CS584: Evaluating Modern Embedding Techniques for Semantic Search capabilities through the Classification of Reddit User Comments

Brenden Brusberg[1]
[1]Stevens Institute of Technology, bbrusber@stevens.edu

Abstract

Over the past decade, embedding techniques like BERT have gained a lot of popularity, however, there has been no clear consensus on which metric for evaluating an embedding's representation and their respective uses. Currently, there are many different evaluators for embeddings that can be categorized into two groups. Intrinsic evaluators, that measure how well an embedding represents the words in the model that creates the embedding. The other group is extrinsic evaluators that measure how well these embeddings can be used as input features for tasks outside the model that created the embeddings. The purpose of this final project is to look at how effective are the BERT and Sentence-BERT models at the extrinsic task of classifying Reddit comments to their Subreddits to effectively measure the representation of the documents for semantic search capabilities. This project will look into how well the pre-trained embeddings of BERT and S-BERT compare to BERT and S-BERT embeddings fine-tuned with additional layers on Reddit data. The expected results will allow us to compare the effectiveness of these models and determine how well the pre-trained embeddings are at generalizing to extrinsic tasks, in this case classifying Reddit comments, to determine the effectiveness of each encoding's representation.

1 Introduction

This project stems from the question: if I have a large corpus of text across many varied topics and I want to search for specific documents that are similar to a search query, how do I measure the accuracy of my model? Embeddings have enabled data scientists the ability to measure semantic similarity between words and with modern techniques they can even measure the similarity between sentences and even documents. For example, you have a survey with an open response question with hundreds of thousands of responses and you want to find representative quotes or you are making a tool that can search for documents from your company that stretch past multiple years and projects. This approach works by encoding your corpus of documents to embeddings, creating a vector space that enables the data scientist or user with the ability to find similar documents to an inputted document using cosine similarity. The question this project wants to answer is how to measure the quality of this vector space on its ability to represent the documents in a semantic search? This project proposes to use an extrinsic evaluation of embeddings via classification based on cosine similarity. Thus, if your embedding space provides a quality representation of your documents, then based on cosine similarity, you can classify a document based on a set of extrinsic evaluation labels.

The reason why pre-trained embeddings are so powerful for these tasks for semantic/elastic search capabilities is that what happens if you are looking for something the training set does not contain. Embeddings with co-similarity would enable the user to find the next closest search. Moreover, embeddings offer a very computationally inexpensive and fast capability to classify/find related searches. This specific classification problem will be on Reddit comment data which has a wide and varied set of mutually inclusive and exclusive communities. Thus, the same classification model will be used for each embedding technique to minimize the effect on the classification to focus on the ability of the embeddings to produce document vectors.

2 Background/Related Work

As previously stated in the abstract, there are two ways to evaluate embeddings. One being intrinsic evaluators and the other being extrinsic evaluators. Intrinsic evaluators are described as "Methods of intrinsic evaluation are experiments in which word embeddings are compared with human judgments on word relations." Whereas extrinsic evaluators are described as "based on the ability of word embeddings to be used as the feature vectors of supervised machine learning algorithms (like Maximum Entropy Model) used in one of various downstream *Bakarov created A survey of word embeddings evaluation methods* [3] . There is no one way to evaluate the quality of word embeddings. Many papers break it down by the task the embeddings are created for. However, embeddings in transfer learning can be applied to tasks they weren't originally trained on. Common problems happen when evaluating word embeddings, let alone document embeddings. One of the largest problems is the ambiguity and obscureness of the notion of semantics. Semantics is a subjective and emergent structure over many language users over time. Overall, there is a lack of properly labeled data, especially, for cosine similarity which can only be found in a few STS data sets. Moreover, there is not a lot of correlation between intrinsic evaluators and extrinsic evaluators. Q-VEC scores look to tackle this problem but are only evaluated on word embeddings and have not been applied to sentence or document embeddings. Even if one could produce a correlation between intrinsic and extrinsic evaluators there is an extreme lack of proven significance tests when evaluating embeddings. Moreover, there are inherent problems to embeddings that revolved around the hubness problem where certain embeddings for frequent terms seem to center at a cluster of other embeddings which can interfere with classification or downstream tasks.

In this project, we aim to use an extrinsic evaluation of popular pre-trained embeddings in their ability to perform a classification task on Reddit data to measure the quality of the embeddings generated. In 2018, *Bakarov created A survey of word embeddings evaluation methods* [3] which outlined possible ways of intrinsic and extrinsic evaluators for embeddings. Of which, text classification which will be used in this project was outlined. For this classification task, we will be looking at document embeddings where each comment is a document rather than word embeddings. In order to specifically look at the quality of embeddings for semantic search, which primarily uses co-sine similarity, we will be using KNN which will use (1 - cosine similarity score). For an overview on document embeddings and their techniques, refer to this article [7].

For evaluating word embeddings *Evaluating word embedding models: methods and experimental results* [16] is a paper that summarizes more modern internal and extrinsic evaluators as of 2019. Earlier word embedding evaluators of note [10][14].For evaluating sentence embeddings, examples of extrinsic evaluators can be found in *A Fine-grained analysis of sentence embeddings using auxiliary prediction tasks* [1]. For a more modern approach and examples of both extrinsic and intrinsic emulators please refer to *A simple but tough-to-beat baseline for sentence embeddings* [2] which goes over how their research group evaluated their approach to sentence embeddings over previously existing embeddings. Embeddings and their generated vector representations create a vector space in which word/sentence/-document representation is unique to the embedding layers that generated them. Therefore, comparing embeddings and their evaluators is not a direct apples-to-apples comparison. *Evaluation of word vector representations by subspace alignment*[15] breaches the gap between evaluating different embeddings directly through using sub Space Alignment. They proposed a QVEC—a score to score the intrinsic evaluation of an embedding which the paper shows to have a high correlation of how different models compared in extrinsic evaluation.

3 Experimental Design

In this work, Google Colab Pro will be used as a primary provider for a GPU and compute recourse. I upgraded to Google Colab Pro and upgraded Google Drive to a 2 TB tier to house all data and ensure constant GPU run time. Python 3.7 was used on the Google Colab notebook as well as running Jupyter notebook on my local machine a 16-inch with 2.3GHz 8-core Intel Core i9, Turbo Boost up to 4.8GHz, with 16MB shared L3 cache with an AMD Radeon Pro 5500M with 4GB of GDDR6 memory and automatic graphics switching. Google Colab uses GPU: GPU Architecture is a Tesla P100-PCIe-16GB NVIDIA Pascal, CUDA Cores 3584, Double-Precision Performance 4.7 TeraFLOPS, Half-Precision Performance 18.7 TeraFLOPS, GPU Memory 16GB VRAM, CPU: 1xsingle core hyperthreaded Xeon Processors @2.3Ghz i.e(1 core, 2 threads). A TPU instance is used for evaluating all the embeddings as once as it has up to 35GB of ram, but the TPU is not used. This can be diverted by just evaluating each type of encoding seperately.

I will split the data into 3 sets. Training data representing 44,000 documents (4,000 documents per class) and 11,000 documents for testing (1,000 documents per class). Of the remaining 390,000 documents scrubbed will be sued for fine-tuning encoding models. Each of the 4 encoding models will clean and tokenize data on their own, encode both the train and test sets separately. I will then evaluate the embeddings produced by eye through PCA and then through a KNN classifier using (1 - co-sine similarity scores)

3.1 Data Set

Reddit comments are responses to a post that is posted on a Subreddit. These comments will generally talk about the post that pertains to the Subreddit. Thus, these comments can be treated as documents that can be labeled to a Subreddit. The Reddit comments used in this experiment are specifically from November of 2019 as this was the smallest downloadable file available containing Reddit comment data with over 500GB of Reddit data compressed.[18]. Reddit comment data proposes a very unique Natural Language Programming challenge as it spans many languages, cultures, and online communities whom all different ways to spell, abbreviate and communicate. The most recent data set was updated in 2017. Each row of the data contains one row in JSON containing the information of one comment. The most important features are Subreddit ID/Name and comment body. There are 1.2 million categories/Subreddits over 1.7 billion documents/comments in the data. However, the categories are heavily unbalanced. Only roughly 138,000 of those 1.2 million subeditors are active (sufficiently large) at the time of the creation of the dataset. Therefore, I will only be looking at Subreddits that meet a sufficient amount of comments that also meet a certain length. I will only be taking comments that are longer than 6 tokens and less than 500 from these Subreddits: [r/WritingPrompts, r/history, r/Jokes, r/Fitness, r/legaladvice, r/Music, r/space, r/food, r/gaming, r/wallstreetbets, r/politics]. The reason why I want to take comments with more than 6 tokens is that I do not want to classify short responses and want to have the data set only be composed of complete sentences or thoughts. I do not want to go over 512 tokens as the input to an out-of-the-box BERT model is limited to 512 tokens. 97 percent of the documents in our training data can be represented in 200 tokens. In total, I tried taking at least 50,000 comments from each class. I ended up with 446,105 comments where every class at least has 20k comments. I randomly pulled 5k comments from each class to use as a train and test set for the KNN classifier. The rest of the documents I saved for fine-tuning BERT and S-BERT.

{"subreddit":"<sub_reddit_name>", "body":"<comment>"}

3.2 Encoding Methods

The following encoding methods were used on the training and testing sets to create document embeddings for each document/comment. These encoded documents will be used as training and testing data for the classifier and visual analysis.

3.2.1 BERT

BERT provides contextual embeddings via its hidden layers. 'bert-base-uncased' was used as a base model where each of its 12 hidden layers (each containing 768 features) can provide a useful contextual embedding. Huggingface's transformer library was used in conjunction with Torch and TensforFlow. There are many different ways to use the hidden layers to create a final embedding representation of the input. You can use the first, last, or any of the hidden layers on their own. You can also do combinations of averaging, summing, and concatenating the hidden layer outputs. In the end, the best representation I found was by concatenating the last four hidden layers to make a one contextualized embedding with 3072 features.

The input to this model of BERT only takes in 512 tokens, however, our specialized vocabulary is not entirely contained in the 'bert-base-uncased' tokenizer. The witty way the tokenizer and BERT model gets around unseen vocabulary is by break unknown tokens into sub-tokens using another NLP model. However, since I want to use this model as a baseline of BERT's ability to transfer its learning into contextualized embeddings, the tokenizer will create out at 500 token zero-padded vectors for input.

3.2.2 Sentence-BERT

Sentence Transformers python library was used for this project and was developed by the creators of the originating paper of Sentence-BERT. Sentence transformer provides pretrained siamese and triplet encoded networks on top of a BERT derivative model. S-BERT is evaluated on STS tasks and transfer learning tasks. I am using 'roberta-large-nli-stsb-mean-tokens' which produces embeddings with 1024 dimensions. I picked this model because it has the best performance of co-sine similarity for similar data sets that involve similar short sentence pairs. This model is trained with CosineSimilarityLoss.

The way I encoded the documents was by breaking the documents into sentences using NLTK Punkt's sentence tokenizer. I fed each sentence into 'roberta-large-nli-stsb-mean-tokens' S-BERT and then average all sentence embeddings per document to create a document embedding. This assumes that each sentence is independent thought and the contextualized embeddings of each sentence will not be based on possible sentences before or after. Therefore, the averaging is a mean representation of each independent sentence.

3.2.3 Reddit-BERT

I added 961 new vocab pulled from finding the top TF-IDF terms from each class that showed up at least 60 times. The reason for 60 is to capture the most used words and most important words. 60 set as the min count per class provided us with 1,016 terms to add to BERT. Since the model only allowed 980 new vocab words to the 'bert-base-uncased' model without changing the original weights for the existing tokens, I took out terms in the new extended vocabulary that did not make sense to vocab. Some of these words can be found here:

- r/music: audioslave, grunge, metallica, soundgarden, spotify, synthpop,
- r/wsb: bearish, bezos, brokerage, bruh, drizzle, payout, reimbursement, stonks, tendies, robinhood, aapl, stonks , wsb
- r/gaming: bethesda, cutscene, ftl, fortnite, gta, n64, overwatch, pve, pvp, quests, rdr2, remaster, tf2, witcher, zelda, minecraft, ps1, ps', ps3, ps4, ps5
- r/politics: biden, bipartisan, boomer, buttigieg, dems, potus,
- r/fitness: caffeine, calisthenics, overtraining, pullup, pulldown, underweight,
- r/food: caramel, carrots, celery, cilantro, broccoli, mcdonalds, meats, overcooked, pickle, sushi, taco,
- r/space: spacex, spaceflight, supermassive, supernova,
- r/legaladvice: suing
- r/writingprompts: acronyms, abbreviations
- r/Jokes: , 'punchline', 'puns'
- r/history: 'ww1', 'ww2'

All 961 new terms are expressed in the Jupyter notebook.

Now that I added the new vocab I have to re-tokenize the data set. In this case, because I have most of the tokens I want to especially represent, and be able to fit all training data in 16GB of VRAM, I will represent 97 percent of the training data by using the max amount of tokens to be 200 while padding the rest with 0's. During this tokenization, I now want to use the generated attention mask that I ignored during the creation of the baseline BERT model. Using the GLUE sequence classification layer on top and using the existing labels, an additional 5 epochs of training, batch size 32, a validation set of 10 percent, resulting in 401,494 training samples 44,611 validation samples from a set of extra documents I pulled from the data. Using a Huggingface adam fixed weight decay optimizer with a learning rate of 3e-5, adam epsilon 1e-8 (adam epsilon is the default parameter for transformer tuning, however, I decreased the learning rate to account for the smaller training data size) After training, the validation set reached an accuracy close to 80 percent and can be used and shows that a linear layer on top of the pooled output layer of the 12 hidden layers used to classify is what I want to try to approach in terms of accuracy. However, I just used this accuracy to determine if the training was effectively tuning the parameters to update the extended vocabulary and not measure how good the embedding is. The document embeddings were again created by concatenating the last four hidden layers of the model.

3.2.4 Reddit-Sentence-BERT

I am using the newly created 'Reddit-BERT' as a tokenizer and contextualized embedding input into S-BERT. Training data for S-BERT is formatted as a combination of SNLI and MultiNLI, which is referred to as AllNLI. Each entry to the training set consists of a sentence pair(Sentence A (Premise) Sentence B (Hypothesis)) and a label. I manually labeled 500 sentence pairs on a scale from 0 as a contradiction, 1 as neutral, and 2 as entailment. I then standardized this to be between 0-1 to match cosine similarity scores. To fine-tune S-BERT you ideally want to train a separate bi-encoder on the manually labeled data set otherwise known as the gold data set. Then you can use this new bi-encoder to label the rest of the sentence pairs in your training set between 0-1 to create a silver data set. Then you would ideally fine-tune your model inputting both the gold and silver data set. Since I was only able to label 500 sentence pairs, this eventually left the silver data set to overfitting.

However, the gold training data set is fine on its own. Fine-tuning BERT is done through Cosine Similarity Loss over a Siamese network where each side of the model has input sentence A or B into the selected BERT model. A pooling layer is added to both sides where it is fed into the final layer which calculates the cosine similarity of the output of each pooling layer. I selected 4 epochs with a batch size of 16 because I do not have a large amount of data. The hyper-parameters for S-BERT are best handled by the Sentence-Transformer package, especially, on such a small gold data set.

3.3 Classification

At this point I have two data sets composing of 4,000 and 1,000 rows respectively from each label, each row contains its id, the Subreddit label, the comment body, BERT ids, BERT document vector, S-BERT document vector, Reddit-BERT-ids, Reddit-BERT document vector, and finally Reddit-S-BERT document vector. A KNN classifier that looks at (1 – cosine similarity score) for distance will be fitted on the BERT model's encoding and K will be parameter tuned to give the best results. The best K for the base BERT model turned out to be 25. I kept k at 25 for the other 3 KNN classifiers so I can compare the classifier as much as possible based on their ability to classify based solely on (1 – cosine similarity score).

3.4 Evaluation

The encoded embeddings and their ability to perform in a semantic search will be evaluated and compared by their respective KNN's ability to classify the documents to their Subreddit label. This performance will be measured by the KNN's multi-class ROC-AUC score on classifying comments to their Sub-Reddit. Accuracy, precision, recall, and f1 score will also be used to access the models while training and testing as supporting metrics. The higher the ROC-AUC score of the classification models, the better the embeddings are at generalizing to Reddit comment data and representing the embeddings in a semantic search scenario.

Moreover, visual analysis through principal component analysis can show how well the embeddings are at representing the documents through showing clusters of documents that are separated by class. However, this should only be used as an inference, as reducing these large vector spaces with 1024 and 3072 dimensions down to 2 will lose a lot of meanigful data.

Referring to figure 1, the 'bert-base-uncased' model is clearly struggling with the additional unseen vocab. The projection created from S-BERT is relatively better than BERT, but not great. The classes are clustered but there is no clear separation. Reddit-BERT looks great, there is clear separation, however, there is room for improvement between separating the classes and creating tighter clusters. Reddit-Sentence-BERT takes the cake between these four models, however, it can be better.

4 Experimental Results

Quite clearly we can see that the naive approach for encoding with BERT on domain-specific knowledge like Reddit struggles with new vocabulary. The performative scores other than Roc Auc are not good. The Roc Auc is not the worst, yet, it can be but can be largely improved. However, holistically looking at the embeddings I can determine that 'bert-base-uncased' is not an adequate model to use for encoding Reddit comments. Manually testing semantic search, I can clearly see that the results are not semantically similar to the manual input. Looking at S-BERT we can see huge improvements, especially in Roc Auc. However, this model is nowhere close to being used as a classifier, however, the underlying embeddings seem to prove somewhat useful in semantic search looking at results by hand (simply encoding a sentence I give the model and looking at the top N closest documents).

The largest jump in performance is given by fine-tuning Reddit-BERT. There is a clear large jump in the KNN's ability to classify documents base on co-sine similarity scores that this model could realistically be used for classification. Yet when testing semantic search by hand and the results are generally good only shows the need for this evaluation. If I can not determine how well the results of the semantic search are through testing manually inputted documents then the KNN's ability to classify should be used instead. However, Reddit-S-BERT is the best model tested and even out performed the validation set used while training Reddit-BERT.

	Accuracy	Roc Auc Score	Precision Score	Recall Score	F1 Score
BERT	0.3615	0.6488	0.4021	0.3615	0.3688
S-BERT	0.5798	0.7689	0.6055	0.5798	0.5771
Reddit-BERT	0.7426	0.8585	0.7758	0.7426	0.7513
Reddit-S-BERT	0.8244	0.9034	0.8459	0.8244	0.8282

5 Conclusion and Future Work

The naive attempt at encoding documents with BERT subverted my expectations by wildly under performing. The effect of BERT under performing when encountering unknown domain-specific vocabulary has been well-documented [6] [12] [17]. The Naive approach with BERT most likely has too large of an input length leading to sparsely zero-padded inputs. It is clear that fine-tuning the base line models improved metrics tremendously. Fine-tuning the 'bert-

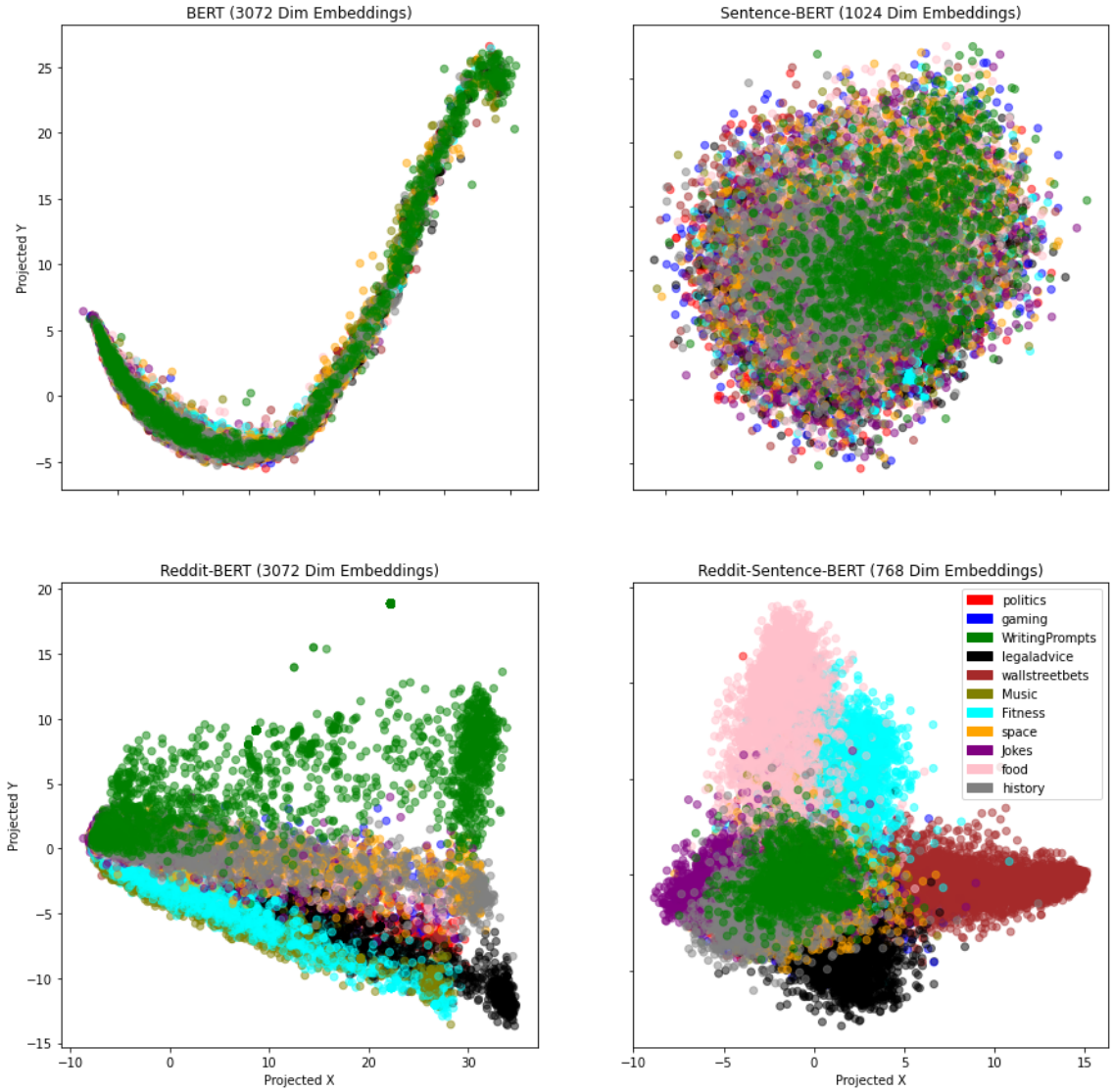

Figure 1: PCA Dimensionality Reduction to 2 Dimension for all Encodings

base-uncased' model was the most beneficial jump in performance for the amount of work. It may take more training time, yet, will produce better embeddings that can later be used by Reddit-S-BERT for even better performance. S-BERT, although had the highest preforming metrics, it is a difficult model to implement with the manual labeling of data and the risk of over fitting from generating a silver data set Also to note, dimension size is not a determining feature in embedding quality, the best encoding method had 768 dimensions whereas the worst model has 3076.

In terms of evaluating embeddings' ability to search PCA is a great and fast way to identify if your embeddings are performing well. A KNN is better at evaluating the quality of service or tool that uses cosine similarity to semantically search for similar documents. Especially, when documents returned in a semantic search may look good by the eye, a KNN can help determine which results are more representative of your training data set.Creating better embeddings for classification using cosine similarity will create better embeddings for semantic search. To improve embedding representation, data cleaning and tokenization are crucial. Especially with Reddit comments that can often be very messy unstructured data. To better improve S-BERT, creating a bi-encoder with a larger, more representative, gold data set would massively help fine-tune S-BERT by providing a silver data set [13].

Future work for improving the evaluation of document embeddings includes developing Q-VEC for sentence/document embeddings that will enable a score that correlates to performance across different embedding spaces. For example, BERT uncased has a hidden layer size of 768, we created an embedding space was 3096, S-BERT was 1024.

It is hard alone to directly compare them when their parameters are turned on different data sets, but having different dimension sizes can discard any straight forward way of comparing the embeddings by themselves. One could look at transforming embedding spaces onto each other for a more direct comparison. Another way to improve the evaluation of embeddings would look at using angular distance/angular similarity for the KNN instead of (1 - cosine similarity score) for better standardization of embeddings.

References

[1] Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., Goldberg, Y. (2016). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. arXiv preprint arXiv:1608.04207.

[2] Arora, S., Liang, Y., Ma, T. (2016). A simple but tough-to-beat baseline for sentence embeddings.

[3] Bakarov, A. (2018). A survey of word embeddings evaluation methods. arXiv preprint arXiv:1801.09536.

[4] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[5] Faruqui, M., Tsvetkov, Y., Rastogi, P., Dyer, C. (2016). Problems with evaluation of word embeddings using word similarity tasks. arXiv preprint arXiv:1605.02276.

[6] Müller, M., Salathé, M., Kummervold, P. E. (2020). Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter. arXiv preprint arXiv:2005.07503.

[7] Palachy, S. (2019). Document Embedding Techniques. https://www.kdnuggets.com/2019/10/beyond-word-embedding-document-embedding.html

[8] Pennington, J., Socher, R., Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[9] Reimers, N., Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084.

[10] Schnabel, T., Labutov, I., Mimno, D., Joachims, T. (2015, September). Evaluation methods for unsupervised word embeddings. In Proceedings of the 2015 conference on empirical methods in natural language processing (pp. 298-307).

[11] Sia, S., Dalmia, A., Mielke, S. J. (2020). Tired of Topic Models? Clusters of Pretrained Word Embeddings Make for Fast and Good Topics too!. arXiv preprint arXiv:2004.14914.

[12] Tai, W., Kung, H. T., Dong, X. L., Comiter, M., Kuo, C. F. (2020, November). exBERT: Extending Pre-trained Models with Domain-specific Vocabulary Under Constrained Training Resources. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings (pp. 1433-1439).

[13] Thakur, N., Reimers, N., Daxenberger, J., Gurevych, I. (2020). Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks. arXiv preprint arXiv:2010.08240.

[14] Tsvetkov, Y., Faruqui, M., Dyer, C. (2016). Correlation-based intrinsic evaluation of word vector representations. arXiv preprint arXiv:1606.06710.

[15] Tsvetkov, Y., Faruqui, M., Ling, W., Lample, G., Dyer, C. (2015, September). Evaluation of word vector representations by subspace alignment. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 2049-2054).

[16] Wang, B., Wang, A., Chen, F., Wang, Y., Kuo, C. C. J. (2019). Evaluating word embedding models: methods and experimental results. APSIPA transactions on signal and information processing, 8.

[17] Wolf, T. (2019). Some additional experiments extending the tech report" Assessing BERTs syntactic abilities" by Yoav Goldberg. Technical report.

[18] Link to Data: https://files.pushshift.io/reddit/comments/