Utrecht University

Department of Information and Computing Science

Applied Data Science master thesis

Scaling and Normalization of Embeddings: Evaluating the Impact of Active Learning on ASReview Performance

First examiner:
Rens van de Schoot

Second examiner:
Duco Veen

Candidate:
Sjardi Djoy Willems

In cooperation with:
ASReview

July 1, 2024

Abstract

Active learning enhances efficiency in systematic reviews by optimizing the work saved over random sampling (WSS) and identifying relevant papers. This study investigates the impact of various preprocessing techniques on the performance of active learning models. Specifically, it evaluates the effectiveness of TF-IDF, SBERT, and Doc2Vec embeddings combined with different normalization and scaling methods, using Naive Bayes and logistic regression classifiers.

The findings indicate that TF-IDF embeddings, particularly with L2 normalization and adding the absolute minimum value paired with Naive Bayes, performed the best, achieving high recall and low average time to find relevant documents. The highest WSS of SBERT combinations is achieved by combining z-score or Pareto normalization and absolute minimum scaling with logistic regression, showed 3% lower WSS and required computational resources. Doc2Vec, although less effective than SBERT, performed well with z-score or Pareto normalization and CDF scaling without needing a GPU.

While TF-IDF remains a robust benchmark, SBERT and Doc2Vec offer promising alternatives for improving systematic reviews, warranting further exploration with additional configurations and fine-tuning. Further research should explore more combinations of feature extractors, classifiers, and normalization and scaling techniques.

Contents

1. Introduction

As citations double approximately every 12 years, researchers are more inclined to choose older articles with more citations and visibility [1]. Researchers face growing challenges in identifying the studies most relevant to their work. While this vast amount of information presents new opportunities, Della Briotta Parolo et al. (2015) argue that a direct result of this growth is the rapid decay of attention that can be devoted to individual papers. Consequently, less relevant papers are often found, and the quality of research may diminish. Systematic reviews and meta-analyses are essential for research findings, but the traditional manual review process is labor-intensive and time-consuming.

In response to the problem of finding relevant papers in less time, Van De Schoot et al. (2021) developed ASReview, a tool designed to conduct systematic reviews or meta-analyses as efficiently and transparently as possible. ASReview addresses these challenges by using active learning, a subset of machine learning, to streamline the review process. This innovative approach not only saves time but also enhances the quality of the review by ensuring that the most relevant studies are identified.

Active learning involves an iterative process where the researcher makes decisions about the relevance of scientific articles. After each decision, the machine learning algorithm learns and adapts, providing increasingly accurate results tailored to the researcher's needs [2]. This feedback loop enables the algorithm to achieve greater accuracy with fewer labeled training data by selectively learning from the most informative examples [3]. This process stands in contrast to traditional machine learning methods that require large, fully labeled datasets upfront.

Active learning enhances the review process through several key components: the classifier, feature extraction, balance strategy, and active learning query strategy. Central to this process is feature extraction, which transforms raw article data into a set of features that the classifier can understand and process. Common techniques include text vectorization methods like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings, which capture the semantic meaning of the text [4]. Effective feature extraction is crucial, as it directly impacts the classifier's ability to categorize articles accurately [5].

Despite the effectiveness of active learning [6, 7, 8, 9], current methods such as TF-IDF and Doc2Vec [10, 11] are fast, but may not capture the full semantic complexity of the text [12]. Additionally, these methods produce only positive embeddings, potentially limiting the classifier's performance [13]. This research aims to explore alternative feature extraction methods, including those that generate negative embeddings, and evaluate their impact on ASReview's performance. By addressing this gap, we seek to enhance the accuracy and efficiency of systematic reviews, making it easier for researchers to identify the most relevant studies.

After feature extraction, scaling and normalization methods are often applied to optimize the data for classification [14, 15]. Scaling techniques like Z-score scaling, which standardizes data to have a mean of zero and a standard deviation of one, are widely used in machine learning due to their effectiveness in handling features with different scales [16]. [17] for example shows the effectivesness of z-score normalization on sequence classification tasks with SBERT, as well as [18] shows promising result with z-score and Doc2Vec embeddings.

Other scaling techniques in this study are the L2 normalization [19] and Pareto scaling [20].

Which adjust the range or distribution of the data, enhancing the classifier's ability to process features effectively. The normalization methods ensure all values are positive, which is particularly important for certain classifiers like Naïve Bayes [21]. These methods include Min-Max normalization [22], normalization by adding the absolute value of the lowest number, Cumulative Distribution Function (CDF) [23] normalization, and Sigmoid normalization [24]. Each technique offers unique advantages in preparing data for machine learning algorithms, contributing to the overall effectiveness of the classification process.

The classifier, a machine learning model, categorizes articles as relevant or irrelevant based on the extracted features. The chosen classifiers each offers different advantages depending on the dataset and research context [25]. For this study the best performing classifiers from ASReview are picked, namely TF-IDF and Naive bayes. The balance strategy addresses imbalanced datasets, where relevant articles are vastly outnumbered by irrelevant ones, using techniques like oversampling, undersampling, or specialized loss functions [26]. The active learning query strategy determines how the algorithm selects the most informative samples to learn from using different sampling methods [27].

This experimental setup of 102 simulations focuses on transforming the outcomes of feature extractors with normalization and scaling techniques to enhance compatibility with naive Bayes classifiers and evaluate their impact on active learning performance. By optimizing these components and their interactions, ASReview aims to offer a more powerful and efficient tool for conducting systematic reviews. The central question guiding this research is how feature extraction methods, including those generating negative embeddings, and their subsequent normalization and scaling impact the performance of active learning on ASReview's accuracy and efficiency.

This study investigates the comparison between advanced feature extraction methods like transformers and traditional approaches such as TF-IDF and Doc2Vec in capturing semantic complexities, while also considering their computational trade-offs. Additionally, it explores the transformation of negative embeddings into positive ones through normalization and scaling techniques, examining their effects on the overall performance of classifiers in active learning. Through these investigations, this research aims to reduce the workload on researchers and ensure more comprehensive, high-quality reviews that advance scientific knowledge in an era of rapidly expanding literature.

2. Method

2.1 Overview

This section describes the methods used to answer the research question on the effectiveness of different feature extraction methods combined with normalization techniques and classifiers. It covers the classifiers, feature extraction techniques, parameters, implementation details, and evaluation metrics. The design construction is as follows: 1 dataset x 3 feature extraction techniques x 4 normalization methods x 3 scaling methods x 2 classifiers + 3 FE x 4 normalization methods x 2 classifiers + 3 FE x 2 classifiers = 102 simulations. ASReview offers a wide variety of adjustable parameters. If any parameter (like balancing method and query strategy) is not described here, the default values are used.

2.2 Data

The SYNERGY dataset is a free and open source dataset assembled to study selection processes in systematic reviews [28]. Each dataset within SYNERGY are labeled, this means the accuracy of the classifiers can be checked and makes them usable for systematic reviews. The dataset used from SYNERGY is the van_de_schoot_2018 dataset that focuses on Post-Traumatic Stress Disorder (PTSD). This dataset is compiled from four major databases: Pubmed, Embase, PsychInfo, and Scopus [29]. The dataset contains 4544 useable rows. These 4544 rows contain: DOI, OpenAlex ID, label, PubMed ID, and retrieval method. From the labeled data, 38 rows are labeled as relevant.

2.3 Data Preprocessing

The data preprocessing approach employs a two-step process: scaling followed by normalization. This sequence is crucial for handling embeddings that may contain negative values, which can be problematic for certain classifiers like Naïve Bayes.

2.3.1 Scaling Methods

Scaling is applied first to adjust the range or distribution of the data. The following scaling techniques are used:

1. **Z-score Scaling**: Also known as standardization or Pearson scaling, this well-known scaling technique rescales the data so that the features have zero mean and unit variance [16]. The dataset is scaled by the standard deviation itself [20]. The scaled dataset retains its original distance and order. It's defined as:

$$z = \frac{x - \mu}{\sigma} \tag{2.1}$$

   where z is the standardized value, x is the original value, μ is the mean of the dataset, and σ is the standard deviation of the dataset.

2. **Pareto Scaling**: Where the z-score is calculated by dividing by the standard deviation, the Pareto score is calculated by dividing by the square root. The variance of the new features is equal to the standard deviation of non-normalized features [16]. The scaled dataset retains its original distance and order. It's defined as:

$$x' = \frac{x - \bar{x}}{\sqrt{\sigma}} \tag{2.2}$$

   where x' is the Pareto scaled value, x is the original value, $\bar{x}$ is the mean of the variable, and σ is the standard deviation of the variable.

3. **L2 Normalization**: Also known as vector or Euclidean normalization, L2 normalization scales the data so that the sum of the squares of each element equals 1. This method transforms each data point by dividing it by the L2 norm (or Euclidean norm) of the vector. Unlike Z-score and Pareto scaling, L2 normalization does not retain the original order or relative distances between data points. The direction of the vectors is maintained, making it useful in embeddings [19]. It's defined as:

$$x'_i = \frac{x_i}{\sqrt{\sum_j x_j^2}} \tag{2.3}$$

   where x'_i is the L2 normalized value and x_i is the original value.

2.3.2 Normalization Methods

After scaling, normalization methods are applied specifically to ensure all values are positive, which is necessary for certain classifiers like Naïve Bayes. The normalization methods used are:

1. **Min-Max Normalization**: This method provides a linear transformation on the original range of data [22]. It is typically scaled to the range [0,1]. The relative distance and order are preserved in this method. It's defined as:

$$x' = \left(\frac{x - X_{\min}}{X_{\max} - X_{\min}}\right) \times (\text{new\_max}_X - \text{new\_min}_X) + \text{new\_min}_X \tag{2.4}$$

   where x' is the normalized value, x is the original value, $X_{\min}$ is the minimum value of the feature, $X_{\max}$ is the maximum value of the feature, new_max$_X$ is the new maximum value of the feature (1), and new_min$_X$ is the new minimum value of the feature (0).

2. **Normalization by Adding Absolute Value of the Lowest Number**: This method shifts all values in the data by adding the absolute value of the minimum value in the data to ensure all values are positive. This also keeps the same distance between numbers and order as in the original dataset. It's defined as:

$$x' = x + |X_{\min}| \tag{2.5}$$

where x' is the normalized value, x is the original value, and $|X_{\min}|$ is the absolute value of the minimum value of the feature.

3. **CDF Normalization**: This method applies the cumulative distribution function to transform data values into a probability-based scale, typically in the range [0, 1]. The CDF for a normal distribution is defined as:

$$F_X(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)\right] \tag{2.6}$$

where $F_X(x)$ is the CDF value, x is the original value, μ is the mean of the dataset, σ is the standard deviation of the dataset, and erf is the error function.

4. **Sigmoid Normalization**: This method applies the sigmoid function to scale data values into the range (0, 1) in a non-linear fashion. The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.7}$$

where $\sigma(x)$ is the normalized value and x is the original value and e is the base of the natural logarithm (approximately 2.71828).

In the experimental setup, one scaling method is applied followed by one normalization method. This two-step process that scales the data, and ensures that all values are positive, making it suitable for all classifiers in the study, including those sensitive to negative values. All feature extraction methods were also used without any normalization or scaling and with only a normalization method.

2.3.3 Feature Extraction Technique

This technique transforms the raw abstracts from the articles into a set of features that the classifier can understand and process. The techniques for this study are Doc2Vec, SBERT as implemented in ASReview and TF-IDF (Term Frequency-Inverse Document Frequency) as a comparison benchmark.

2.3.4 Classifier

The classifier is a machine learning model that categorizes articles as relevant or irrelevant based on the features extracted from them. For this study, the naïve Bayes algorithm is employed. Caklovic (2022) demonstrated its effectiveness for ASReview. However, it is not suitable for negative embeddings without adding some normalization method that makes the embeddings positive. Every combination that is done with naïve Bayes is also done with logistic regression as a benchmark.

Feature Extraction	Scaling Technique	Normalization Technique	Classifier
Doc2Vec	Z-score Pareto L2 Norm.	Min-Max Sigmoid CDF Absolute Minimum Scaling	Naïve Bayes Logistic Regression
SBERT			
TF-IDF			

Table 2.1: The feature extraction methods, scaling techniques, normalization techniques and classifiers used.

2.3.5 Performance Metrics

Different performance metrics are used to evaluate the efficiency and effectiveness of the simulation processes, as described below:

Work Saved over Sampling (WSS): Work Saved over Sampling (WSS) measures the efficiency gain of using active learning compared to random screening in record review processes. Specifically, it quantifies the percentage decrease in the number of records that need to be screened when employing active learning techniques. WSS is typically evaluated at a predetermined level of recall for relevant records, such as WSS@95%. This means it shows how much screening effort can be reduced while still identifying 95% of the relevant records. In other words, it demonstrates the labor savings achieved at the expense of missing 5% of the pertinent information [30].

Other metrics: The *Average Time to Discovery* (ATD) indicates how many records need to be screened on average to find all relevant records in the dataset. The *recall* is the amount of relevant records that have been found at a certain point during the screening phase, also known as Relevant References Found (RRF). For example, RFF10% indicates the number of relevant records found after screening the first 10% of records.

Compute time: The time measured is from the moment the main batch file calls the ASReview Makita template command until the jobs.bat is done. This is all measured automatically in the main batch file. The GPU utilization is calculated per combination simulation ran.

2.3.6 Execution

All the code was executed on a High Performing Computer with an A10 Nvidia GPU, 96GB RAM and a Intel Xeon Gold 6342. The code can be found here: `https://github.com/sjardi/ADS_Master_Thesis`. It has a readme file with all the list of libraries and commands needed to run the code. By firing the main bat file (e.g. doc2vec_main.bat) it iterates through all the classifiers, feature extractors and normalization / scaling techniques.

3. Results

All metrics that are not further discussed here, can be found in appendix A.

TF-IDF Embeddings

For TF-IDF embeddings, the combination with the naïve bayes algorithm proved to be the most efficient and highest performing. Specifically, the combination of naïve bayes, l2 normalization and adding the absolute minimum achieved a recall at 0.1 threshold (*recall_0.1*) of 97.3, a Work Saved over Sampling at 0.95 recall (*WSS@95*) of 90.4, Extra Relevant Records (*erf_0.1*) of 86.5, and an Average Time to Discover (*ATD*) of 90. These techniques demonstrated the best recall scores, highest work saved over sampling, and lowest average time duration, indicating extremely efficient document retrieval. Though these are only very slightly better by 0.7% compared to the baseline. The top three best performing all are normalized with l2 normalization. The small difference of WSS and ATD will not probably not be noticeable for most use cases.

Combination	**wss 0.95**	**atd**	**runtime**	**GPU(%)**
(baseline) logistic - TF-IDF	89.4	117	65.8	0.00
(baseline) naïve bayes - TF-IDF	89.7	91	53.8	0.00
naïve bayes - l2 norm. - absmin	90.4	90	163.8	0.15
logstic - l2 norm. - absmin	90.1	105	234.8	0.17
naïve bayes - l2 norm. - sigmoid	89.9	108	241.5	0.32

Note. The baseline and the three best performing combinations with the TF-IDF classifier.

SBERT Embeddings

For SBERT embeddings, the two top performing combination was logistic regression with z-score normalization and logistic regression with absolute minimum scaling and pareto normalization. Both configurations achieved a *recall_0.1* of 94.6, a *WSS@95* of 87.5 and 87.4, an *erf_0.1* of 83.8, and an *ATD* of 150 and 140 respectively. This technique provided high recall and efficient document retrieval, making it effective for use with SBERT embeddings. Yet the WSS is 3% lower than TF-IDF, the average time to detection is 60 documents higher while recall and ERF are slightly lower.

Combination	**wss 0.95**	**atd**	**runtime**	**GPU(%)**
logistic - SBERT	87.0	139	228.5	57
logistic - z-score - absmin	87.5	151	287.0	11
logistic - pareto - absmin	87.4	140	246.0	13
logistic - l2 norm. - minmax	87.1	127	219.0	15

Note. The baseline and the three best performing combinations with the SBERT classifier.

Doc2Vec Embeddings

For Doc2Vec embeddings, the top performing configurations were naïve bayes with pareto or z-score and CDF. This configuration achieved a *recall_0.1* of 94.6, a *WSS@95* of 84.5, an *erf_0.1* of 83.8, and an *ATD* of 220. Yet the baseline still did better with a runtime of 150 while the other metrics were the same. The findings suggest that optimizing the Doc2Vec configuration can impact the recall and efficiency of regression models.

Combination	**wss 0.95**	**atd**	**runtime**	**GPU(%)**
(baseline) Logistic - Doc2Vec	84.5	228	150.0	0.0
naïve bayes - pareto - cdf	84.5	181	220.0	0.1
Logistic - z-score - cdf	84.3	194	221.0	0.2
naïve bayes - l2 norm. - cdf	83.9	191	220.0	0.0

Note. The baseline and the three best performing combinations with the Doc2Vec classifier.

4. Conclusion

This study investigated the effects of various normalization and scaling techniques applied to different document embeddings within the context of active learning for systematic reviews. The primary objective was to identify preprocessing methods that enhance recall, reduce workload, and improve the overall efficiency of the active learning process.

The findings indicate that TF-IDF embeddings consistently achieved the highest efficiency. Specifically, the combination of Naive Bayes with TF-IDF, l2 normalization and adding the absolute minimum value configuration achieved a recall at 0.1 threshold (*recall_0.1*) of 0.973, a normalized Work Saved over Sampling (WSS@95) of 0.904, an Extra Relevant Records (*erf_0.1*) of 0.865, and an Average Time to Discover (ATD) of 90. These results demonstrate the high performance of this combination in identifying relevant records quickly and accurately while reducing the workload needed for screening documents.

SBERT embeddings also showed strong performance, particularly with the z-score normalization combined with absolute minimum scaling and logistic regression. This configuration achieved a *recall_0.1* of 0.946, a *WSS@95* of 0.875, an *erf_0.1* of 0.838, and an *ATD* of 150.568. However, this approach required more computational resources compared to TF-IDF, making it less efficient overall despite its high recall potential.

For Doc2Vec embeddings, the top performing configurations were Naive Bayes with Pareto or z-score and CDF. This setup achieved a *recall_0.1* of 94.6, a *WSS@95* of 84.5, an *erf_0.1* of 83.8, and an *ATD* of 220. Yet, the baseline still did better with a runtime of 150 while the other metrics were the same.

Overall, while TF-IDF remains a robust and efficient benchmark for active learning in systematic reviews, Doc2Vec configurations show considerable potential, and SBERT provides high-recall options despite higher computational costs. These findings are consistent with prior research emphasizing the importance of efficient preprocessing methods in enhancing the performance of machine learning models in text classification tasks. Yet when adding normalization or scaling methods the runtime increases, sometimes to even 2 to 20 times the baseline value.

Future research should explore additional normalization techniques and embedding types, and consider the integration of more advanced machine learning algorithms to further enhance the efficiency and accuracy of systematic reviews. Additionally, the trade-offs between computational efficiency and recall performance should be further investigated to optimize model configurations for practical applications in various domains.

4.1 Discussion

Despite the promising results, this study has several limitations. The evaluation was conducted on a specific set of embeddings and normalization techniques, and there might be other combinations that would work better. Additionally, the computational efficiency of the models was assessed without any extensive algorithmic optimizations. Due to time constraints the code is hardcoded. If Makita were to support pre- or post-processing methods like normalization and scaling as a parameter, there can easily be added much more algorithms and combinations for simulations.

Future research should explore a broader range of embeddings and preprocessing methods, including more advanced machine learning algorithms and hybrid approaches that combine multiple embeddings. Moreover, integrating domain-specific knowledge into the preprocessing pipeline could further enhance model performance. For example, fine-tuning a transformer with a dataset of papers in the research field could provide better contextual understanding and improve outcomes [31, 32, 33].

The feature extraction, scaling, and normalization techniques utilized in this study are optimized for GPU acceleration, significantly enhancing computational efficiency. However, the classifiers themselves (except SBERT) do not leverage GPU capabilities, resulting in a bottleneck in processing speed. To address this, leveraging a GPU-enabled library such as NVIDIA's RAPIDS cuML library [34, 35] could allow for GPU acceleration of classifiers like logistic regression and naïve Bayes, thereby improving overall performance.

The computing power is high on SBERT, but still no more than 22% utilization for non default combinations. Yet, the hardware costs come close to 10.000 euro. This is not something that every researcher has to its disposal. But using ASReview as a SaaS solution can be a solution.

In conclusion, this study provides insights into the optimization of document embeddings for active learning in systematic reviews. The results demonstrate the effects of selecting appropriate preprocessing techniques to balance recall, workload, and computational efficiency, paving the way for more effective and efficient systematic review processes.

5. Acknowledgement

With the help of Makita it became much easier to create and execute multiple simulation. Thanks for Jelle Teijema for providing insights and feedback on this process.

5.1 Data scripts

All code can be found on: - ASReview: https://github.com/asreview - Makita: https://github.com/asreview/ makita - Data: https://github.com/asreview/synergy-dataset - Code used in this paper: https://github.com/ thesis_ADS

5.2 Generative A.I.

Most of the code and scripts are created with help of ChatGPT or Claude AI. Parts of the text in this paper are checked and improved by the LLM's.

Bibliography

[1] Pietro Della Briotta Parolo et al. "Attention decay in science". In: *Journal of Informetrics* 9.4 (Oct. 2015), pp. 734–745. ISSN: 1751-1577. DOI: 10.1016/j.joi.2015.07.006. URL: http://dx.doi.org/10.1016/j.joi.2015.07.006.

[2] Ruoyu Wang. "Active Learning-Based Optimization of Scientific Experimental Design". In: *2021 2nd International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*. IEEE, Nov. 2021. DOI: 10.1109/icaice54393.2021.00060. URL: http://dx.doi.org/10.1109/ICAICE54393.2021.00060.

[3] Jingyu Shao, Qing Wang, and Fangbing Liu. "Learning to Sample: an Active Learning Framework". In: *CoRR* abs/1909.03585 (2019). arXiv: 1909.03585. URL: http://arxiv.org/abs/1909.03585.

[4] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. July 2008.

[5] R. Kavitha and E. Kannan. "An efficient framework for heart disease classification using feature extraction and feature selection technique in data mining". In: *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*. 2016, pp. 1–5. DOI: 10.1109/ICETETS.2016.7603000.

[6] Amirhossein Saeidmehr, Piers David Gareth Steel, and Faramarz F. Samavati. "Systematic review using a spiral approach with machine learning". In: *Systematic reviews* 13.1 (Jan. 2024). DOI: 10.1186/s13643-023-02421-z. URL: https://doi.org/10.1186/s13643-023-02421-z.

[7] Kevin E. K. Chai et al. "Research Screener: a machine learning tool to semi-automate abstract screening for systematic reviews". In: *Systematic reviews* 10.1 (Apr. 2021). DOI: 10.1186/s13643-021-01635-3. URL: https://doi.org/10.1186/s13643-021-01635-3.

[8] Brian E. Howard et al. "SWIFT-Active Screener: Accelerated document screening through active learning and integrated recall estimation". In: *Environment international* 138 (May 2020), p. 105623. DOI: 10.1016/j.envint.2020.105623. URL: https://www.sciencedirect.com/science/article/pii/S0160412019314023.

[9] Eduardo Mosqueira-Rey et al. "Human-in-the-loop machine learning: a state of the art". In: *Artificial intelligence review* 56.4 (Aug. 2022), pp. 3005–3054. DOI: 10.1007/s10462-022-10246-w. URL: https://doi.org/10.1007/s10462-022-10246-w.

[10] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *arXiv (Cornell University)* (Jan. 2014). DOI: 10.48550/arxiv.1405.4053. URL: https://arxiv.org/abs/1405.4053.

[11] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[12] Kowsari et al. "Text Classification Algorithms: A Survey". In: *Information* 10.4 (Apr. 2019), p. 150. ISSN: 2078-2489. DOI: 10.3390/info10040150. URL: http://dx.doi.org/10.3390/info10040150.

[13] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. DOI: 10.48550/ARXIV.1301.3781. URL: https://arxiv.org/abs/1301.3781.

[14] Kelsy Cabello-Solorzano et al. *The Impact of Data Normalization on the Accuracy of Machine Learning Algorithms: A Comparative Analysis*. Jan. 2023, pp. 344–353. DOI: 10.1007/

978-3-031-42536-3\{_}33. URL: https://doi.org/10.1007/978-3-031-42536-3_33.

[15] Lihao Ge and Teng-Sheng Moh. "Improving text classification with word embedding". In: *2017 IEEE International Conference on Big Data (Big Data)*. 2017, pp. 1796–1805. DOI: 10.1109/BigData.2017.8258123.

[16] Dalwinder Singh and Birmohan Singh. "Investigating the impact of data normalization on classification performance". In: *Applied soft computing* 97 (Dec. 2020), p. 105524. DOI: 10.1016/j.asoc.2019.105524. URL: https://doi.org/10.1016/j.asoc.2019.105524.

[17] Hassan Sajjad et al. "Effect of Post-processing on Contextualized Word Representations". In: *arXiv (Cornell University)* (Jan. 2021). DOI: 10.48550/arxiv.2104.07456. URL: https://arxiv.org/abs/2104.07456.

[18] Tomasz Walkowiak and Mateusz Gniewkowski. "Evaluation of vector embedding models in clustering of text documents". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Ed. by Ruslan Mitkov and Galia Angelova. Varna, Bulgaria: INCOMA Ltd., Sept. 2019, pp. 1304–1311. DOI: 10.26615/978-954-452-056-4_149. URL: https://aclanthology.org/R19-1149.

[19] Jarrod Haas, William Yolland, and Bernhard Rabus. "Exploring Simple, High Quality Out-of-Distribution Detection with L2 Normalization". In: *arXiv (Cornell University)* (Jan. 2023). DOI: 10.48550/arxiv.2306.04072. URL: https://arxiv.org/abs/2306.04072.

[20] Robert A Van Den Berg et al. "Centering, scaling, and transformations: improving the biological information content of metabolomics data". In: *BMC genomics* 7.1 (June 2006). DOI: 10.1186/1471-2164-7-142. URL: https://doi.org/10.1186/1471-2164-7-142.

[21] Ana Caklovic. *Out with the Old and in with the New? - A Comparison of Classical vs. State-of-the-Art Feature Extractors in the Context of Systematic Reviews*. 2022. URL: https://studenttheses.uu.nl/handle/20.500.12932/42409.

[22] S. Gopal Krishna Patro and Kishore Kumar Sahu. "Normalization: a preprocessing stage". In: *arXiv (Cornell University)* (Jan. 2015). DOI: 10.48550/arxiv.1503.06462. URL: https://arxiv.org/abs/1503.06462.

[23] Lingzi Zhang et al. *Are ID Embeddings Necessary? Whitening Pre-trained Text Embeddings for Effective Sequential Recommendation*. 2024. arXiv: 2402.10602 [cs.IR]. URL: https://arxiv.org/abs/2402.10602.

[24] Xiaoyan Zhuo, Jialing Zhang, and Seung Woo Son. "Network intrusion detection using word embeddings". In: *2017 IEEE International Conference on Big Data (Big Data)*. 2017, pp. 4686–4695. DOI: 10.1109/BigData.2017.8258516.

[25] Liang Yao, Chengsheng Mao, and Yuan Luo. "Graph Convolutional Networks for Text Classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 7370–7377. ISSN: 2159-5399. DOI: 10.1609/aaai.v33i01.33017370. URL: http://dx.doi.org/10.1609/aaai.v33i01.33017370.

[26] Rie Johnson and Tong Zhang. "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks". In: *arXiv (Cornell University)* (Jan. 2014). DOI: 10.48550/arxiv.1412.1058. URL: https://arxiv.org/abs/1412.1058.

[27] Punit Kumar and Atul Gupta. "Active Learning Query Strategies for Classification, Regression, and Clustering: A Survey". In: *Journal of Computer Science and Technology* 35.4 (July 2020), pp. 913–945. ISSN: 1860-4749. DOI: 10.1007/s11390-020-9487-4. URL: http://dx.doi.org/10.1007/s11390-020-9487-4.

[28] Jonathan De Bruin et al. *SYNERGY - Open machine learning dataset on study selection in systematic reviews*. Version V1. 2023. DOI: 10.34894/HE6NAQ. URL: https://doi.org/10.34894/HE6NAQ.

[29] Rens Van De Schoot et al. "Bayesian PTSD-Trajectory Analysis with Informed Priors Based on a Systematic Literature Search and Expert Elicitation". In: *Multivariate behavioral research* 53.2 (Jan. 2018), pp. 267–291. DOI: 10.1080/00273171.2017.1412293. URL: https://doi.org/10.1080/00273171.2017.1412293.

[30] Rens van de Schoot et al. "An open source machine learning framework for efficient and transparent systematic reviews". In: *Nature Machine Intelligence* 3.2 (Feb. 2021), pp. 125–133. ISSN: 2522-5839. DOI: 10.1038/s42256-020-00287-7. URL: http://dx.doi.org/10.1038/s42256-020-00287-7.

[31] Cristiano Mesquita Garcia et al. "Improving Sampling Methods for Fine-tuning SentenceBERT in Text Streams". In: *arXiv (Cornell University)* (Mar. 2024). DOI: 10.48550/arxiv.2403.15455. URL: https://arxiv.org/abs/2403.15455.

[32] Juri Grosjean and Jannis Vamvas. *Fine-tuning the SwissBERT Encoder Model for Embedding Sentences and Documents*. May 2024. URL: https://arxiv.org/abs/2405.07513.

[33] Bin Wang and C.-c. Jay Kuo. "SBERT-WK: A Sentence Embedding Method by Dissecting BERT-Based Word Models". In: *IEEE/ACM transactions on audio, speech, and language processing* 28 (Jan. 2020), pp. 2146–2157. DOI: 10.1109/taslp.2020.3008390. URL: https://doi.org/10.1109/taslp.2020.3008390.

[34] Sebastian Raschka, Joshua Patterson, and Corey Nolet. "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence". In: *Information* 11.4 (Apr. 2020), p. 193. DOI: 10.3390/info11040193. URL: https://doi.org/10.3390/info11040193.

[35] RAPIDS Development Team. *RAPIDS: Libraries for End to End GPU Data Science*. 2023. URL: https://rapids.ai.

Appendices

A. Metrics

This chapter shows all the metrics gained from running the code. It is color coded per column: best score, second best, lowest score, second lowest.

A.1 SBERT

Combination	recall 0.1	wss 0.95	erf 0.1	atd	runtime	GPU(%)
logistic - SBERT	97.3	87	86.5	139	228.5	57
logistic - z-score - absmin	94.6	87.5	83.8	151	287	11
logistic - pareto - absmin	94.6	87.4	83.8	140	246	13
logistic - l2 normalize - minmax	94.6	87.1	83.8	127	219	15
logistic - pareto - minmax	94.6	87.1	83.8	127	223	15
logistic - z-score - minmax	94.6	87.1	83.8	127	212	7
naïve bayes - z-score - absmin	94.6	86.4	83.8	136	184	18
logistic - sigmoid	94.6	85.4	83.8	155	258	13
logistic - pareto - cdf	94.6	85.4	83.8	152	249	13
logistic - z-score - cdf	94.6	85.4	83.8	153	250	13
logistic - cdf	94.6	85.4	83.8	152	254	13
logistic - l2 normalize - cdf	94.6	85.4	83.8	153	255	13
logistic - pareto - sigmoid	97.3	86.0	86.5	116	209	16
logistic - l2 normalize - sigmoid	94.6	85.9	83.8	171	202	16
logistic - l2 normalize - absmin	94.6	84.9	83.8	154	198	17
logistic - z-score - sigmoid	94.6	84.9	83.8	145	230	14
naïve bayes - pareto - absmin	94.6	84.9	83.8	138	186	18
NB - l2 normalize - minmax	94.6	84.8	83.8	131	182	18
naïve bayes - pareto - minmax	94.6	84.8	83.8	131	181	18
naïve bayes - z-score - minmax	94.6	84.8	83.8	131	182	18
naïve bayes - pareto - sigmoid	94.6	84.7	83.8	136	189	18
NB - l2 normalize - sigmoid	94.6	84.0	83.8	145	181	18
naïve bayes - sigmoid	94.6	84.1	83.8	143	188	18
logistic - absmin	94.6	84.1	83.8	127	211	16
naïve bayes - absmin	94.6	84.3	83.8	127	151	22
naïve bayes - z-score - sigmoid	94.6	83.9	83.8	148	187	18
naïve bayes - cdf	91.9	82.4	81.1	153	174	19
naïve bayes - l2 normalize - cdf	91.9	82.4	81.1	153	176	19
naïve bayes - pareto - cdf	91.9	82.4	81.1	153	174	19
naïve bayes - z-score - cdf	91.9	82.4	81.1	153	172	19

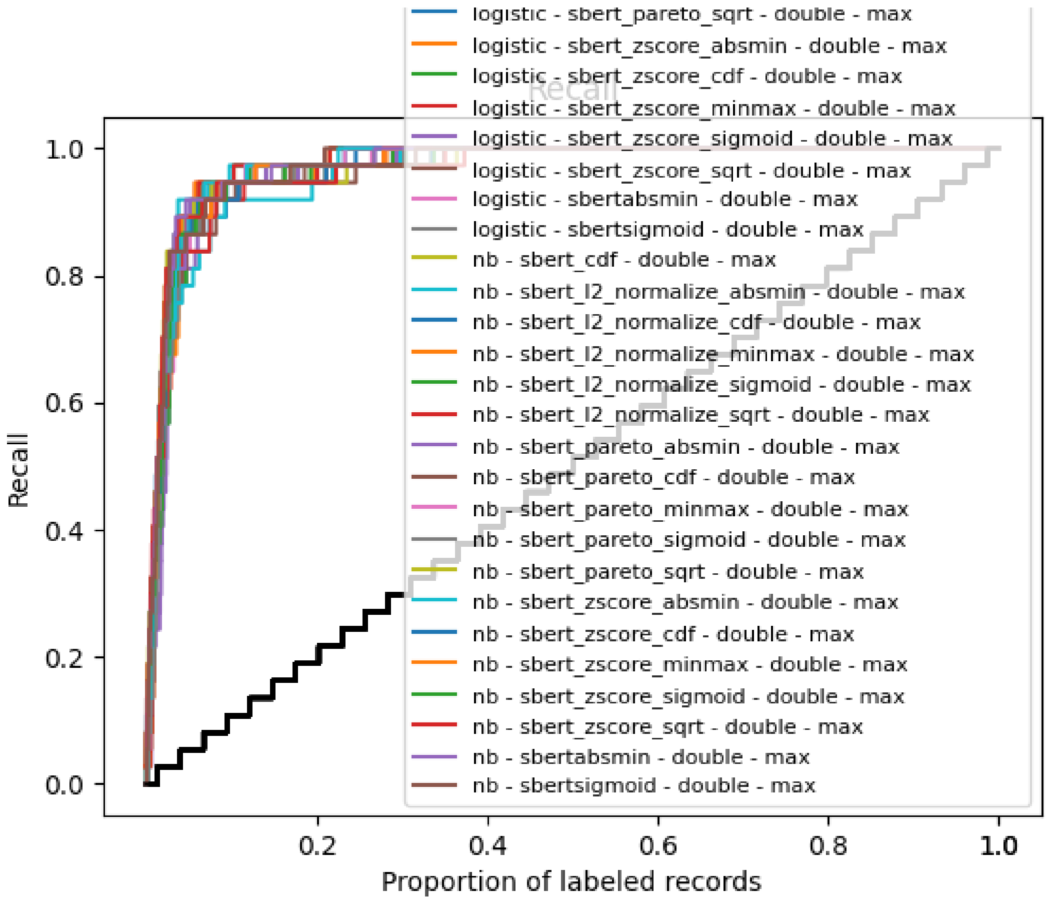

Figure A.1: All SBERT - combinations Recall Plot

[H] *Note.* This table summarizes the performance metrics for logistic and naïve Bayes (NB) regression models using different SBERT configurations.

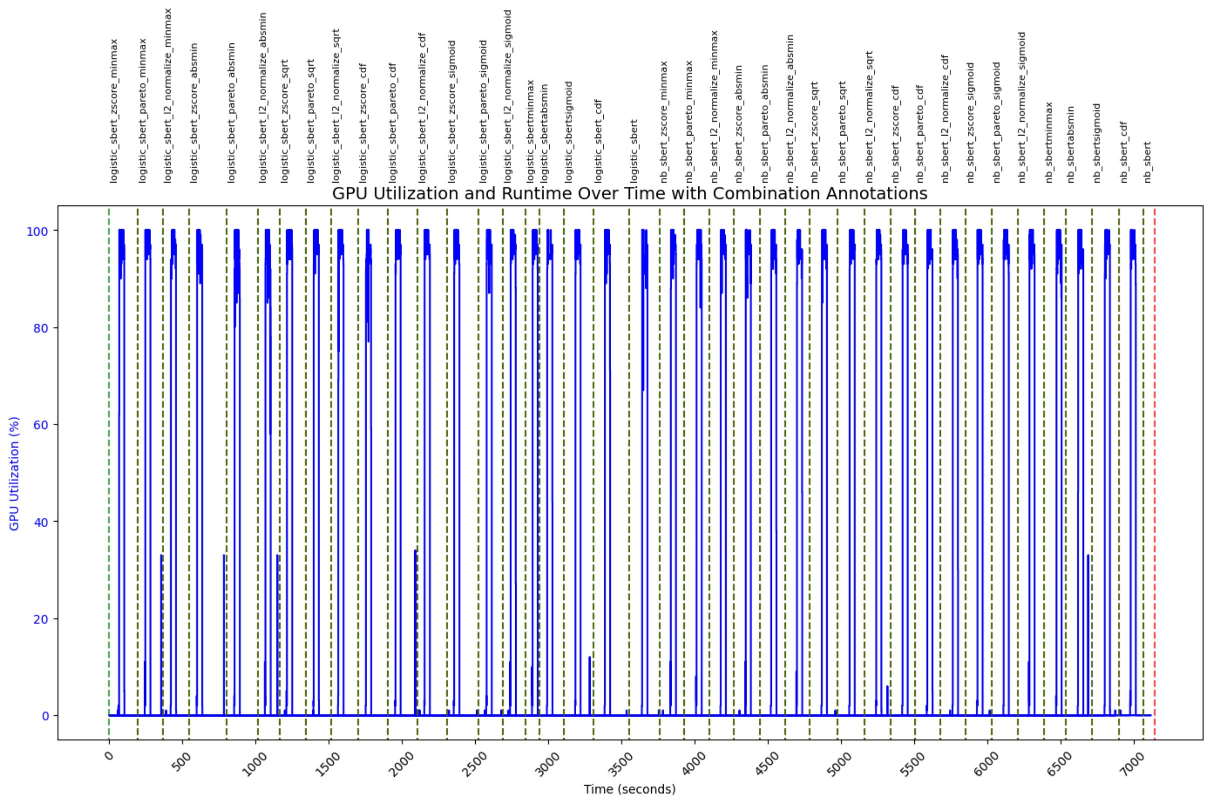

Figure A.2: The blue line is when the GPU was running and for how much percentage the GPU is being utilized. It is visible that both SBERT and its normalization and scaling techniques are using a lot of GPU power. The only downtime is from using the classifiers.

A.2 Doc2Vec

Combination	recall 0.1	wss 0.95	erf 0.1	atd	runtime	GPU(%)
(baseline) Logistic - Doc2Vec	.946	.845	.838	228	150	0.0
NB - pareto - cdf	.946	.845	.838	181	220	0.1
Logistic - z-score - cdf	.946	.843	.838	194	221	0.2
NB - l2 norm - cdf	.946	.839	.838	191	220	0.0
NB - pareto - sigmoid	.946	.838	.838	197	225	0.0
NB - z-score - cdf	.946	.838	.838	198	220	0.0
Logistic - l2 norm - cdf	.946	.837	.838	197	222	0.0
Logistic - z-score - absmin	.919	.830	.811	210	218	0.1
NB - l2 norm - absmin	.919	.830	.811	221	223	0.1
Logistic - pareto - sigmoid	.919	.830	.811	202	226	0.0
NB - z-score - sigmoid	.919	.825	.811	203	223	0.0
Logistic - pareto - minmax	.919	.825	.811	224	211	0.2
Logistic - pareto - cdf	.919	.823	.811	219	225	0.1
Logistic - l2 norm - absmin	.892	.823	.784	245	365	0.0
Logistic - pareto - absmin	.919	.821	.811	210	221	0.1
NB - z-score - absmin	.892	.821	.784	240	223	0.0
NB - pareto - absmin	.892	.818	.784	247	225	0.1
NB - pareto - minmax	.892	.818	.784	239	221	0.0
Logistic - z-score - sigmoid	.919	.817	.811	215	224	0.0
NB - z-score - minmax	.919	.807	.811	245	330	0.0

Combination	recall 0.1	wss 0.95	erf 0.1	atd	runtime	GPU Util. (%)
NB - l2 norm - sigmoid	.892	.805	.784	240	222	0.0
Logistic - z-score - minmax	.892	.799	.784	234	182	0.2
Logistic - l2 norm - sigmoid	.649	.786	.541	178	985	0.0

Note. The average GPU utilization is calculated per combination. If the utilization is below 0.01% it will show up as zero. All methods do use GPU enabled functions, but by far the most time is spent by the classifiers.

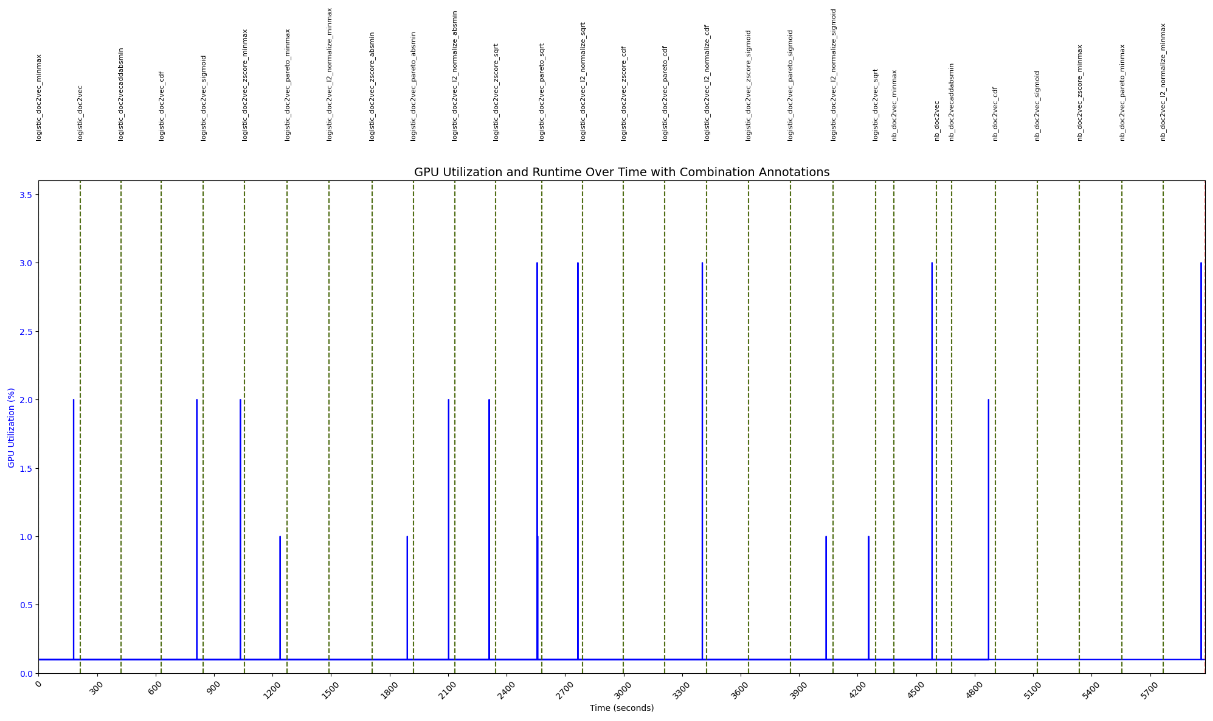

Figure A.3: GPU utilization of Doc2Vec

The small spikes indicate a very short burst of (low) GPU utilization. This indicates it is not optimized for using a GPU. This hinders performance and might not speed up the process much. The outcome metrics should be the same.

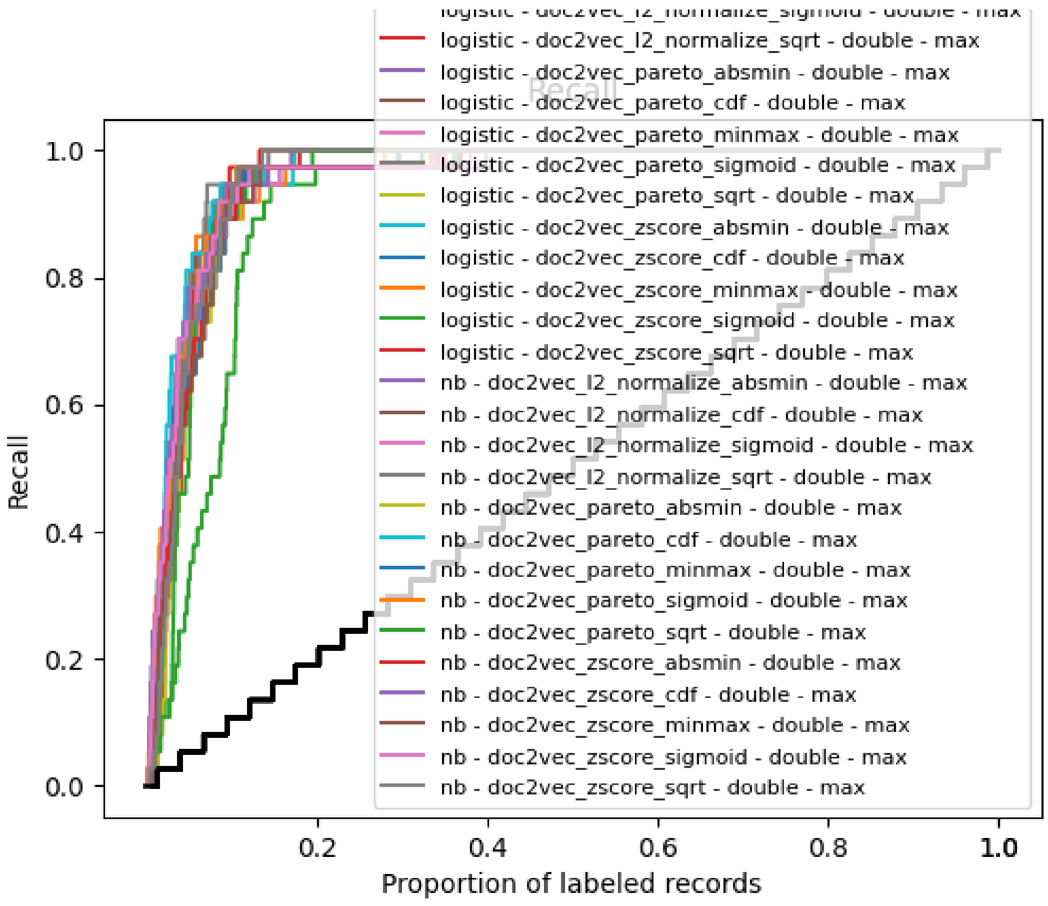

Figure A.4: All Doc2Vec combinations Recall Plot

A.3 TF-IDF

Combination	recall 0.1	wss 0.95	erf 0.1	atd	runtime	GPU(%)
(baseline) logstic - TF-IDF	0.973	0.894	0.865	117	65.8	0.00
(baseline) naïve bayes - TF-IDF	0.973	0.897	0.865	91	53.8	0.00
naïve bayes - l2 normalize - absmin	0.973	0.904	0.865	90	163.8	0.15
logstic - l2 normalize - absmin	0.973	0.901	0.865	105	234.8	0.17
NB - l2 normalize - sigmoid	0.973	0.899	0.865	108	241.5	0.32
logstic - sigmoid	0.973	0.899	0.865	109	732.0	0.04
naïve bayes - sigmoid	0.973	0.899	0.865	101	86.2	0.35
NB - l2 normalize - minmax	0.973	0.897	0.865	91	148.2	0.22
naïve bayes - absmin	0.973	0.897	0.865	91	147.0	0.19
naïve bayes - pareto - minmax	0.973	0.897	0.865	91	148.5	0.61
naïve bayes - z-score - minmax	0.973	0.897	0.865	91	148.5	0.61
logstic - l2 normalize - minmax	0.973	0.894	0.865	117	251.0	0.35
logstic - pareto - minmax	0.973	0.894	0.865	117	265.1	0.02
logstic - z-score - minmax	0.973	0.894	0.865	117	260.7	0.40
logstic - absmin	0.973	0.894	0.865	117	277.8	0.12
naïve bayes - pareto - sigmoid	0.973	0.890	0.865	85	79.9	1.37
logstic - l2 normalize - sigmoid	0.973	0.876	0.865	132	518.5	0.16
naïve bayes - cdf	0.973	0.867	0.865	115	89.1	0.82
naïve bayes - l2 normalize - cdf	0.973	0.867	0.865	115	79.4	1.26

Combination	recall 0.1	wss 0.95	erf 0.1	atd	runtime	GPU Util. (%)
naïve bayes - pareto - cdf	0.973	0.867	0.865	115	80.0	1.34
naïve bayes - z-score - cdf	0.973	0.867	0.865	115	80.5	1.54
logstic - cdf	0.946	0.844	0.838	133	1500.1	0.08
logstic - pareto - cdf	0.946	0.844	0.838	134	1271.1	0.13
logstic - z-score - cdf	0.946	0.844	0.838	135	1316.2	0.08
logstic - l2 normalize - cdf	0.946	0.844	0.838	138	1508.8	0.07
logstic - pareto - absmin	0.946	0.835	0.838	266	837.3	0.13
logstic - z-score - absmin	0.811	0.721	0.703	416	1090.3	0.12
naïve bayes - z-score - absmin	0.270	0.012	0.162	1622	1587.7	0.04

Note. The lowest score is an outlier, that's why two different values are selected as lowest value per column.

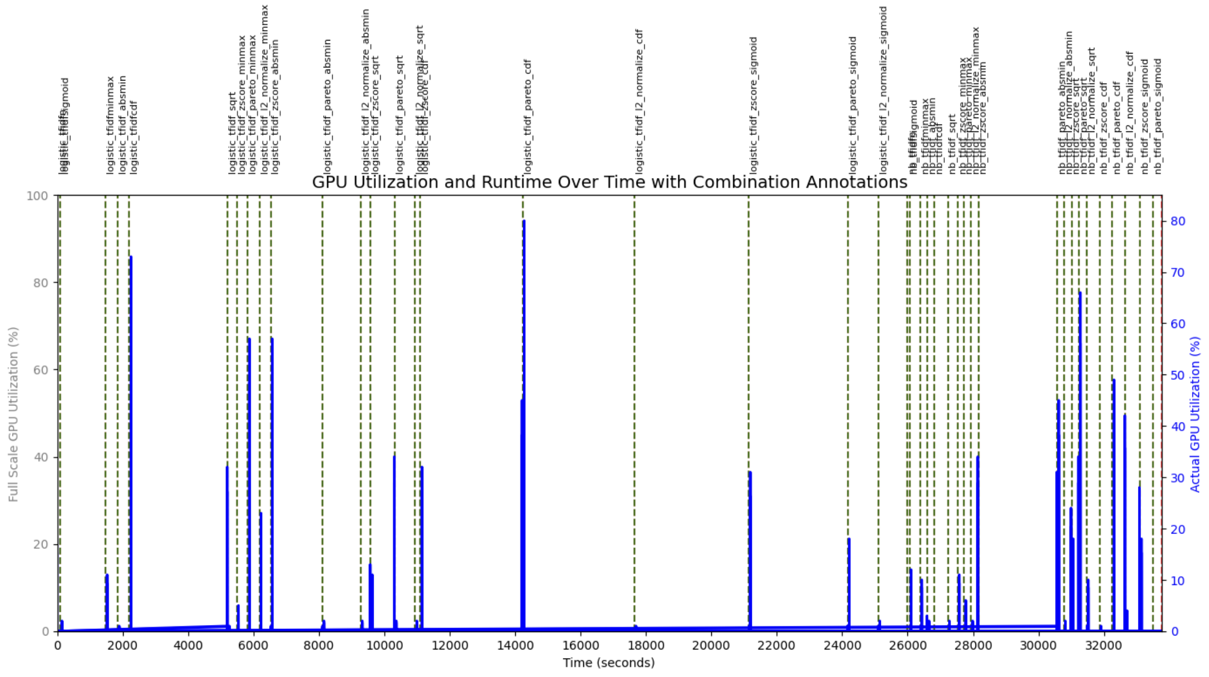

Figure A.5: All TF-IDF runtime and GPU utilization. There are big outliers, these combinations do not seem to go well together. Generally this also means lower WSS and even recall as seen in the table.

B. Visualization

This chapter shows all graphs that help to visualize the transformations of the feature embeddings.

B.1 SBERT

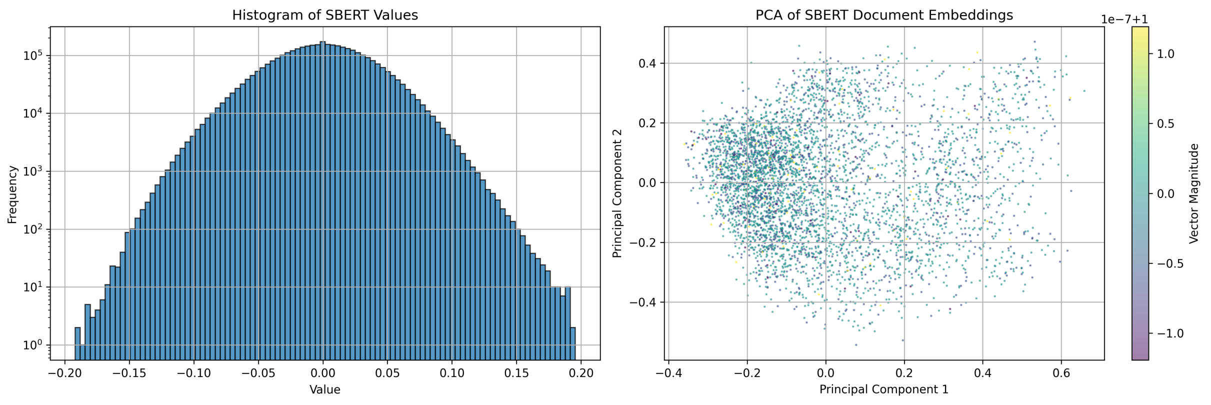

Figure B.1: Default SBERT.

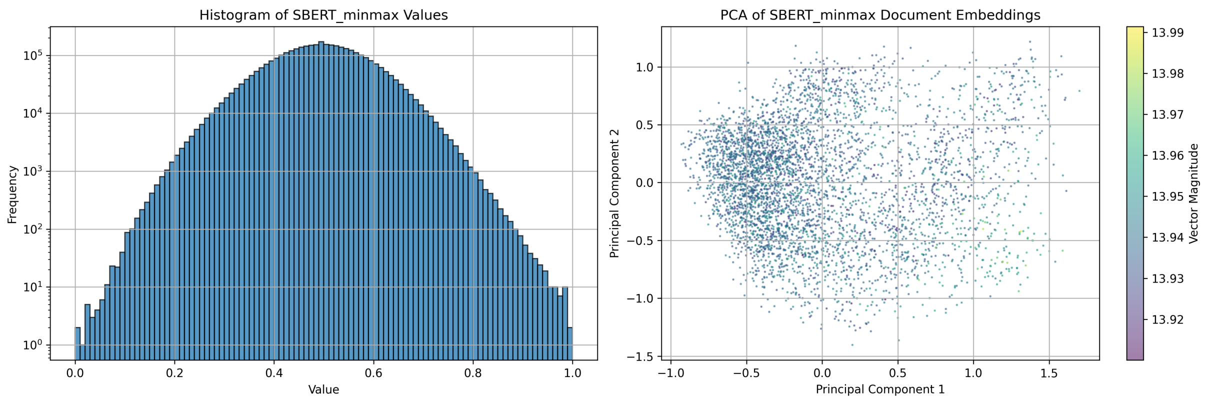

Figure B.2: SBERT with Min-Max Normalization.

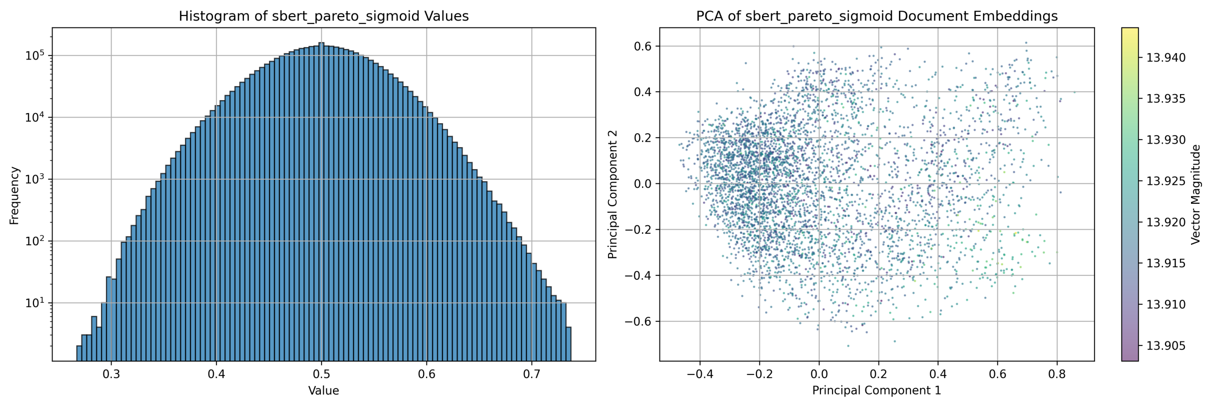

Figure B.3: SBERT shows less shape distortion than Doc2Vec.

B.2 Doc2Vec

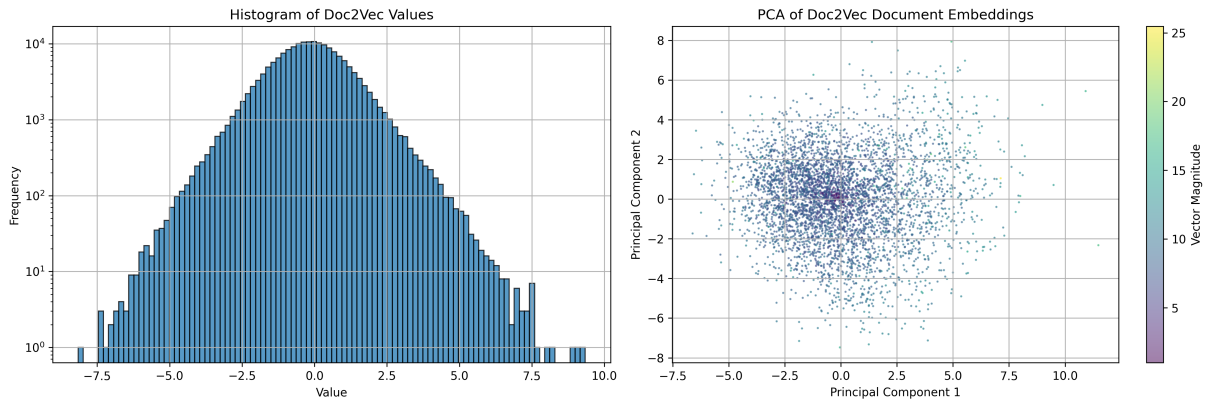

Figure B.4: Default Doc2Vec, no transformation.

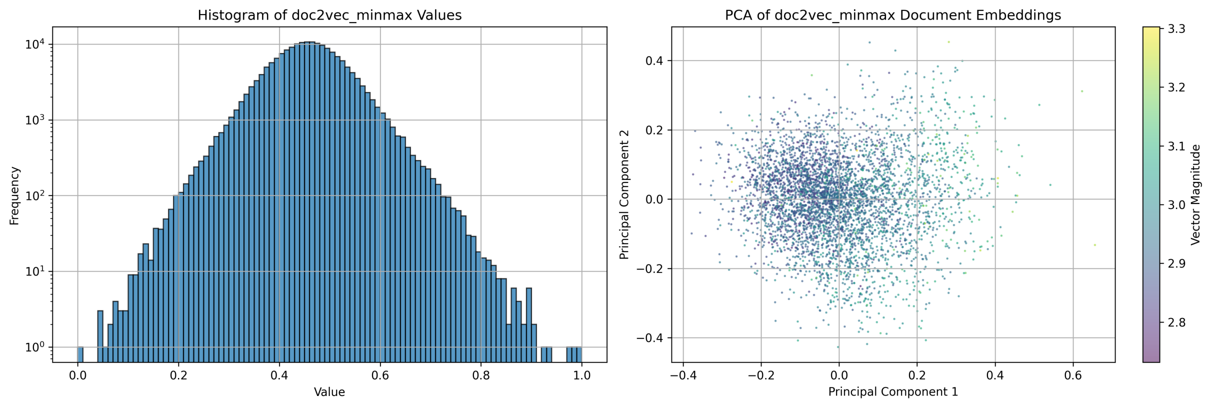

Figure B.5: Doc2Vec with Min-Max scaling. Note the x-axis. Where it first was between -7.5 and 10 is now 0 to 1 while the shape remained largely the same. Already a change in the PCA graph is visible.

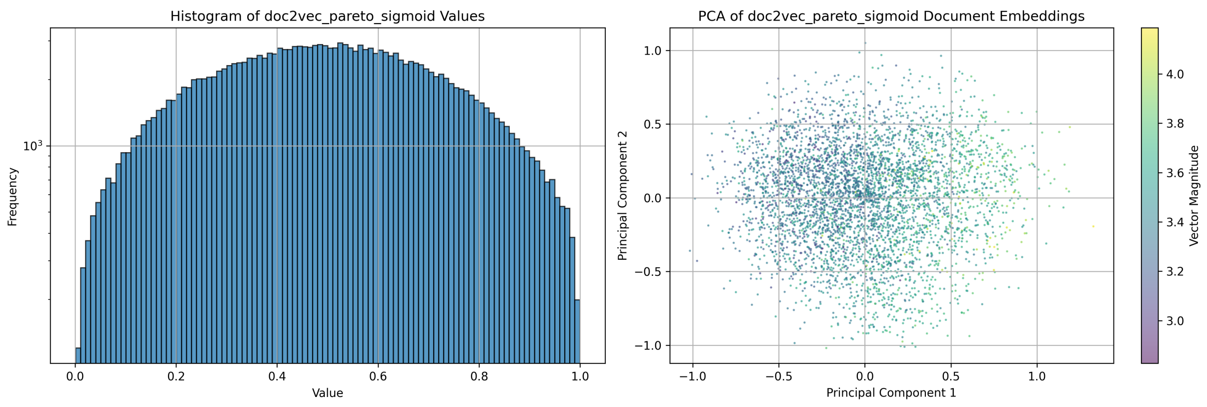

Figure B.6: A much rounder shape and much more equally distributed. The documents are also way more equally distributed in the PCA graph.

B.3 TF-IDF

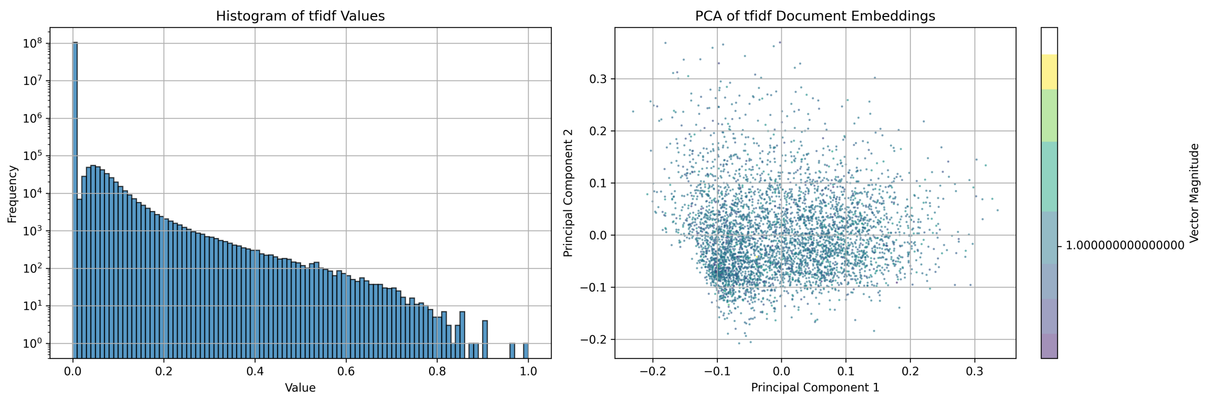

Figure B.7: Default TF-IDF embeddings. The high peak at zero shows the sparsity of TF-IDF embeddings.

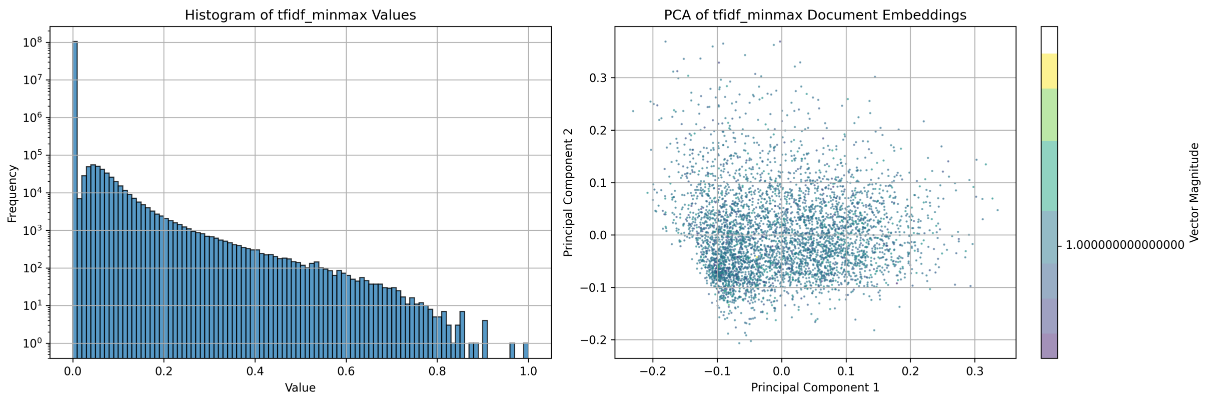

Figure B.8: Min-Max should change almost nothing for the already positive embeddings. And that is shown here.

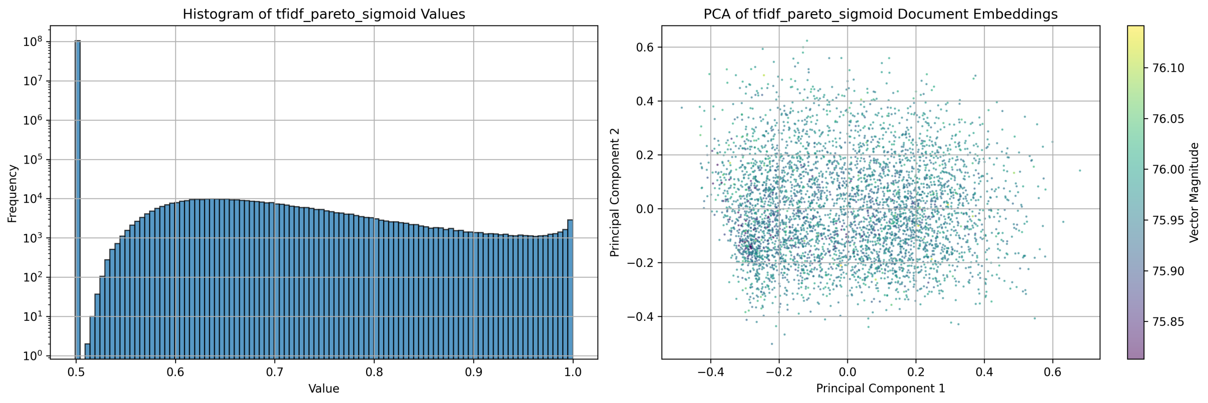

Figure B.9: Showing the "S" shape that logistic regression and sigmoid functions are known for. The PCA graph shows the embeddings are much more equally spread now.