Semantic Similarity Search

Josh Hedtke and Sergei Petrov

December 13, 2019

1 Introduction

At Gridspace (the company at which Josh Hedtke works), there are hundreds of billions of trascribed conversation segments holding relevant, searchable information related to customers' quality assurance and analytics goals. Given a query sentence or small paragraph q, semantic search seeks to find the top n most semantically similar sentences/paragraphs, some of which may have no or few tokens in common with q. To accomplish this, we generated BERT (Bidirectional Encoder Representations from Transformers) [2] embeddings $\in \mathbb{R}^{768}$ to represent sentences, much like the word2vec embeddings model [10]. However, BERT takes into account both left and right context of every word in the sentence to generate every word's embedding representation. For example, in the phrases "in the jail cell" and "mitochondria in the cell," the word "cell" would have very different BERT embedding representations in each of the sentences.

BERT beats all other models in major NLP test tasks [2]. We finetuned the pre-trained BERT model on a downstream, supervised sentence similarity task using two different open source datasets. We constructed a linear layer that took as input the output of the BERT model and outputted logits predicting whether two hand-labeled sentences were similar or not. After finetuning, we took the $11th$ hidden layer from the BERT model as the embeddings for our sentences.

Surprisingly, the finetuned BERT embeddings performed worse than the base pre-trained BERT embeddings, which performed quite well. We also discovered that reducing the dimension of the BERT embeddings markedly improved their performance in returning relevant search results. This suggests that the raw BERT embeddings didn't generalize well to the validation set.

2 Related Work

The concept of a transformer, based only on attention mechanisms without usage of recurrent and convolutional layers was proposed in 2017 by Vaswani, A. et al. [3]. The original BERT technique, accompanied by a set of pre-trained models, was released in 2018 by Devlin, J et al. [5]. After the release, researchers started to investigate its various applications and possible modification. The most extensive and widely used repository we worked with is the Huggingface-transformers [7], in which different modifications of BERT are implemented. There are two solutions we came across that are designed to calculate sentence similarity using transformers. One of them does not accept custom models for embeddings calculation, so it could not be used for our purpose. The other solution, called BERT as a Service [8], we utilized to calculate embeddings using the bert-base model along with its fine-tuned versions. Some works related to BERT fine-tuning approaches were also released, from which one of the most detailed is by Sun, C. et al. [4] describes different fine-tuning strategies and hyperparameters.

3 Dataset and Features

3.1 Datasets

The validation dataset consisted of 89 pairs of sentences which by design had similarity 1: *they said before i had a visa, they told me I had a visa earlier.* Another dataset was constructed pairing the first sentences from the original validation dataset with unrelated sentences, what gave us a set of pairs with similarity 0: *they said before i had a visa , in this section we demonstrate the workflow by training and testing a model for salt body interpretation.* Two finetuned models were trained on public datasets MRPC and STS-B. Both datasets have sentence pairs and similarity labels, for STS-B initial similarity scores on a scale of 5 were converted into binary labels. MRPC: *0 Rudder was most recently senior vice president for the Developer & Platform Evangelism Business. Senior Vice President Eric Rudder, formerly head of the Developer and Platform Evangelism unit, will lead the new entity.* STS-B: *3.800 A man is playing a large flute. A man is playing a flute.*

3.2 Features

BERT tokenizes the input sequence adding a [CLS] token for classification tasks and [SEP] tokens to mark the end of each sentence. Each token is encoded with its corresponding embedding vector of a length 768, what results in a matrix of a size $\#\_of\_tokens \times 768$. During the training process, embeddings are initialized randomly and adjusted in the process of training. In the evaluation mode embedding representations for tokens are "looked up" in a learned dictionary. To account for absolute and relative positions of tokens positional encoding vectors are calculated as sine and cosine functions and added to the matrix of vectorized tokens. If input comes in pairs, sentence embeddings are added to differentiate between sentences. We added 0 and 1 to each token in the first and the second sentence respectively. As all sentences were padded to be of the same length, attention mask was added to define which tokens are real and which are padding tokens that should not be taken into account. A resulting matrix then goes to the attention layer and follows the process described in 4.2.

4 Methods

4.1 Embeddings Baseline

For our baseline, we embedded words in $\mathbb{R}^{300}$ word2vec vectors. We then generated sentence/paragraph embeddings by averaging the embeddings of all the tokens in the sentence/paragraph.

4.2 Pre-trained BERT Model

We employed the transfer learning paradigm by starting with a pre-trained BERT model [2]. In particular, we used the Bert base model which has twelve transformer layers, twelve attention heads at each layer, and hidden representations h of each input token where $h \in \mathbb{R}^{768}$. BERT learns a representation of each token in an input sentence that takes account of both the left and right context of that token in the sentence. BERT masks 15% of input tokens and predicts them. It uses self-attention in a transformer stack to do this [3]. Given an input of X of tokenized and embedded sentences, a single attention layer learns three matrices that project X into three subspaces, Q, K, and V, which are used to calculate how much "attention" each token should pay to every other token in learning its representation. Let $x^{(i)}$ be a single token input and $q^{(i)}$, $k^{(i)}$, and $v^{(i)}$ be its associated vectors. An attention score is calculated between $x^{(i)}$ and $x^{(k)} : k \in \{0, 1, ..., n\}$, where $n =$ tokenized sequence length, by computing $\langle q^{(i)}, k^{(k)} \rangle$. The score vector is then normalized via softmax, dotted with each $v^{(i)}$, and summed to get a vector representing how much attention should be paid to each kth word in representing the ith word.

That is a single attention head in a single encoder layer. There are six encoders that feed into six decoders. The decoder is the same as the encoder except there is an extra sub-layer that takes in input from final K and V matrices outputted by the final encoder layer. The decoder then outputs a sentence and attends only to the left outputted tokens as it does so. The outputs of the final decoder are projected into a higher dimensional logits vector, usually representing words in a vocabulary. The outputs are used to predict the masked words as well as on a supervised next sentence prediction task.

The contextual embedding representation h for each token is usually extracted from one of the final layers. In our case, the eleventh.

4.3 Finetuning on Sentence Similarity Tasks

We finetuned a BERT pretrained model called BERT base uncased from the google bert repo [5]. Taking the authors' advice, we finetuned the model on two supervised classification tasks. First, we used one of the Google Bert modules [5] to finetune on the MRPC dataset with binary labels 0 for not paraphrase and 1 for paraphrase. The module uses a hidden layer $h \in \mathbb{R}^{768}$ that outputs binary logits which are then passed through softmax. We matrix multiplty the representation of the $[CLS]$ token outputted by the pre-trained model by the weight matrix of the hidden layer. We use the authors' recommended hyperparameters, including a batch size of 32, an initial learning rate of $5e^{-5}$, 3 training epochs, and a 0.1 dropout probability. We changed the max sequence length from 128 to 256.

We also finetuned on the STS-B dataset using our own custom module written in pytorch inspired by Chris McCormick's implementation [6] and using the HuggingFace transformers library. We used a similar procedure, except that we took the

output from the $11th$ layer of the pretrained BERT model as input to a single linear layer with a sigmoid activation function and we used a max sequence length of 128. We discretized the labels l_i as follows:

$$\forall i, f(l_i) = \begin{cases} 0 & l_i < 4.0 \\ 1 & l_i >= 4.0 \end{cases}$$

And we used the standard binary cross entropy loss function, with batch size b

$$BCE = \frac{1}{b}\sum_{i=1}^{b} f(l_i)\log(\widehat{f(l_i)}) + (1 - f(l_i))\log(1 - \widehat{f(l_i)})$$

4.4 Extracting the Contextualized Embeddings

After finetuning, we extracted pooled embeddings to represent sentences in $\mathbb{R}^{768}$. To do this, we did mean pooling on the $11th$ finetuned layer. In other words, we averaged the hidden vectors for each of the tokens from the $11th$ layer. Devlin et al. suggest that the second to last hidden layer (the $11th$), concatenation of the last four hidden layers, and a weighted sum of the last four hidden layers give the best results for a Named Entity Recognition task. So we went with the $11th$ layer for our sentence similarity task. We used a program called bert-as-service built by Han Xiao [8] to generate these contextualized embeddings.

5 Results

5.1 BERT Finetuning

We recorded validation and training accuracy for both finetuning runs.

Model	Train Accuracy	Validation Accuracy
Finetuned on MRPC		0.858
Finetuned on STS-B	0.817	0.836

5.2 Search Accuracy on Validation Set

Given a query embedding q, we searched the validation set V for matching embeddings v_i in hopes of matching the embedding hand-labeled as matching q, call it m. We used two metrics: "top5" and "top1." We compared q with v_i using cosine similarity: $\cos\theta = \frac{q^T v_i}{||q||||v_i||}$. We sorted matches by cosine similarity. Call the set of top5 matches TF and the singleton set of top1 matches TO. Then we calculated $top5 = \sum_{i=1}^{n} \frac{1\{v_i \in TF\}}{n}$ and $top1 = \sum_{i=1}^{n} \frac{1\{v_i \in TO\}}{n}$.

Model	Top1 Accuracy	Top5 Accuracy
Baseline	0.831	0.876
Bert Base	0.843	0.921
Finetuned on MRPC	0.640	0.809
Finetuned on STS-B	0.719	0.786

5.3 Dimension Reduction

We wondered whether we had overfit the fine tuned models and whether we could reduce the dimensionality of the embedding space to make search faster, so we explored using Principal Components Analysis (PCA). We found that the search accuracy increased markedly after reducing the dimension of the embeddings:

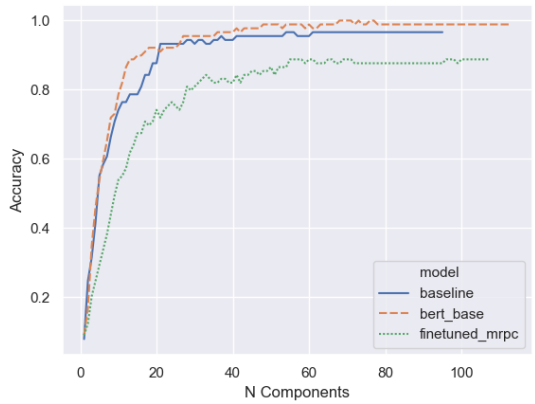

Figure 1: Top5 Accuracy

Model	N Components	Top5 Accuracy
Baseline	95	0.966
Bert Base	113	0.988
Finetuned on MRPC	107	0.888

Figure 2: Top 5 Accuracy Improvements

We also investigated how much of the semantic information was lost by reducing dimension.

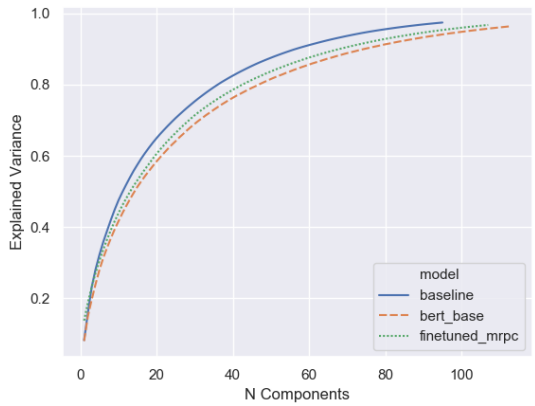

Figure 3: Explained Variance on Validation Set

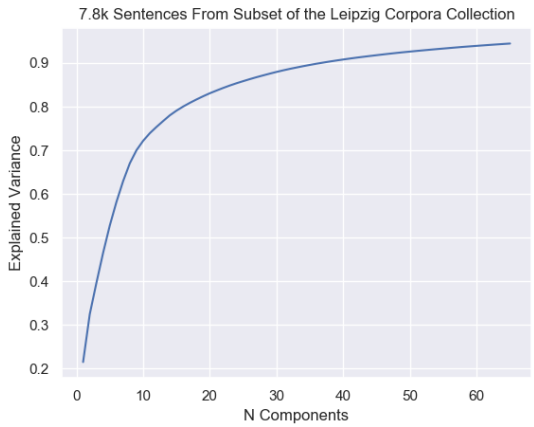

Figure 4: Explained Variance On Larger Dataset

Reducing the dimension increased accuracy across the board, suggesting that our hypothesis of overfitting embedding representations of size d on a validation of set n with $d > n$ may be true.

Reducing dimension from 768 to $\approx$ 100 preserved ninety-five percent of the information in the validation set embeddings. This effect held for a much larger set–$7.8k$ randomly sampled sentences from a set of $78k$ sentences from the Leipzig Corpora Collection [9]. The preservation of information was not a quirk of the small size and similarity of the set of validation sentences.

5.4 K-means clustering

We experimented with the K-means clustering algorithm to identify patterns in sentences embeddings calculated with different models. The dataset with 0 similarity pairs was used to perform division into 2 clusters. Separation of the data into two semantically different groups was successful only with embeddings of the baseline and BERT base models. Also, the original validation dataset with similar pairs was separated into 4 clusters. Though results obtained with BERT base embeddings are hardly interpretable, they are quite representative in terms of how a length of a sentence affects its embedding. Sentences were obviously clusterd according to their lenght, i.e. to number of tokens in a sentence.

1	I can definitely help you and just to make sure i've got the right account what is your full name; What is your income either per year or every month or bimonthly whatever way is easiest; They say they won't take an email but i'm just gonna forward it to them will you send that to me; Hey i need to make sure my insurances are listed on the new address i just changed usaa just notified me to call in
2	Batching doesn't start until midnight so it might not be done processing yet; We could transfer money into my usaa checking account when i need it; I can't hear you let me turn the volume up; Will i see the automatic payments plugged in already when i go online
3	They told me I had a visa earlier; I did it online before; Last time i did it online; How may I help you
4	Um and then once you get it uh let me know if you want to open it and let me know that it looks good for ya uhm so that way you don't have to call back; For my car loan tried to go through today uhm there wasn't enough money in in the account so i'm currently showing negative uhm in the account and the transaction is still pending; [laughter] not a problem sir it happens to me all the time so i know how it goes if you need any help from us sir feel free to give us a call back okay; I like talking to people like you who aren't with usaa so i can explain what'll happen 'cause some people don't get it as readily as you did

Table 1: Clusters obtained with BERT base embeddings

6 Conclusion

Our finetuning experiments did not do well. We hypothesize that the training sets we used had sentences labeled as dissimilar that we would have coded as similar given the knowledge we have about our domain and validation set. Defining the notion of "similarity" is critical to any supervised training task in this domain. We hope to mitigate this problem by training on a larger set of sentences taken from the same domain as our validation set. We will try to generate labeled sentences in an unsupervised fashion by clustering them, then labeling sentences from the same cluster as similar and sentences from the furthest apart two clusters as dissimilar.

We would like to further explore the minimum dimensionality required to capture all the semantically meaningful information in a sentence in a large, high variance dataset.

And finally, we will explore other methods for generating representations of sentences, including the Universal Sentence Encoder models [11].

7 Contributions

Sergei obtained fixed length sentence embeddings using BERT as a Service and their cosine simlarity scores on the validation set. He performed analysis comparison with baseline. Sergei implemented the finetuning on the STS-B dataset and converted the model obtained by Josh from the pytorch to the tensorflow format. Sergei experimented with the K-means clustering on different models embeddings. He wrote second and third section of the report.

Josh implemented the baseline and set up the training environments for the finetuning runs. Josh finetuned on the MRPC dataset and tweaked and ran the program to finetune on the STS-B dataset. Josh wrote the bert_service.py program which calculated the metrics for the results. Josh did the PCA experiments and generated the figures for them.

Josh wrote the first, fourth, fifth, and sixth sections of the final report.

8 Links

https://github.com/j-hedtke/cs221-229

Here is the link to the cs221 project (of which Josh was a part):

https://www.overleaf.com/read/krxntvbbqbjh

References

[1] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. The SICK (Sentences Involving Compositional Knowledge) dataset for relatedness and entailment, May 2014.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[4] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? *Chinese Computational Linguistics*, page 194–206, 2019.

[5] https://github.com/google-research/bert.

[6] Chris McCormick and Nick Ryan. Bert fine-tuning tutorial with pytorch. 2019.

[7] https://github.com/huggingface/transformers.

[8] Han Xiao. https://github.com/hanxiao/bert-as-service.

[9] https://corpora.uni-leipzig.de/en?corpusid=eng_news_2016.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[11] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018.