UNIVERSITÄT BONN

Problem Solving

Comparing Various Dimensionality Reduction Approaches For Embedding Vectors

Group 14
Behnam Fanitabasi, Mahnaz Mirhaj,
Behzad Shomali, Sven Knauer, Arwah Muhammad Jawwad

Contents

1 Introduction

In recent years the trend was to develop more powerful language models which often correlates to an increase in size. As different language models usually contain high dimensional embeddings this entails a number of challenges. Storing a high number of embeddings requires quite a lot of memory, while working with them requires significant amounts of time, energy, and computational resources, thus limiting the accessibility of this technology. To ease this impediment, it would be necessary to create embeddings with drastically fewer dimensions, that still provide (most of) the capabilities of the original large embedding.

In the current day of ever-increasing news, social media, and alike, there is an ever-growing flood of textual data that needs to be processed, to take advantage of its information. Besides enhancing these established kinds of usages, our proposed methods also could clear the way for new adaptations and use cases in the fields of the Internet of Things (IoT) and mobile, where time, memory, and energy constraints are particularly tight.

2 Data Exploration and Preprocessing

In order to compare the results of the language models with compressed embeddings we train them to perform a downstream task such as classification on different datasets. We use datasets with varying numbers of instances and distributions of the length of the documents with different languages for better generalization. Thus, our experiments are not dependent on only a specific dataset. For this purpose, we will be using the following three classification datasets to compare the results:

1. **Dataset: CheckThat Lab - Subjectivity Detection (Galassi et al. 2023):** The task is to predict whether a given sentence from a news article is subjective or objective. We choose the multilingual dataset consisting of Arabic, Dutch, English, German, Italian, and Turkish.
   The train set contains 6628 sentences,
   The test set contains 600 sentences,
   The data contains binary labels SUBJ or OBJ.
   The lengths of documents (number of the words and numbers) have a mean of 19, a max of 121, and min of 1. The mean of word frequencies is 3 with the most frequent word repeated 1400 times and the least one repeated only 1 time.

2. **Dataset: Aspect Sentiment Classification Dataset (Pontiki et al. 2014):** The aim is to classify the sentiment toward each (English) sentence as positive, negative, or neutral given the aspect and the review sentence.
   The train set contains 2163 sentences,
   The test set contains 638 sentences,
   Each entity has a classification label, 'polarity' with positive, negative, or neutral values.
   The lengths of documents (number of the words and numbers) have a mean of 17, a max of 59, and min of 2. The mean of word frequencies is 12 with the most frequent word repeated 2161 times and the least one repeated only 1 time.

3. **Dataset: FEVER (Fact Extraction and VERification) (Thorne et al. 2018):** The task is to predict whether a given (English) claim is supported or refuted by verifying the facts or whether notEnoughInfo is present to make a decision.
   The Train set consists of 145,449 sentences,
   The Test set consists of 19,998 sentences,
   The claims are classified as Supported, Refuted, or NotEnoughInfo.
   The lengths of documents (number of the words and numbers) have a mean of 8, a max of 55, and min of 2. The mean of word frequencies is 44 with the most frequent word repeated 49522 times and the least one repeated only 1 time.

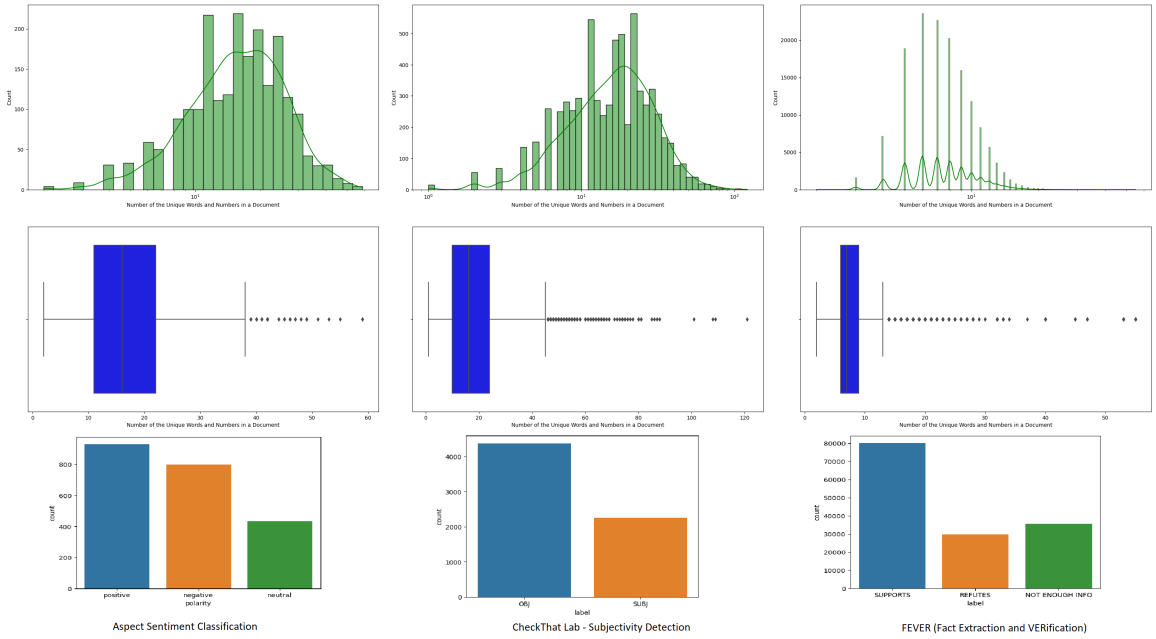

Figure 1: Distributions of the length of documents and different classes (labels) of the three datasets

3 Methodology

3.1 Embedding Space

- *Sentence-BERT. Pretrained Model* 2019: To create sentence embedding, we have used the paraphrase-multilingual-MiniLM-L12-v2 SBERT model. This pre-trained model is chosen due to available resources, task definition, and multilingual datasets. This model can keep the semantic information, is 5 times faster than the best SBERT model, and has been trained in over +50 languages. The dimension of the produced embeddings is 384.

- GloVe: To create word embeddings, we have used the pretrained GloVe vectors (Jeffrey P. and D.Manning 2014) to learn word representations. We use the pretrained vectors with 50 dimensions as given in glove.6b.50d.txt. The relatively low dimensionality strikes a balance between capturing meaningful semantic relationships and managing computational complexity.

3.2 Approach 1

A Variational Autoencoder (VAE) (Kingma and Welling 2019) is a generative model that combines autoencoder concepts with probabilistic modeling. Its goal is to learn a hidden representation of input data, capturing its underlying patterns in an unsupervised manner, enabling the generation of new, similar samples, and providing a more meaningful representation. VAE structure consists of an Encoder, Latent Space, and Decoder.

- Encoder: Input data is fed into an encoder neural network, which maps it to a distribution in the latent space, reducing dimensionality and deriving mean and variance parameters for a (multivariate Gaussian) distribution.

- Latent Space (Sampling): The ”reparameterization trick” combines the mean with the product of standard deviation and random noise to create a latent vector, enabling random sample generation.
- Decoder: The decoder network reconstructs input data by mapping the latent vector back to the original data space.

During training, a VAE optimizes two functions:

- Reconstruction Loss: Measures the dissimilarity between input and reconstructed data.
- Kullback-Leibler (KL) Divergence: Encourages the learned latent space distribution to approach a prior distribution. This regularization promotes desired properties in the latent space.

$$L(\phi,\theta,x) = \frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2 + KL[G(Z_\mu, Z_\sigma), \mathcal{N}(0,1)]$$

$$L_{KL}\left[G(Z_\mu, Z_\sigma)\|\mathcal{N}(0,1)\right] = -0.5\sum_{i=1}^{N}\left(1 + \log(Z_{\sigma_i}^2) - Z_{\mu_i}^2 - e^{\log(Z_{\sigma_i}^2)}\right)$$

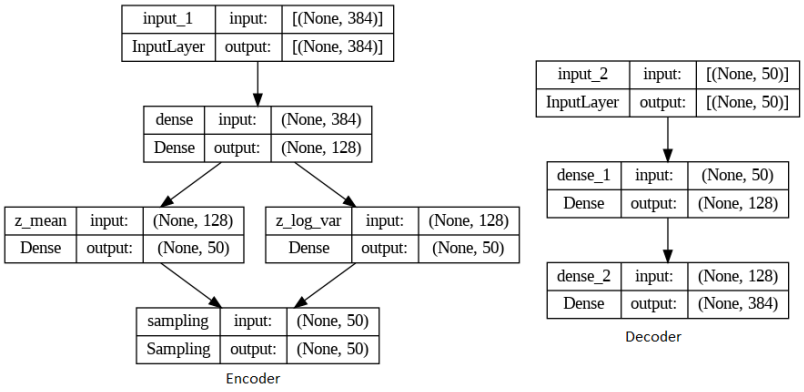

Figure 2: The architecture of the encoder and decoder of the Variational Autoencoder used in this project which takes the embeddings with dimension 384 and creates a probability distribution with latent vectors having dimension 50

3.3 Approach 2

In this approach, we tackled the challenge of dimensionality reduction using traditional methods such as Principal Component Analysis (PCA) (Hotelling 1933), t-distributed Stochastic Neighbor Embedding (t-SNE) (Van der Maaten and Hinton 2008), and Linear Discriminant Analysis (LDA) (Mika et al. 1999). For each algorithm, selecting the appropriate hyperparameters, particularly the number of components (n_components), was crucial. In this regard, we end up using values of 100 and 50 for PCA and t-SNE algorithms respectively. Furthermore, since the number of classes, while using LDA is limited to the number of available classes, we could only use at most values 2 or 3 in our experiments.

In this approach, we encountered unique challenges with t-SNE when n_components exceeded 3, leading us to forgo the Barnes-Hut approximation in favor of an exact computation method despite its higher computational complexity ($O(N^2)$ time compared to $O(NlogN)$ time for Barnes-Hut). To

address the computational burden during classifier training, we strategically used a reduced subset of 1000 training samples with t-SNE for the first two datasets, CheckThat Lab and Aspect, achieving a delicate balance between computational feasibility and embedding quality for the successful application of the technique. Furthermore, as the size of the FEVER dataset was huge, it was almost infeasible to apply t-SNE hence the results are not available (N/A).

3.4 Approach 3

In the paper (Ksh. Nareshkumar Singh 2022) an algorithm for dimensionality reduction using word embedding has been proposed. The approach presented in this paper utilizes the 'GloVe' word embedding technique to achieve dimensionality reduction by assessing the similarity scores obtained from tf-idf calculations between words. This process involves eliminating redundant features from the data. The pipeline for the system design starts of with calculating the document representation of the data using tf-idf followed by similarity detection using GloVe, dimensionality reduction and then performing a downstream classification task. We produced a similar approach on both word embedding obtained by GloVe and document embedding obtained by SBERT.

- Document Representation: The weight of each document vector is calculated using the term frequency-inverse document frequency (tf-idf) weighting scheme.
- Similarity Detection: In the case of GloVe the 50-dimensional pre-trained word vectors were used while in the case of the document embedding the multilingual SBERT model was used.
- Dimensionality Reduction: The paper proposes a . removal of redundant feature (rRF) algorithm. The objective is to identify synonym words by clustering word vectors based on their similarity. Words within the same cluster are considered similar or redundant features in text classification. In the used approach, each cluster is represented by the term with the highest sum of tf-idf over the entire corpus. This representation aids in removing redundant features while maintaining classification performance.

The proposed method calculates similarity scores between data and takes into account the non-linear relation between data. The tf-idf scores in case of SBERT are the mean tf-idf weights of all words in the sentence. In case of GloVe embeddings, a similar approach is used.

4 Experimental Setup and Evaluation

Using the aforementioned language models, we first calculate the embeddings for the documents (sentences/claims) of the three datasets. The proposed methodologies are then used to extract the compressed embeddings from the original ones. We use these compressed embeddings in classification tasks with various Machine Learning classifiers (Naive Bayes, Support Vector Machine, K-Nearest Neighbours, Random Forest, and Multilayer Perceptron) to evaluate how much information they can store. To define the classifiers, we make use of the Scikit-Learn (Pedregosa et al. 2011) module. The models' configurations are as follows:

- SVM with "rbf" kernel
- MLP with hidden layers (200, 100)
- KNN with K = 50
- Random Forest with the number of trees (estimators) equal to 100

Other hyper-parameters are set as Scikit-Learn default values. The test/development sets (original and compressed embeddings) are available on the web pages for each dataset, and these sets are used to evaluate the classifiers after they have been trained using the provided train sets (original and compressed

embeddings). The accuracy, precision, recall, F1-score, inference time (the runtime of classifiers using various embeddings), and memory consumption (the amount of memory required to store the embeddings) are calculated and compared for each dataset.

5 Evaluation and Results

The Appendix contains charts of the results (f1-score, time inference, memory consumption) for several classifiers and datasets. The classifiers' results for the various embeddings are presented in the tables below. Runtime is reported in seconds and memory usage is in bytes.

Table 1: Performance Metrics for Different Embeddings on CheckThat Lab

Embeddings	SVM		MLP		KNN		Naive Bayes		Random Forest		Memory
	f1	time	f1	time	f1	time	f1	time	f1	time	
Original	0.72	5.208	0.69	8.948	0.64	0.101	0.65	0.014	0.64	9.791	9162240
VAE	0.71	2.202	0.82	12.583	0.62	0.059	0.61	0.008	0.73	5.059	1325600
rRF	0.68	0.878	0.68	2.788	0.62	0.08	-	-	0.61	2.082	4041216
PCA	0.72	2.028	0.67	5.517	0.65	0.03	0.63	0.008	0.6	5.155	2386000
LDA	0.7	0.699	0.71	3.358	0.72	0.016	0.7	0.005	0.63	0.798	47720
t-SNE	0.4	0.108	0.5	3.2	0.41	0.0158	0.25	0.002	0.49	0.5	200000

Table 2: Performance Metrics for Different Embeddings on Aspect Sentiment Classification

Embeddings	SVM		MLP		KNN		Naive Bayes		Random Forest		Memory
	f1	time	f1	time	f1	time	f1	time	f1	time	
Original	0.61	0.673	0.63	2.281	0.51	0.027	0.66	0.006	0.48	2.681	3322368
VAE	0.59	0.272	0.61	6.760	0.48	0.041	0.6	0.006	0.49	1.779	432600
rRF	0.67	0.25	0.67	2.83	0.56	0.023	-	-	0.65	1.45	3230208
PCA	0.62	0.32	0.61	4.11	0.51	0.018	0.53	0.002	0.48	1.72	865200
LDA	0.61	0.080	0.63	2.64	0.63	0.013	0.64	0.001	0.64	0.25	34608
t-SNE	0.25	0.095	0.30	2.097	0.32	0.014	0.26	0.001	0.39	0.491	200000

Table 3: Performance Metrics for Different Embeddings on FEVER

Embeddings	SVM		MLP		KNN		Naive Bayes		Random Forest		Memory
	f1	time	f1	time	f1	time	f1	time	f1	time	
Original	0.48	11077.61	0.49	1082.99	0.38	11.832	0.48	0.241	0.38	312.84	223409664
VAE	0.39	4648.72	0.44	833.13	0.35	3.229	0.35	0.182	0.35	158.65	29089800
rRF	0.49	4.055	0.49	3.425	0.47	0.084	-	-	0.45	4.209	6865920
PCA	0.45	4921.21	0.47	727.78	0.37	3.819	0.43	0.0711	0.38	170.12	58179600
LDA	0.43	861.86	0.44	54.272	0.45	0.296	0.44	0.013	0.45	23.1	2327184
t-SNE	-	-	-	-	-	-	-	-	-	-	-

6 Analysis and Discussion

After completing the pipeline, VAE consistently exhibited lower memory consumption compared to the original embeddings across all tested datasets (Fig. 10-12). In terms of inference time, the VAE mostly outperformed the original embeddings, with the exception of certain cases e.g. the MLP approach (Fig. 7-9). Regarding the F-score, the VAE approach demonstrated strong performance. Interestingly, the VAE approach outperformed the original embeddings in certain scenarios, such as the random forest approach for the CheckThatLab dataset (Fig. 4-6).

The primary challenge is tuning the hyperparameter, particularly in finding a balance between the KL-divergence loss and reconstruction loss. To tackle this, we undertook a series of experiments and used a beta parameter to have a trade-off between the two errors.

After applying the various methods in the approach 2 section, the results demonstrate that LDA emerged as one of the top-performing techniques in terms of both f1-score and memory consumption. This outcome was somewhat expected, given LDA's supervised nature in dimensionality reduction, which utilizes class labels to find discriminative features. However, it is still surprising that even with the reduction of a 300-dimensional vector into a 2-3 dimensional vector through a supervised approach, the representation remains highly representative and retains meaningful information.

Additionally, our analysis of PCA's cumulative variance ratio plotted against the number of principal components revealed no distinct elbow point (Fig. 3). This observation implies that there is no intrinsic number of dimensions in the original embedding, making the dimensionality reduction process more delicate.

The approach 3 using the rRF algorithm also showed lower memory consumption. The F-score was comparable to the other approaches used except for the FEVER dataset as that dataset is a challenging test bed.

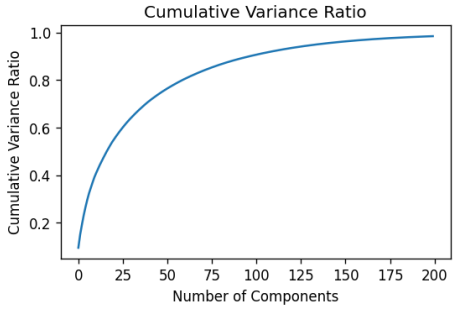

Figure 3: Example of cumulative variance ratio plot for the Aspect dataset

7 Conclusion

In this project, we examined various dimensionality reduction approaches for embedding vectors. The investigated approaches (1) Variational Autoencoder (VAE), (2) traditional methods, and (3) a novel approach using document embeddings along with 'GloVe' word embedding and assessing the similarity scores obtained from tf-idf calculations between words, have been trained to perform downstream tasks like classification on different datasets, avoiding dependency on a specific dataset. Experimental results show that the f-measure of the proposed approaches performs similarly to the baseline (constructed with the full dimensionality embeddings) in terms of classification performance. The various approaches did not outperform the original embedding but showed marginal losses in the given classification tasks.

While some showed quite parallel performance to the original, showing just a minor degradation in classification abroad all datasets, some methods clearly prefer a certain dataset or task. There was a strong correlation between performance and inference time/memory consumption as can be seen in Figures 7 and 8 in relation to Fig. 4-6. The methods working with smaller dimensionality have a much quicker inference time, approximately two orders of magnitude less, than the bigger investigated approaches (Fig. 7-9), but can not compete with their f1-scores. On the other hand, also the bigger approaches could most of the time significantly reduce memory consumption during classification. In most cases, memory consumption could be at least reduced by over 50 percent. In general, this class of performance can be achieved by different methods, while offering a big gain in inference time as well as memory consumption, which is especially important when targeting bigger datasets like FEVER. In summary, the experiments confirmed our assumption that it is indeed possible to reduce the dimensionality of embedding vectors substantially while preserving much of their potential. Contrary to our initial expectations, diminishing dimensionality did not lead to a steep decline in performance; instead, it exhibited a gradual impact. While conducting our investigation we also observe the dependency of the classification performance on many factors: chosen classification algorithms, using different textual representation methods, feature selection methods, dimension reduction techniques, various (hyper-)parameters on the chosen methods, etc.

8 Future Works

In the future, a fascinating and promising field of study could involve using compressed and limited precision representations such as quantization in combination with dimensionality reduction techniques to achieve even smaller embeddings. The goal would be to reduce the size of data representations while maintaining their useful information. One potential area of exploration is to develop algorithms that automatically determine the appropriate levels of post-processing for these compressed representations, optimizing for performance in downstream applications.

The exciting challenge lies in developing algorithms that can automatically determine the optimal levels of post-processing for these compressed embeddings. This task involves finding a balance between the reduction in size and the preservation of critical information to maintain good performance in downstream applications. Such algorithms would simplify the dimensionality reduction pipeline for other tasks and make it easier to use these compressed embeddings effectively in real-world applications.

References

Galassi, Andrea et al. (2023). "Overview of the CLEF-2023 CheckThat! Lab Task 2 on Subjectivity in News Articles". In.

Hotelling, Harold (1933). "Analysis of a complex of statistical variables into principal components." In: *Journal of educational psychology* 24.6, p. 417.

Jeffrey P., Richard S. and Christopher D.Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Association for Computational Linguistics*, pp. 1532–1543. DOI: http://dx.doi.org/10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

Kingma, Diederik P. and Max Welling (2019). "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4, pp. 307–392. DOI: 10.1561/2200000056. URL: https://doi.org/10.1561%2F2200000056.

Ksh. Nareshkumar Singh S. Dickeeta Devi, et al. (2022). "A novel approach for dimension reduction using word embedding: An enhanced text classification approach". In: *International Journal of Information Management Data Insights* 2 (1). DOI: 10.1016/j.jjimei.2022.100061. URL: https://doi.org/10.1016/j.jjimei.2022.100061.

Mika, Sebastian et al. (1999). "Fisher discriminant analysis with kernels". In: *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*. Ieee, pp. 41–48.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pontiki, Maria et al. (Aug. 2014). "SemEval-2014 Task 4: Aspect Based Sentiment Analysis". In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, pp. 27–35. DOI: 10.3115/v1/S14-2004. URL: https://aclanthology.org/S14-2004.

Sentence-BERT. Pretrained Model (2019). Retrieved from https://www.sbert.net/docs/pretrained_models.html.

Thorne, James et al. (2018). "FEVER: a Large-scale Dataset for Fact Extraction and VERification". In: *ArXiv* abs/1803.05355. URL: https://api.semanticscholar.org/CorpusID:4711425.

Van der Maaten, Laurens and Geoffrey Hinton (2008). "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11.

9 Appendix

In the figures presented in the Appendix section, the notation provides information about the configurations that were used in the experiments. Specifically, the notation indicates the classifier used, along with its settings.

- MLP (.) corresponds to the Multi-Layer Perceptron classifier, and the value inside the parentheses indicates the size of the hidden layers.
- SVM- corresponds to the Support Vector Machine algorithm, and the notation further specifies the kernel function used.
- RandomForest tree is denoted as RandomForest-, and the notation further represents the number of estimators, which corresponds to the number of trees in the forest.
- KNN (K-nearest neighbor) is represented as KNN-, and the notation further indicates the number of nearest neighbors considered for classification.

After specifying the classifier, the type of dimensionality reduction technique used is also mentioned. If nothing is mentioned, it means that the results correspond to the original vectors. Aside from that:

- VAE corresponds to approach 1.
- PCA, t-SNE, and LDA correspond to approach 2.
- rRF corresponds to approach 3.

By employing this consistent notation, the report ensures clarity and allows readers to understand the specific configurations and dimensionality reduction approaches utilized in the experiments.

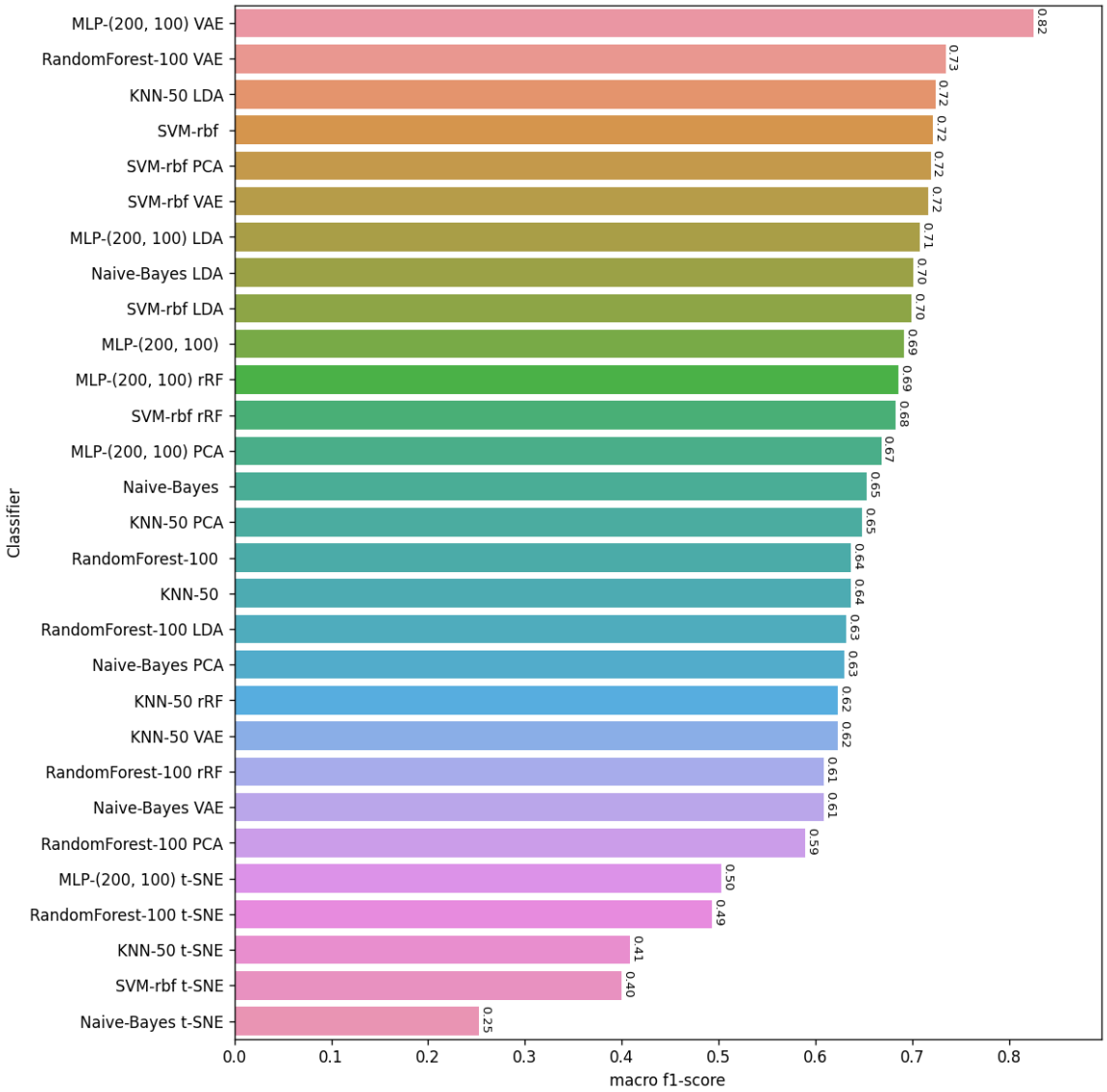

Figure 4: Comparison of different classifiers' performance in terms of macro f1-scores for the CheckThatLab dataset

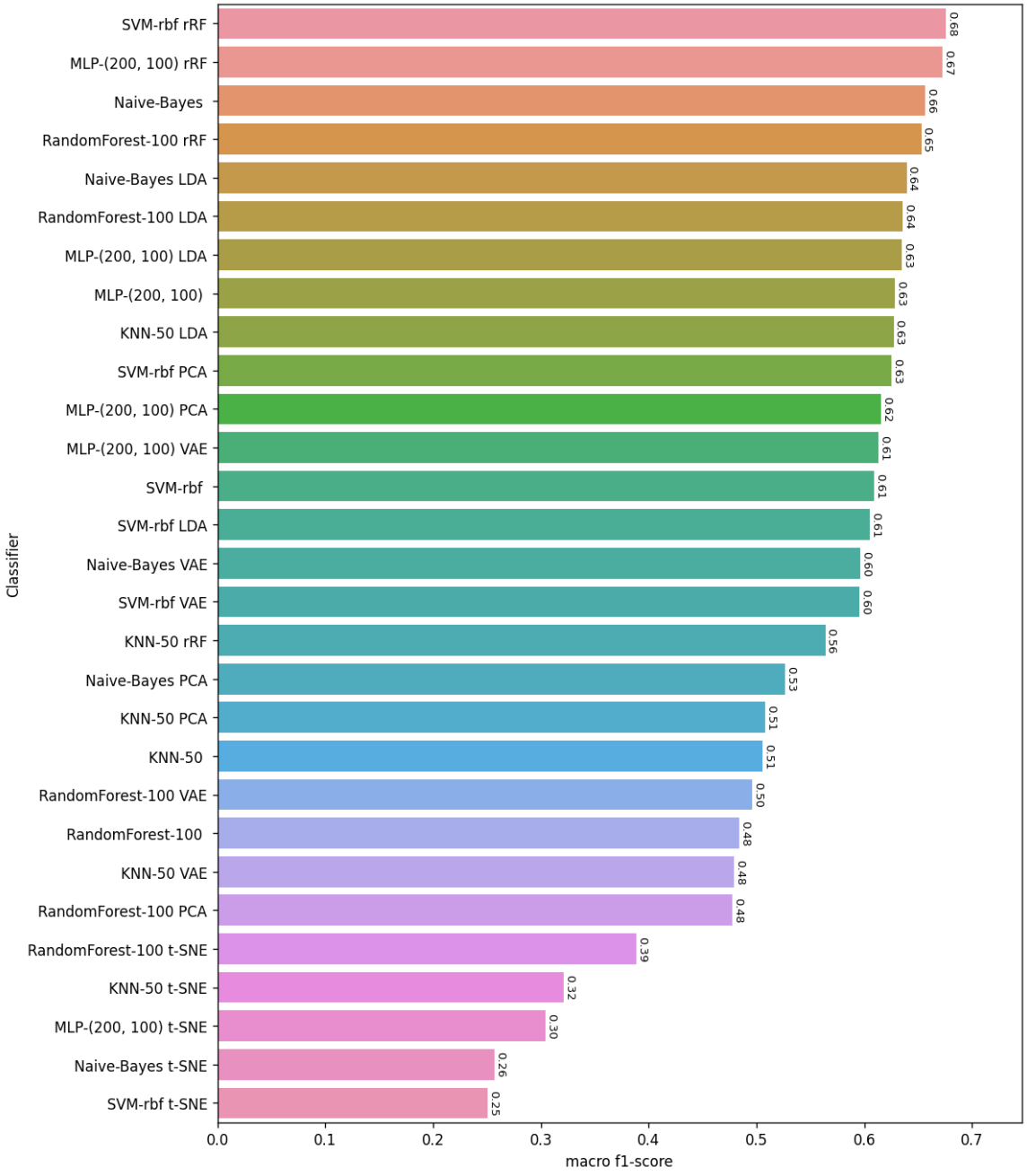

Figure 5: Comparison of different classifiers' performance in terms of macro f1-scores for the Aspect Sentiment Classification dataset

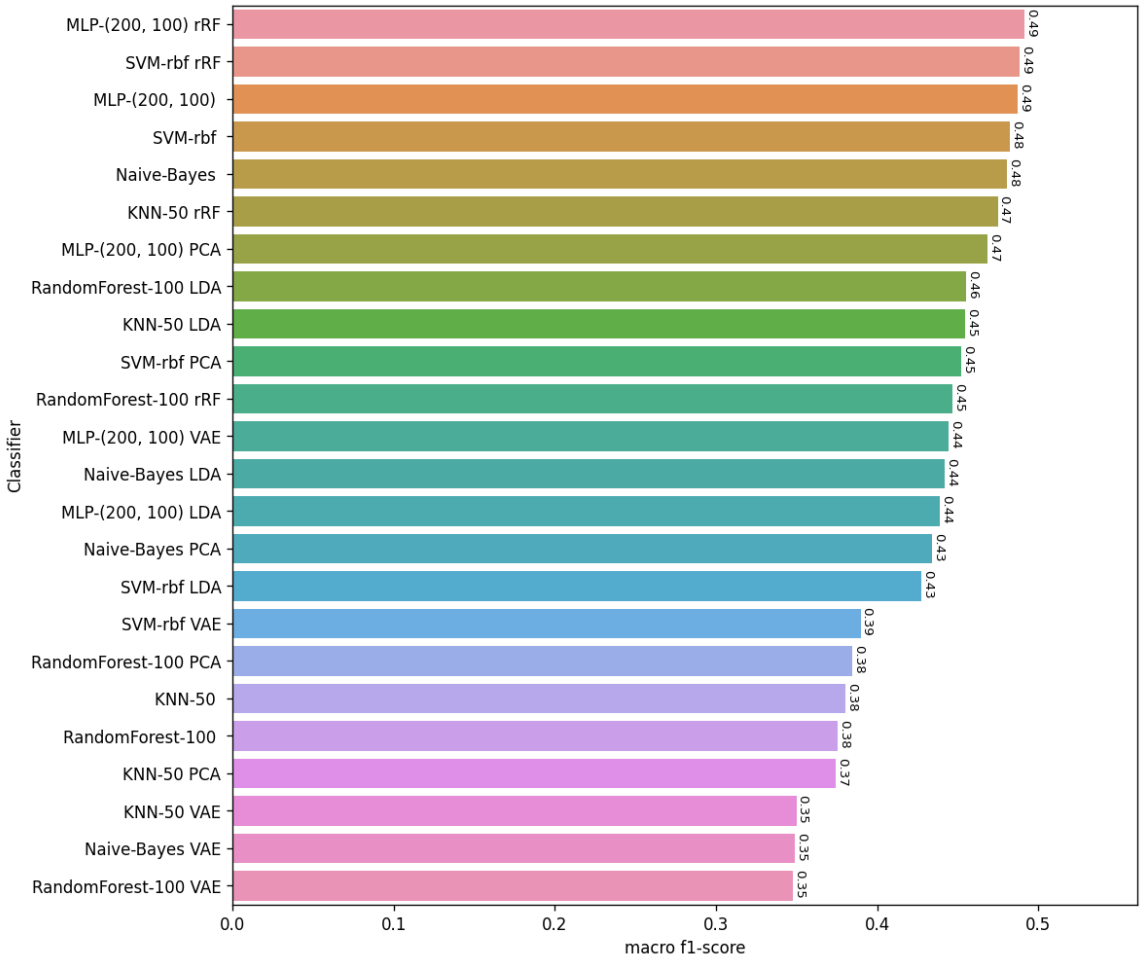

Figure 6: Comparison of different classifiers' performance in terms of macro f1-scores for the FEVER dataset

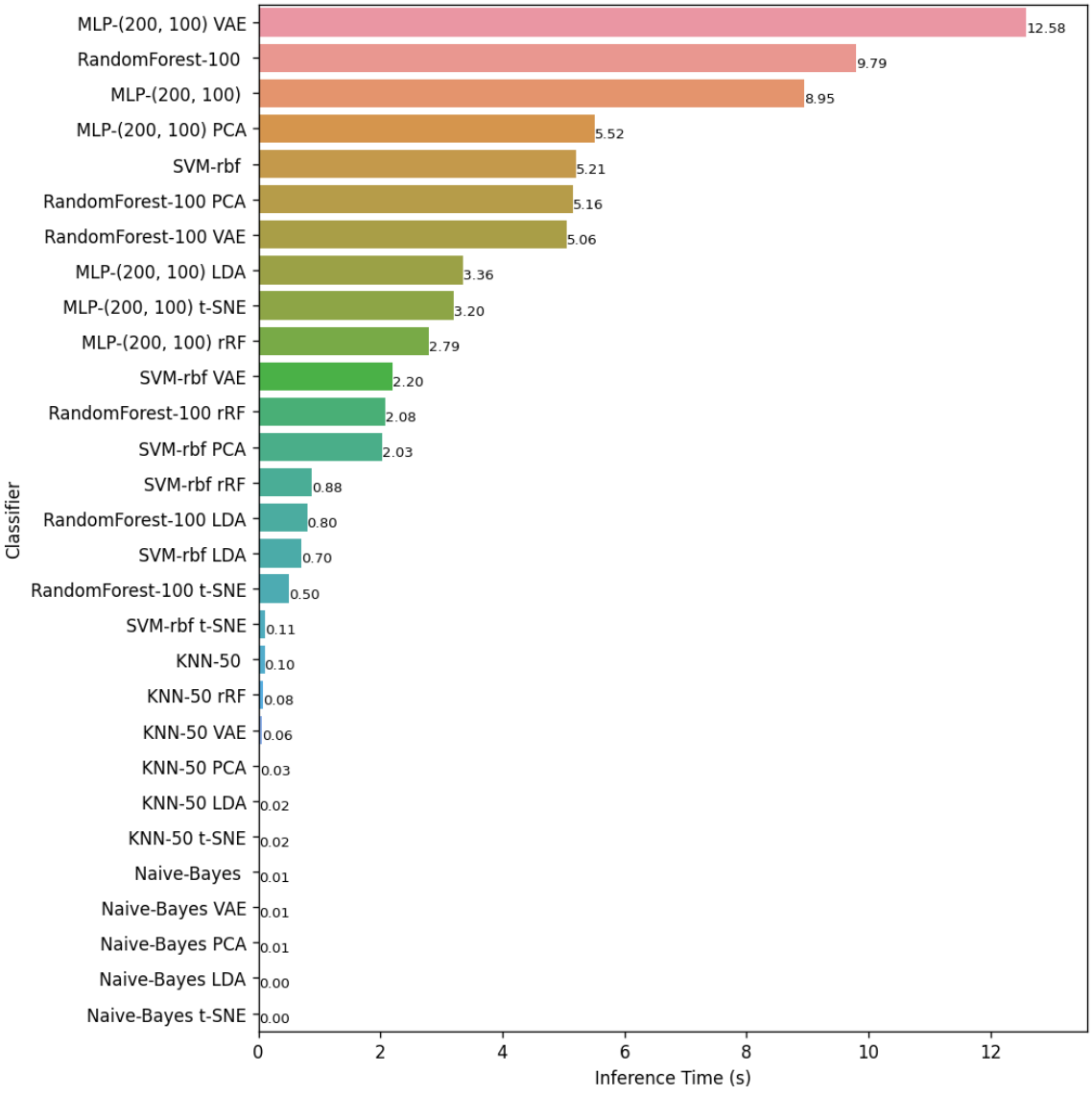

Figure 7: Comparison of inference time of different classifiers for the CheckThatLab dataset

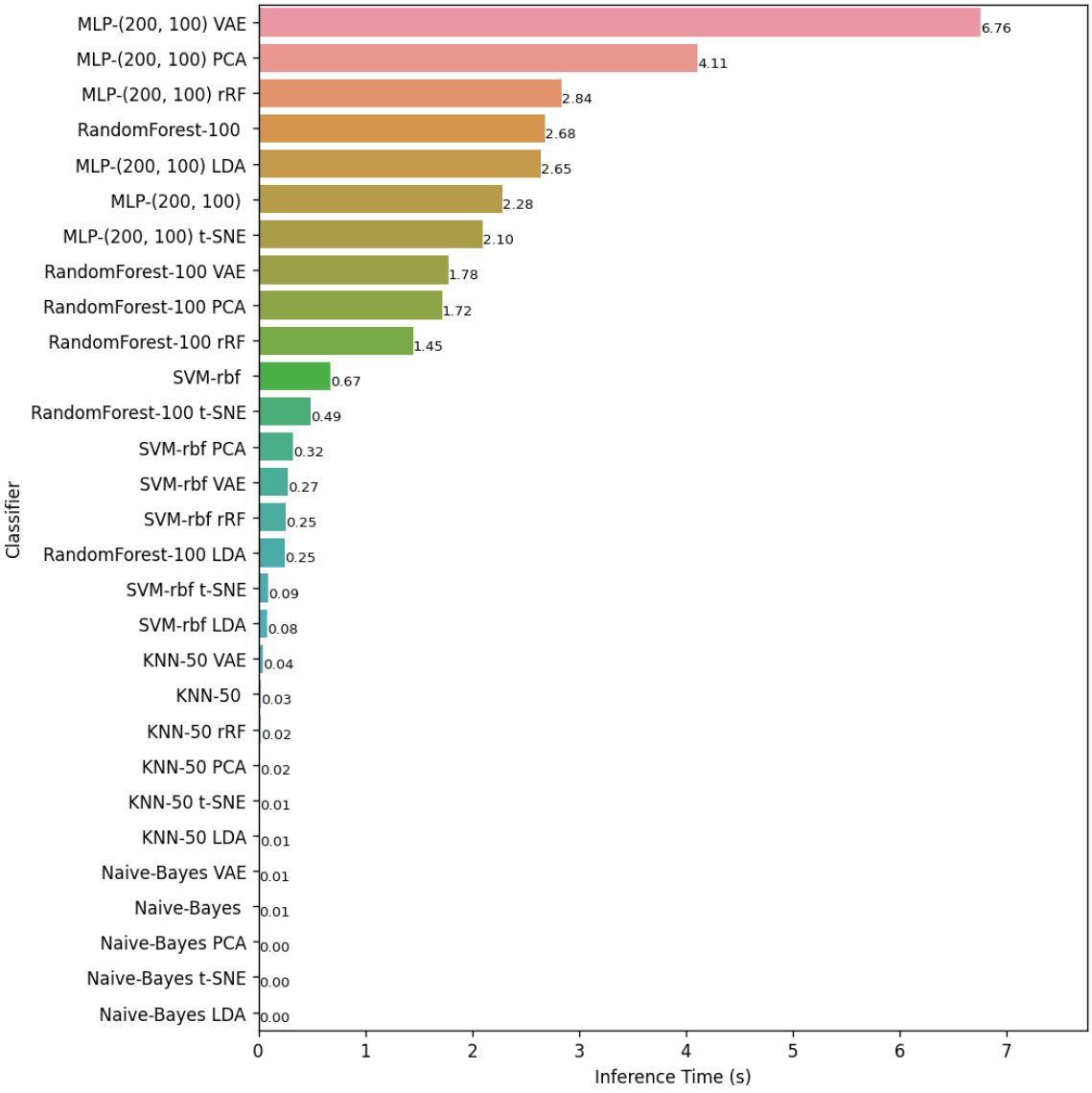

Figure 8: Comparison of inference time of different classifiers for the Aspect Sentiment Classification dataset

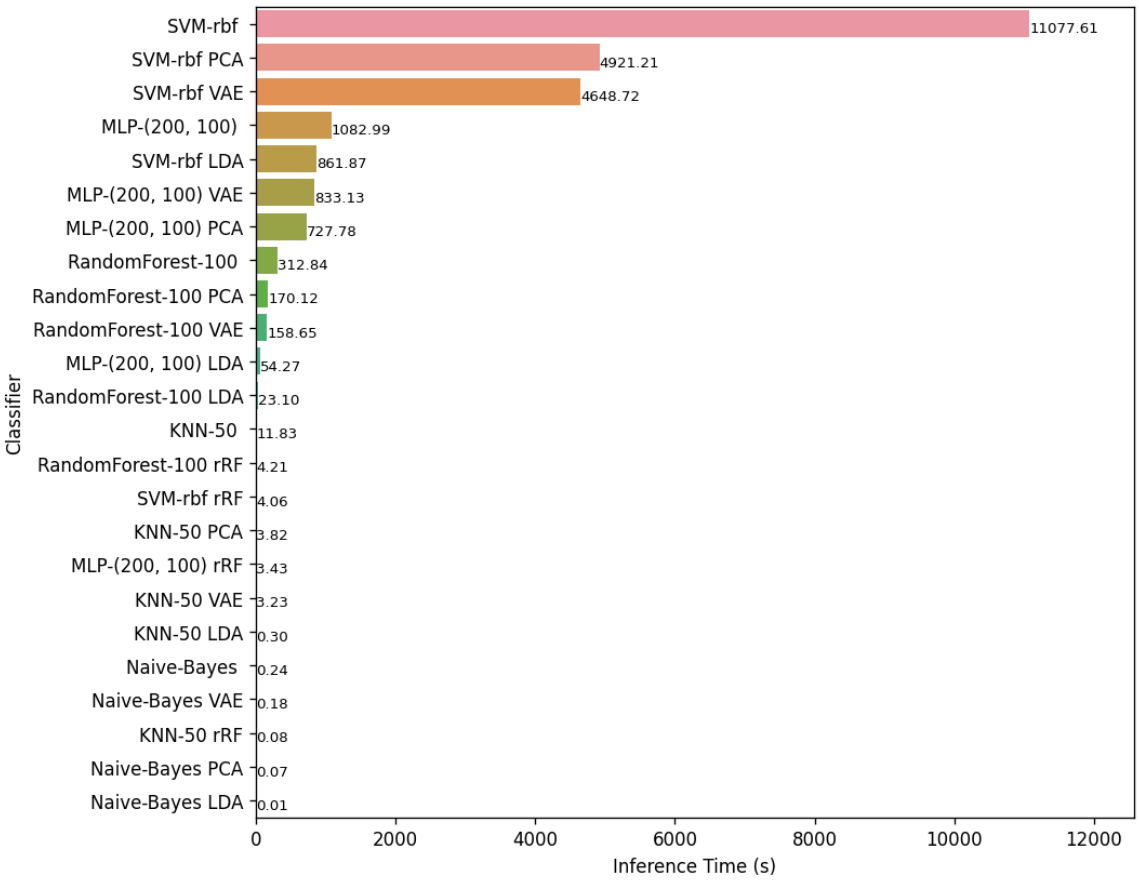

Figure 9: Comparison of inference time of different classifiers for the FEVER dataset

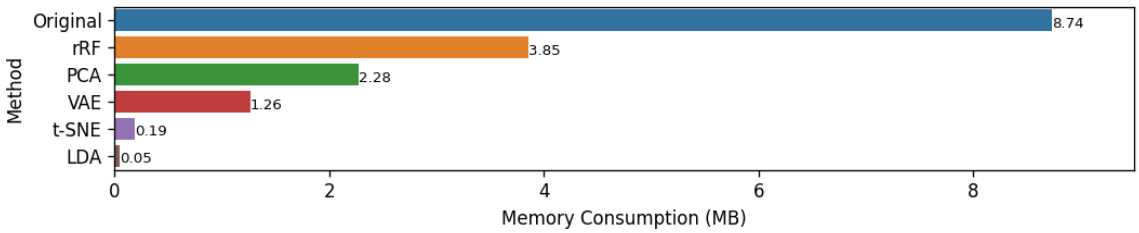

Figure 10: Comparison of memory consumption of different classifiers for the CheckThatLab dataset

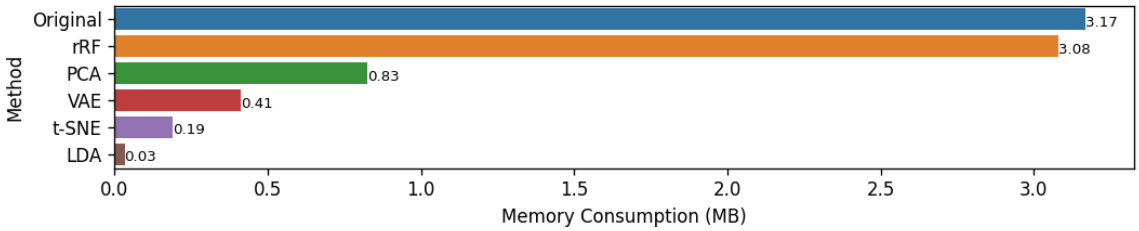

Figure 11: Comparison of memory consumption of different classifiers for the Aspect Sentiment Classification dataset

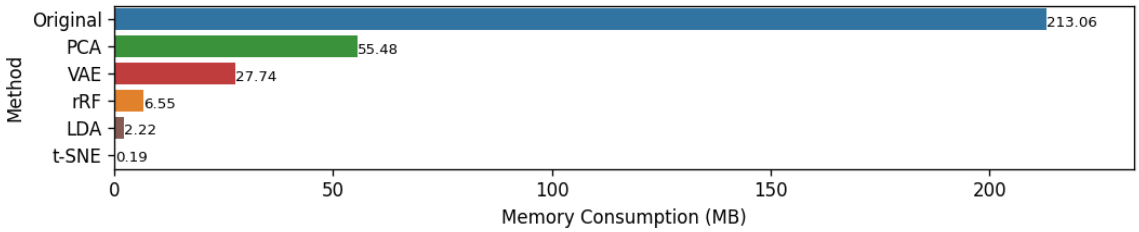

Figure 12: Comparison of memory consumption of different classifiers for the FEVER dataset