Clustering BERT Embeddings for Classification in Sentiment Analysis via Dot Product (CBERTdp)

Thomas Vecchiato
880038@stud.unive.it

Riccardo Zuliani
875532@stud.unive.it

Alice Schirrmeister
1000371@stud.unive.it

Isabel Marie Ritter
1000095@stud.unive.it

1 Introduction

Neural Networks have demonstrated remarkable prowess in achieving State-of-the-art results across diverse domains. However, their efficacy comes at the cost of computationally complex processes. Thus, our approach tries to enhance efficiency by redistributing the tasks to optimize the computational workload on Artificial Neural Networks (ANNs), integrating methods that involve less expensive computations. Specifically, we aim to simplify classification tasks by employing K-Means clustering and performing dot-product. As an example we make use of Sentiment Analysis on different kinds of reviews. We utilize a pretrained BERT (Devlin et al., 2019) model to obtain embeddings for the reviews, subsequently subjecting these to K-Means clustering and performing dot-product operations to obtain the final label. We chose Sentiment Analysis as an example due to its wide-ranging applications and inherent significance. By implementing our approach, we anticipate achieving not only satisfactory results in terms of accuracy but also developing a more resource-efficient mechanism for executing Sentiment Analysis.

2 Related work

Clustering BERT embedding is not a new idea, indeed much research has been conducted in this field. (Reimers et al., 2019) show the power of contextualized word embeddings to classify and cluster topic-dependent arguments, (Sia et al., 2020) provide benchmarks for the combination of different word embeddings and clustering algorithms analysing their performance under dimensionality reduction with PCA. Moreover (Eklund and Forsman, 2022) use the embeddings obtained by BERT together with UMAP dimensionality reduction and HDBSCAN clustering to model the topics. Other research mostly focuses on incorporating word embeddings into LDA framework like (Dieng et al., 2019) which develop a tool that discovers interpretables topics even with large vocabularies that include rare words and stop words. Continuing many others researches exist regarding only the field of topic modelling via BERT embeddings including (Palani et al., 2021), (Grootendorst, 2022) and (Atagün et al., 2021). Many correlated white papers cover the so called *prototype selection*. It is also important to point out that the following approach is strictly related with MIPS (Maximum Inner Product Search), fundamental topic in IR (Information Retrieval) and beyond.

The method we develop lay the foundations for the creation of new scalable classification strategy, that is computationally fast, with the usage of a small amount of memory and maintain the same accuracy of the best existing classifiers in literature. In the following project we focus on Sentiment Analysis but it is worth highlighting that this approach can be extended to many classification areas. Regarding the word embedding model, we take only the base BERT model into consideration to have a fair comparison between our approaches and the competitors, whereas for the clustering methodology we consider only K-Means due to better scalability.

3 Our approach

Our idea is based on the following methodology: as a first step, for each sentence or passage of our training set we execute BERT (Bidirectional Encoder Representations from Transformers) in order to obtain the embedding vector of that sentence. Once we have all the embeddings, we cluster these embeddings (using FAISS Spherical/Standard K-Means (Johnson et al., 2019)). The value of K for K-Means can vary and computational analyses will be carried out based on which is the best K, also considering the accuracy and the complexity. Each centroid of each cluster is labelled either as positive or as negative (preprocessing for Sentiment Analysis) using majority vote as a criterion. This last part concludes the offline phase which is necessary for us to assign a label when we have a new query. The second step is the online step involving a new sentence or passage. First of all, given the query we are going to execute the BERT model to get the vector representation. Once obtained, we compute the dot-product between the query and the centroids calculated in the offline step. In the case of two centroids as displayed in Figure 1, we assign to the query the label corresponding to the centroid with larger dot-product value. This idea will be extended to having multiple centroids where the final label is assigned according to majority vote. Using this technique we can obtain expressiveness and calculation speed whilst keeping the technique extremely scalable, because in the online phase it is only necessary to run BERT on a single query and a few dot-products to obtain the final solution. In this way, we are able to achieve faster results without employing another neural network for classification or having to use BERT several times. At the same time, using BERT to obtain the embeddings should preserve expressiveness and accuracy.

For further improvement of the model's accuracy, we propose a second approach that utilizes layer-wise embeddings of BERT for the benefit of greater expressiveness. Each layer captures the sentence's meaning at a different level of abstraction resulting in multiple embeddings that provide a richer representation of the meaning of the sentence. The second approach is in turn divided into two sub-projects: in the first one, we consider each layer-embedding separately and for each of them we perform what we have described in the previous approach. As regards the second sub-project, an extra attention layer is added to incorporate all the layer-wise embeddings given by BERT (*layer-aggregation*). In this variation we decided to fine-tune the model by freezing the BERT layers while allowing the newer attention layer to train its weights. This procedure aids to avoid the phenomenon of *Catastrophic Forgetting*. Another different solution is to fine-tune the complete models with a very low learning rate (*2e-5*), as suggested by (Sun et al., 2020) and (Simone, 2023). Continuing the whole model is done to achieve a better vector representation in the output that considers all the different nuances of the different layers in order to solve our problem optimally. Once having extracted the embedding, the remaining procedure is equal to the first approach.

Graphical Abstract Figure 1, 2 and 3 illustrate the pipeline we adopted and describe our idea.

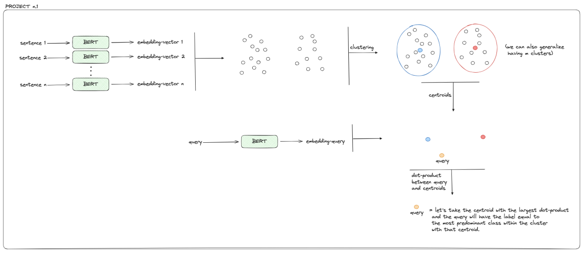
PROJECT n.1
sentence 1
sentence 2
sentence n
BERT
BERT
BERT
embedding-vector 1
embedding-vector 2
embedding-vector n
clustering
(we can also generalize having m clusters)
centroids
query
BERT
embedding-query
query
dot-product between query and centroids
query
= let's take the centroid with the largest dot-product and the query will have the label equal to the most predominant class within the cluster with that centroid.

Figure 1: Describes the main pipeline that we are going to use.

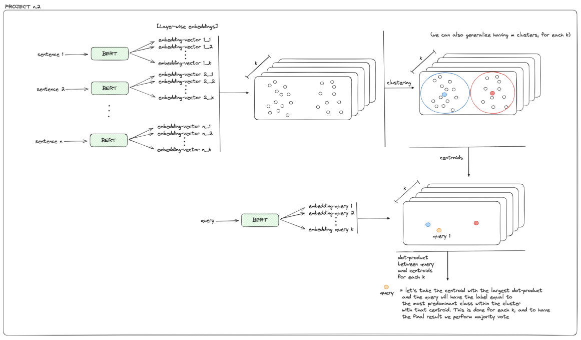

Figure 2: Describes the variation with BERT layer-wise embeddings for greater expressiveness.

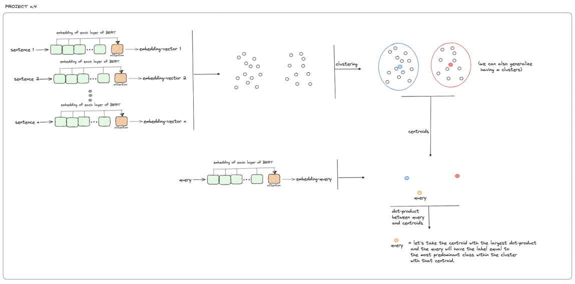
PROJECT n.4
embedding of each layer of BERT
sentence 1
attention
embedding-vector 1
embedding of each layer of BERT
sentence 2
attention
embedding-vector 2
embedding of each layer of BERT
sentence n
attention
embedding-vector n
clustering
(we can also generalize having m clusters)
centroids
embedding of each layer of BERT
query
attention
embedding-query
query
dot-product between query and centroids
query
= let's take the centroid with the largest dot-product and the query will have the label equal to the most predominant class within the cluster with that centroid.

Figure 3: Describes the second variation where layer-embedding concatenation is fed into a Attention layer.

What baseline algorithms will you use? The baseline algorithms that we are going to use are:

Naive baseline: predict the most common class.

Random choice baseline: predict a random label between the set of labels.

Machine learning baselines: Support Vector Machines, Naive Bayes and Logistic Regression, all of them with TF-IDF feature extractor.

3.1 Competitors

Regarding the competitors, we consider the pre-trained BERT model without applying any of our methods and employ a comparison using again BERT for word embedding with the difference that we feed the embedding into a GRU, an LSTM and a Bi-LSTM that have shown to be accurate in Sentiment Analysis tasks. Moreover, we will not only compare the effectiveness of the model but will also do a comparison of efficiency considering computation and memory costs.

3.2 Schedule

03/12: Project proposal.

03/12: Recovery and prepare all datasets for processing and perform the clustering.

08/12: Project review of other teams.

17/12: Poster presentation.

20/12: Idea project n.1 complete.

24/12: Idea project n.2-1 complete.

30/12: Idea project n.2-2 complete.

10/01: Gathering the results of evaluation and complete the part of the code.

16/01: End of the project with the final report.

By following this schedule, we aim to efficiently progress through each phase of the project, ensuring time for evaluation, comparison, and report writing. For a more efficient undertaking we intend to divide the workload of the different approaches between our group members to enable working in parallel.

4 Experiments

To assess the goodness of our clusters we utilize the confidence measurement as proposed by (Vascon et al., 2013) that represents the purity of a cluster. The evaluation of the three models will be done on the test set of each dataset (Section 6). That way we will have a clear estimation on how well each model performs and which is the best among those.

We will compare also the complexity of the different models and competitors to verify whether our approach is indeed beneficial regarding computational costs.

To evaluate the performance of our model, we will employ the **Accuracy Score**, **F1-Score**, **Precision** and **Recall**.

Finally, we will compare our model's performance with the baseline accuracy to evaluate its effectiveness. These metrics collectively provide a comprehensive evaluation of the Sentiment Analysis model, considering both accuracy and the balance between precision and recall.

5 Results

We anticipate that our different models will exhibit superior scalability compared to the competitors, leveraging matrix multiplications and elementary mathematical operations, consequently having a greater speed up. We expect that our approach not only manages to outperform other methods in terms of computational speed but also manages to maintain the same results considering the chosen metrics, aligning closely with the outcomes observed in the competitors.

6 Data

We intend to employ three different datasets available on HuggingFace, all containing reviews and being heavily used in Sentiment Analysis tasks:

IMDb[1] a dataset containing movie reviews for binary sentiment classification. *Labels: 0 Negative, 1 Positive*.

Stanford Sentiment Treebank[2] a dataset introduced by Pang and Lee (2005), consists of 11855 single sentences extracted from movie reviews. *Labels: 0 Negative, 1 Positive*.

Yelp Review Dataset[3] which consists of reviews from Yelp and is extracted from the Yelp Dataset Challenge 2015 data. *Preprocessing the labels in order to have: -1 Negative, 0 Neutral, 1 Positive*.

7 Tools

The tools that we plan to use are the following one:

Google Colaboratory to perform the training and the evaluation process on the available free GPU in order to speed up the computation.

HuggingFace to import the BERT model, its variations and the datasets.

[1] https://huggingface.co/datasets/imdb

[2] https://huggingface.co/datasets/sst2

[3] https://huggingface.co/datasets/yelp_review_full

Pytorch for the training and testing procedure and to manage the operations involving the embedding tensors.

Faiss (Johnson et al., 2019) a Meta library that provides clustering algorithms that are shown to be much scalable than *scikit-learn* implementations. We will use its K-Means implementation.

Matplotlib to plot the resulting scores of our models.

Note the required pre-processing procedure is done by the *BertTokenizer*[4] module of HuggingFace.

References

Ercan Atagün, Bengisu Hartoka, and Ahmet Albayrak. 2021. Topic modeling using lda and bert techniques: Teknofest example. In *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pages 660–664.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Adji B. Dieng, Francisco J. R. Ruiz, and David M. Blei. 2019. Topic modeling in embedding spaces.

Anton Eklund and Mona Forsman. 2022. Topic modeling by clustering language model embeddings: Human validation on an industry dataset. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, Abu Dhabi, UAE. Association for Computational Linguistics.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Sarojadevi Palani, Prabhu Rajagopal, and Sidharth Pancholi. 2021. T-bert – model for sentiment analysis of micro-blogs integrating topic model and bert.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and clustering of arguments with contextualized word embeddings.

Suzanna Sia, Ayush Dalmia, and Sabrina J. Mielke. 2020. Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too!

Innocente Simone. 2023. Sentiment analysis in context: Investigating the use of bert and other techniques for chatbot improvement. Available at `https://hdl.handle.net/20.500.12608/50232`.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification?

Sebastiano Vascon, Marco Cristani, Marcello Pelillo, and Vittorio Murino. 2013. Using dominant sets for k-nn prototype selection.

[4] https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertTokenizer