is a pre-trained NLP model that can then undergo fine-tuning for more specific downstream tasks, such as question and answer or text classification. This is in contrast to the feature-based NLP models, such as ELMo, which employ specific architectures for downstream tasks, leading to specific additional features for each task. Fine-tuning for downstreams tasks allows the generalized upstream BERT model to be trained on a plethora of unannotated text gathered from various sources; in fact, BERT was trained on the BooksCorpus (800 million words) and the English Wikipedia (2.5 billion words), extracting long contiguous sentences instead of a shuffled sentence-level corpus. At the time of publication, BERT was state of the art for 11 downstream NLP tasks.

BERT Transforms Tokens

BERT uses a Transformer to take a text input and outputs (i.e. decodes) an N-dimensional embedding for each token and an N-dimensional embedding for the entire text input, where N = {768, 1024}, depending on the BERT model used. Transformers rely exclusively on self-attention mechanisms to compute the N-dimensional embeddings, compared to previous models that used sequence aligned recurrent neural networks ("RNNs") and convolution. Attention mechanisms allow the decoder to access all of the hidden states in the Transformer, instead of just the intermediate state produced by the encoder's last hidden state. This is best illustrated with an example, "The man saw a boy with a telescope that he received from his **parents**." "The man saw a boy with a telescope…" is a classic example of syntactic ambiguity but "...that he received from his parents" indicates to the attention mechanism that the boy is likely the one with the telescope. If one word is changed, "The man saw a boy with a telescope that he received from his **children**." Now the attention mechanism is able to figure out the reference object for "his" and the prepositional phrase "with a telescope" refers to the man and not the boy, most likely.

The token embedding methodology is a prerequisite to discussing BERT's innovative approach to bidirectional context training. The input to BERT is a set of tokens derived from the original text with a [CLS] token added at the beginning of the first sentence and a [SEP] token added at the end of each sentence. Then, the sentence of origin is added to each token, such as whether it's from Sentence A or Sentence B, and finally the numeric position of the original word is added to each token. The Transformer stacks these layers of tokens and feeds them into the neural network.

BERT Looks Both Ways

It stands to reason that a NLP model trained on both left-to-right and right-to-left context would be superior to a NLP model trained on just one context direction. However, if a naive approach to bidirectional training is used, it would allow a word to "see" itself and predict the target word with ease, a classic case of information leakage in machine learning. BERT avoids this issue by masking 15% of the input tokens at random with a [MASK] token (Masked Language Model). Then the sequence of words is run through the BERT Transformer and the masked tokens are predicted based on the context of surrounding words using a traditional soft-max approach to word probability. However, this leads to an issue in the fine-tuning stage wherein the model only tries to predict when there is a [MASK] token present and in reality it should be trying to predict regardless of what token type is present. The developers of BERT fix this issue by taking the 15% of [MASK] tokens and within that 15%, 80% are actually replaced with the [MASK] token, 10% are replaced with a random token, and 10% are left unchanged.

The Masked Language Model provides context within a sentence, but there is still context needed between sentences (e.g. for question and answer type tasks). This is where the NSP arrives. During training, the BERT model receives input pairs of sentences in order to predict whether the second sentence in the sequence is actually part of the original text. As with any machine learning model, BERT needs variation in the training data. Thus, half of the NSP training examples are from the original text where the second sentence does in fact come after the first sentence and the other half of the examples the second sentence is a random sentence drawn from the training data corpus. In the latter situation, the second sentence has a high probability of being unrelated to the first sentence. The [CLS] token is then the container for the prediction (labeled as IsNext) via softmax whether the second sentence is supposed to follow the first.

SBERT: Sentence BERT

Motivation:

While BERT has innovated on many NLP tasks, it is computationally very expensive. The base BERT model, with an output of 768 dimensions, was trained on 4 TPUs for 4 days and the large BERT model, with an output of 1024 dimensions, was trained on 16 TPUs for 4 days. TPUs are Tensor Processing Units, a specialized chip specifically designed by Google for deep learning tasks. While the base and large BERT models don't need to be retrained, it's still computationally challenging for the average person or small startup trying to utilize this technology. For example, let's you want to find the most similar sentences in a book with 10,000 sentences. Finding the pairs with the highest similarity scores with BERT requires almost 50,000,000 computations (the dreaded $O(n^2)$ time complexity via n*(n-1)/2). Using one of NVIDIA's recent V100 GPUs, this would take over 65 hours. The time derives from BERT's use of a cross-encoder: a sentence pair is fed into the transformer network and the result is the target value.

Solution:

As seen in the vector space model earlier in the first half of CS 410, one way to determine semantically similar sentences and address clustering is to map each sentence to the vector space with an N-dimensional vector then use a similarity metric, such as cosine similarity, to determine which sentences are related to each other. In the context of BERT, this method would involve taking the N-dimensional vector from [CLS] token or average (i.e. pooling) the N-dimensional vectors for each token in the sentence. The developers of SBERT show that either of these methods results in poor sentence embeddings. This led them to develop a siamese architecture for BERT, SBERT, where they use a bi-encoding methodology. Multiple sentences can be embedded simultaneously in parallel then a pooling operation is applied to gather sentence embeddings of a fixed dimension.

Pooling Strategies:

The developers of SBERT utilize three different pooling strategies: Classification Objective Function, Regression Objective Function, and Triplet Objective Function.

The Classification Objective Function appends the target sentence embeddings, *u* and *v*, with the element-wise differential, |*u* - *v*| and then multiplies it by the trainable weight, $W_t \in R^{3n \times k}$, where n is the dimension of the sentence embeddings and k is the number of labels. The objective function is, o = softmax($W_t(u, v, |u - v|)$) with cross-entropy loss as the optimized loss function.

The Regression Objective Function is simply the cosine similarity between the two sentence embeddings, cos_sim(A,B) = A x B / (||A|| ||B||) with mean-squared-error as the optimized loss function.

The Triplet Objective Function takes a base sentence *a*, a related sentence *p*, and an unrelated sentence *n*, and tunes the network such that the distance between *a* and *p* is smaller than the distance between *a* and *n*. The loss function optimized is, $\max(||s_a - s_p|| - ||s_a - s_n|| + e , 0)$, where s_i is the sentence embedding for *a/n/p* and e is a margin that ensures sentence *p* is at least e units closer to sentence *a* than to sentence *n*. The default distance metric is Euclidean distance and the default value for e is 1.

Augmentation:

SBERT and BERT can be combined to leverage a labeled training data set. For example, if you're trying to predict whether a corpus of sentences is unrelated (-1), neutral (0), or related (1), you would ideally have a labeled training data set. However, labeling training data requires hours of work by humans, which is often too expensive for some companies or inaccessible for other reasons. In addition, while the SBERT bi-encoder is computationally more efficient than the BERT cross-encoder, some performance is sacrificed with SBERT. Enter Augmented SBERT. Let's say we start with 1,000 to 5,000 labeled sentence pairs, which is the "gold" data set. We then use a sampling strategy, discussed below, to create a set of new sentence pairs that are not present in the "gold" data set. BERT is trained on this set of new sentence pairs, which produced a "silver" data set. Finally, an SBERT bi-encoder is trained on the extended, gold plus silver combined, data set. This can substantially improve the performance of the SBERT model.There are several sampling strategies to create the new data set for the BERT model.

The first is Random Sampling. Given there are n*(n-1)/2 possible combinations for the gold data set sentences, Random Sampling will likely produce too many dissimilar sentence pairs and not enough sentence pairs, leading to a biased training data set.

The second is Kernel Density Estimation. A large set of randomly sampled sentence pairs are selected from the gold data set and then labeled with BERT. Usually all of the positively labeled pairs will be kept and the negatively labeled pairs will be thrown out so the new training data set has the same proportion of positives/negatives/neutrals as the gold data set. This method is computationally inefficient given the number of discarded pairs.

The third is BM25 Sampling. All the sentences in the gold data set are indexed and then for each sentence, the top k similar sentences are retrieved from the index using the BM25 function. The pairs are then labeled using BERT. The drawback here is that BM25 is only useful for lexical similarity, not for semantic similarity.

The fourth is Semantic Search Sampling. SBERT is trained on the gold data set and for each sentence, the top K similar sentences are retrieved based on the cosine similarity of the query sentence and remaining sentences with embeddings derived from SBERT.

The last is a combination of the above methods. Oftentimes NLP models have to be empirically validated based on the given data set, so which method will work best is unknown until a set of possible options is exhausted.

Conclusion

NLP models have advanced quite rapidly over the past decade, as a researcher at Google published the Word2vec model in 2013, which was considered state of the art at the time. Since then, computational resources have continued to grow more available and at increasingly lower cost, with the major cloud providers such as AWS, GCP, and Azure providing access to a virtual server with multiple NVIDIA GPUs with just a few clicks. Even with the increase in computing resources, training NLP models is still extremely time consuming and expensive. Fortunately, large corporations such as Google are open sourcing their NLP research, which allows innovation such as the SBERT adaptation, developed by researchers at the Technische Universitat Darmstadt, to flourish. The SBERT methodology will likely prove to be extremely valuable to small and medium sized start-ups that do not have the resources or the time to devote to using BERT to its full potential. This is an interesting space to watch closely as more research is made widely available for anyone to improve upon.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” *arXiv preprint arXiv:1810.04805*

Nils Reimers and Iryna Gurevych. 2019. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks” *arXiv:1908.10084v1*

Nandan Thakur, Nils Reimers, Johannes Daxenberger and Iryna Gurevych. 2021. “Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks” *arXiv:2010.08240v2*

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need” *In Advances in Neural Information Processing Systems, pages 6000–6010*