Impact of Bucketing with BERT Embeddings on Model Performance across Training Set Sizes

Christian Harjuno† Victor Silaa † Yoko Nakajima†
Tomoyosi Akiba‡ Hirotoshi Honma‡
† Department of Computer Science, Kushiro College of Technology
p234071@kushiro.kosen-ac.jp, {silaa,yoko,honma}@kushiro-ct.ac.jp
‡ Department of Computer Science and Engineering, Toyohashi University of Technology
akiba@cs.tut.ac.jp

Abstract

We investigate a semantic bucketing strategy based on BERT embeddings to improve overall model generalization performance on scarce data. Samples are grouped by semantic similarity rather than structural similarity, such as length, and evaluated across varying training dataset sizes. Fine-tuned model performance is evaluated using accuracy, precision, recall, and loss. The results are then compared to a model trained on randomly sampled data. Our experiments show that bucketing consistently yields better metrics on smaller dataset sizes yet reveals higher losses on slightly larger data sizes, suggesting less confident probability estimates.

1 Introduction

Pre-trained language representation models, such as BERT, have influenced natural language processing by capturing contextual relationships across massive texts. However, despite their broad linguistic knowledge, these models must be fine-tuned on task-specific datasets to achieve strong downstream performance [5]. The effectiveness of fine-tuning depends heavily on the quality of the training data: datasets must be sufficiently large, balanced across classes, and representative of diverse edge cases to support generalization [19]. This study explores the effects of embedding-based semantic bucketing on model training efficiency. By partitioning datasets using token embeddings rather than token length, we aim to produce buckets that are more coherent in meaning rather than size. We investigate whether subsampling bucketed data, compared against randomized sampling, can improve model performance under varying dataset sizes.

2 Related Studies

Improving the representation of all edge cases and balancing the training data is critical when fine-tuning pre-trained encoders. One well-established method is *stratified sampling*, which preserves the label distribution across subsets and prevents the overrepresentation of classes. This has been shown to enhance generalization, particularly for minority categories [11].

Beyond label-aware sampling, researchers have also explored *bucketing strategies* to manage input variability. A common approach is bucketing by sequence length, which reduces variance in input size and stabilizes training dynamics [3]. However, Qu, Tu, and Bao (2024) show that length-based chunking yields high topic entropy and low completeness scores, indicating that such buckets mix semantically unrelated material [15].

To address this limitation, *embedding-based clustering* has been proposed. Petukhova et al. [14] show that grouping samples according to sentence embeddings yields semantically coherent clusters, improving coverage of diverse semantic phenomena during fine-tuning. Although less explored than length-based bucketing, this line of work suggests that embedding-driven partitioning may improve robustness in semantically diverse domains, such as social media text.

Based on these previous studies, our study examines if creating buckets of semantically similar samples before sampling evenly across buckets improves the fine-tuned model's generalization performance on smaller dataset sizes. While previous work focuses on comparing cluster quality across different models and bucketing algorithms, not much attention has been given to the effects of these clusters on model performance when only a small amount of data is available. Randomly sampling subsets from a semantically diverse dataset can introduce semantic bias, as some regions may be underrepresented, which potentially harms overall model generalization, particularly when sampled subsets are small[17]. Our study aims to address this gap by analyzing whether using embedding-based bucketing to create semantically balanced datasets reduces the amount of data needed to train a well-performing model. The model trained on bucketed data is compared with a model trained on randomly sampled data with varying set sizes. Both models are compared through classification metrics and evaluation loss.

3 Proposed Solution

To mitigate semantic bias in low-sample environments, we propose an embedding-based bucketing strategy to improve BERT fine-tuning on datasets with high semantic variability. First, we convert each text sample into a vector representation using pre-trained BERT embeddings. Next, we reduce the embeddings using UMAP and scale them with StandardScaler[1]. We then cluster the scaled embeddings using HDBSCAN[2] to form buckets of semantically similar examples. During fine-tuning, we sample a fixed number of examples from each bucket with stratified labels using the global (whole dataset) ratio.

This approach is compared against randomized sampling with approximately equal amounts of data. Unlike length-based methods, embedding-based bucketing produces semantically coherent batches, which we hypothesize will preserve semantic diversity within each batch.

3.1 Dimension Scaling

Peterfreund (2021) states that while the 768 dimensions in BERT embeddings allow subtle contextual signals to be captured, such high-dimensional embeddings introduce noise and increase computational complexity for clustering or bucketing tasks. Redundant or irrelevant features in these embeddings can negatively impact clustering performance [13].

To reduce complexity and preserve important semantics, we apply UMAP (Uniform Manifold Approximation and Projection) to project a higher dimension into a smaller dimension size while maintaining important information [10]. UMAP can reduce the dimension size while maintaining both local and global relationships.

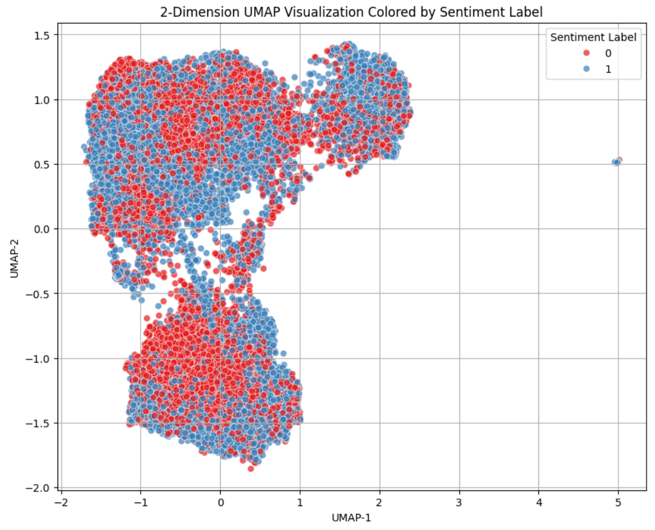
We observed that directly reducing 768-dimensional embeddings to 2 dimensions does not yield visually distinguishable clusters. However, introducing an intermediate reduction to 50 dimensions before projecting to 2 dimensions produced clearer and better-separated clusters, which will facilitate downstream clustering with algorithms such as HDBSCAN, as shown in Figure 1.

The quality of clusters can be further quantified using several metrics. To measure the cohesion of points within clusters and their separation from other clusters, we utilize the Silhouette Score [16]. The Davies-Bouldin Score is employed to assess the ratio of intra-cluster scatter to inter-cluster separation [4]. Finally, the Calinski-Harabasz Score measures the ratio of between-cluster variance to within-cluster variance [1]. Together, these metrics are able to quantify how cohesive and well-separated the buckets are from each other. A well-separated bucket reduces the chances of data points falling near bucket boundaries, which signify poor clustering.

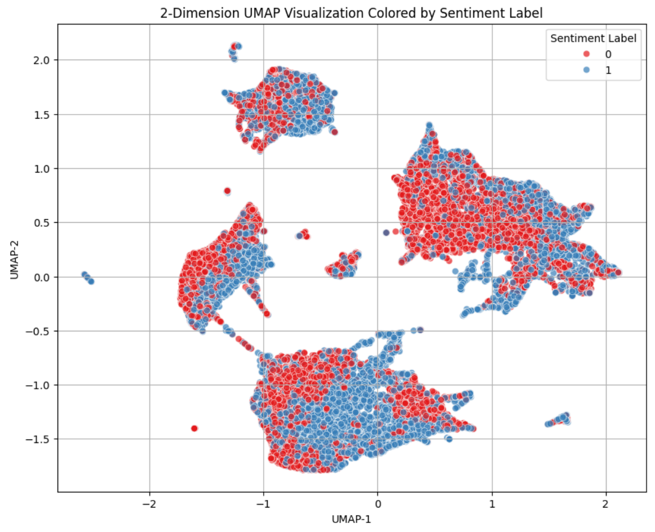
As shown in Table 1, introducing an intermediate reduction to 50 dimensions before projecting to 2 outperforms a direct reduction from 768 to 2 across all clustering metrics. These metrics collectively demonstrate that the intermediate reduction may produce more coherent and better-separated clusters. The resulting clusters obtained via HDBSCAN are visualized in Figure 2.

As shown in Figure 2, introducing an intermediate reduction to 50 dimensions results in less noise (highlighted in purple) compared to a direct reduction from 768 to 2 dimensions.

Based on the observations from the UMAP reductions and HDBSCAN clustering, we will adopt the two-step dimensionality reduction approach — first reducing BERT embeddings from 768 to 50 dimensions, followed by a further reduction to 2 dimensions—for all subsequent clustering experiments.



(a) UMAP Reduction from 768 to 2



(b) UMAP Reduction from 768 to 50, then 2

Figure 1: Comparison of UMAP dimensionality reduction strategies.

This strategy has been shown to potentially produce more semantically coherent clusters, reduce noise, and improve cluster separation, thereby providing a more reliable basis for embedding-based bucketing in downstream fine-tuning tasks. However, the general effectiveness of this strategy is still unclear and needs further testing before it can be fully implemented in the current workflow.

3.2 Clustering with HDBSCAN

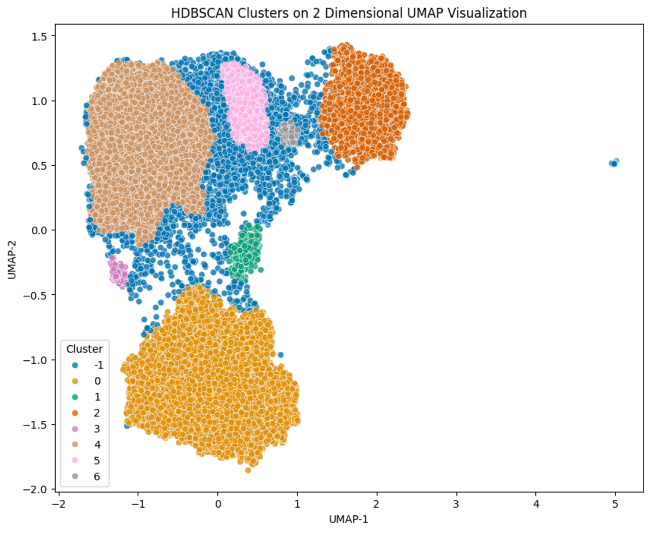
Traditional clustering methods often use K-Means to partition vectors into clusters. While K-Means allows specifying a fixed number of clusters, it assumes that clusters are roughly spherical and of similar size [8]. However, BERT embeddings have been shown to be anisotropic, occupying a narrow cone in vector space [7], making K-Means unsuitable

[1]`https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

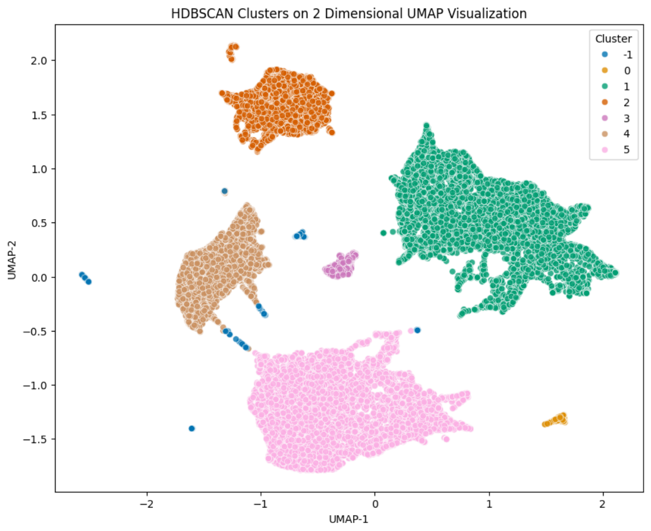
[2]`https://scikit-learn.org/stable/modules/generated/sklearn.cluster.HDBSCAN.html`

for this type of data.

In contrast, HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) [2] can identify clusters of arbitrary shapes and dynamically determine the number of clusters based on the data. Additionally, HDBSCAN can handle noise points effectively, leaving them unassigned to any cluster, which is particularly useful for noisy embeddings.
This makes HDBSCAN particularly suitable for creating semantically coherent buckets from BERT embeddings.



(a) HDBSCAN clusters after UMAP reduction from 768 to 2 dimensions



(b) HDBSCAN clusters after UMAP reduction from 768 to 50, then to 2 dimensions

Figure 2: Visualization of HDBSCAN clustering performance under different UMAP reduction strategies

4 Data

A few considerations went into determining which public dataset will be used in this experiment. Having a clean, balanced, and accurate dataset can help in reducing variability, which affects our final results. Datasets such as the Stanford IMDB dataset (stanfordnlp/imdb) [9] are popular due to their simplicity, cleanliness, and balance. However, the goal of bucketing is to provide the model with a balanced representation of all semantic edge cases in the dataset. If we apply the clustering techniques previously explained in Section 3, we can visualize the clusters as shown in Figure 3

Table 1: Clustering metrics for different UMAP reduction strategies. Arrows indicate whether higher (↑) or lower (↓) values are better.

Metric	**768 → 50 → 2**	**768 → 2**
Silhouette Score (↑)	0.5002	0.3785
Davies-Bouldin Score (↓)	0.4352	0.4817
Calinski-Harabasz Score (↑)	46558.54	41264.99

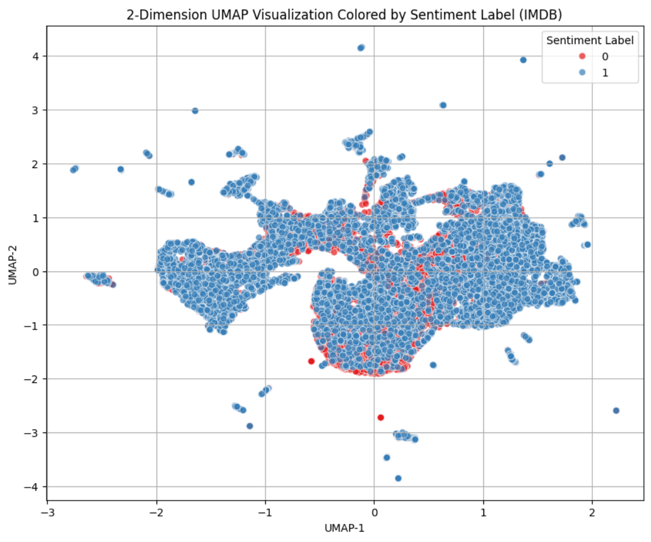


Figure 3: stanfordnlp/imdb embeddings visualized

Due to low semantic variability in the IMDB Reviews dataset, embeddings are tightly packed, forming a single dense cluster as shown in Figure 3. This makes downstream clustering with HDBSCAN more challenging. In contrast, the Amazon Polarity dataset from the Massive Text Embedding Benchmark (MTEB) [12] exhibits clearly separated and well-defined clusters, as illustrated in Figure 2b. On the other hand, while Figure 2a shows clusters that are very close to each other, the edges of the clusters are poorly defined, which causes a lot of data points to be considered as noise.

The higher variability and richer cluster structure allow us to evaluate whether sampling from embedding-based buckets can improve model performance under different dataset sizes.

Based on these considerations, we select the Amazon Polarity dataset from MTEB because it exhibits higher semantic variability compared to datasets like IMDB Reviews. This variability ensures that the buckets created via UMAP reduction and HDBSCAN contain meaningful semantic differences, providing a robust testbed to measure the impact of bucketed sampling on downstream model fine-tuning.

Due to time and resource restraints, we acknowledge that testing the proposed method against a single dataset significantly reduces the generalization of this study. As previously mentioned, different datasets reveal different semantic structures which are still important in properly evaluating the effectiveness of this semantic-based bucketing method. Future work regarding this topic should consider a wide range of datasets with different qualities and semantic variability.

5 Experiments

5.1 Experimental Setup

The objective of this experiment is to explore whether embedding-based clustering impacts the performance of a model across different training set sizes. To properly isolate the effect of the bucketing itself, we utilize the Amazon Polarity Dataset, which is part of MTEB. The dataset is obtained from the HuggingFace repository at `mteb/amazon_polarity`[3].

The Amazon Polarity Dataset contains 3.6 million rows of training data and 400 thousand rows of evaluation data, which is unnecessary for our objectives. Both training and evaluation sets are randomly subsampled with a fixed seed (42), yielding 36,000 training examples and 400 evaluation examples. From this subsample, we calculate the global label ratio between the positive and negative examples, shown in Table 2.

Table 2: Distribution of sentiment labels in the subsampled Amazon Polarity dataset

Sentiment Label	Count	Ratio
Negative (0)	17,911	49.8%
Positive (1)	18,089	50.2%
Total	36,000	100%

The training set is first tokenized using a BERT tokenizer. Embeddings are generated and reduced to 50 dimensions, and subsequently to 2 dimensions, using UMAP. The reduced embeddings are standardized with a `StandardScaler`. This process produces seven buckets (six valid buckets and one noise bucket). The noise bucket is discarded and excluded from further training.

We then construct the training set through the following procedure:

1. Define a target training dataset size T.
2. Divide T by the number of valid buckets (excluding the noise bucket) to obtain the number of samples to be drawn per bucket.
3. For each bucket:
   (a) Compute the number of positive and negative samples to draw by applying the global label ratio (Table 2) to the bucket's sample budget.
   (b) Randomly sample examples from the bucket according to these counts.

This procedure ensures that (i) all buckets contribute equally to the final dataset, and (ii) the global class balance is preserved across the sampled training set. Due to the difference in label counts within each bucket, it is possible for the total training dataset to not reach the previously defined training dataset size T

[3]The dataset is hosted at `https://huggingface.co/datasets/mteb/amazon_polarity`

5.2 Model and Training Details

We employ `BERT-base-uncased` as the backbone model [6], obtained from the HuggingFace repository. [4] A linear classification layer is added on top of the pooled [CLS] representation.

The model is fine-tuned for three epochs using the AdamW optimizer with a learning rate of 3×10^{-5}, linear decay scheduling, and a 10% warmup. Training is performed with a per-device batch size of 4 and gradient accumulation over 8 steps (effective batch size of 32), with weight decay set to 0.01. Training is conducted with eval_accuracy as the primary criterion, with the best-performing model loaded at the end.

All experiments are executed on an Apple M3 processor. To leverage the M3's MPS support, the model is configured with `bfloat16` to allow more efficient tuning.

5.3 Bucketing Method

As described in Section 3.1, the 768-dimensional BERT embeddings are first reduced to 50 dimensions and then to 2 dimensions using UMAP with a seed of 42 for reproducibility. To improve clustering stability for HDBSCAN, the resulting embeddings are scaled using a `StandardScaler` [18]. HDBSCAN is then applied with a minimal cluster size of 200. While a common heuristic is to set 'min_cluster_size' to approximately 1–5% of the dataset (based on informal practice), in our experiment this parameter was tuned to generate the same number of clusters as observed in Figure 2b.

We acknowledge that this method of choosing the minimal cluster size is guided by the data itself and introduces potential biases that skew our results. Future developments of this method should consider a more concrete parameter selection strategies that are independent of the visually observed clusters using evaluation metrics such as Silhouette Scores, Davies-Bouldin Scores, etc.

The resulting clusters (buckets) are then used for stratified sampling as described in Section 5.1.

5.4 Evaluation Metrics

In this experiment, we evaluate the effect of embedding-based bucketing on model performance across multiple training set sizes. For each training set size, we compute accuracy, precision, recall, and F1-score. For each training size, the model is trained on two types of datasets: one created using embedding-based bucketing and another with random sampling (seed 42). The trained model is then used to predict on a fixed evaluation set of 400 examples. The results of which are analyzed to quantify both general and class-wise performance using previously mentioned metrics.

For each target training set size, the model is evaluated, and the resulting metrics are compared with progressively larger dataset sizes.

5.5 Experiment Variants

The main objective of this experiment is to evaluate whether embedding-based clustering influences model performance across varying training set sizes. To properly gauge how bucketing affects model performance, we repeated the

[4]`https://huggingface.co/google-bert/bert-base-uncased`

bucketing and sub-sampling process to generate training sets ranging from 100 to 6000 examples, increasing in increments of 100. For each increment (100, 200, 300, ...), we fine-tune the base bert-base-uncased model and evaluate the trained model's performance with the aforementioned evaluation dataset of 400.

6 Results Evaluation

Evaluating the performance of both the randomized model and bucketed model using the previously mentioned metrics in Section 6, all metrics show that the bucketed model's performance consistently outperforms the randomized model at low sample counts, as seen in Figures 4, 5, and 6.

To properly gauge the overall performance of each model on different sample sizes, we calculate the rolling average of each performance metric with a rolling window of 3. Convergence between the bucketed and randomized metric is calculated when their rolling averages reach a convergence threshold of 0.001. Performance stability (when large improvements are not seen over a period of sample counts) is reached when the convergence threshold is achieved 5 consecutive times.

6.1 General Performance Advantage

In terms of overall accuracy, the bucketed model consistently outperforms the randomized model up to approximately 1000 samples, after which their performances begin to converge and gradually become similar (Figure 4). This indicates that the bucketing method successfully constructs a more balanced and diverse dataset, enabling the model to generalize more effectively across different types of inputs.

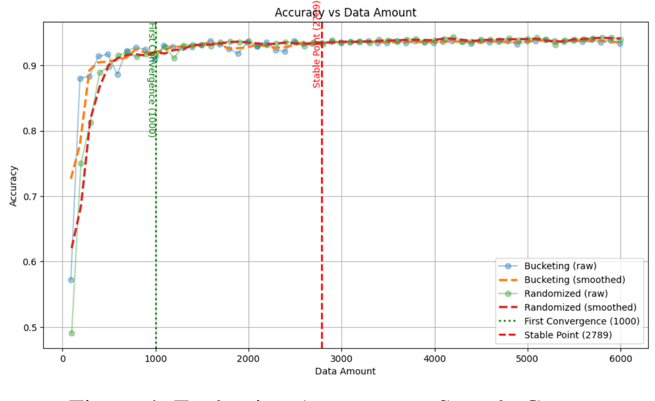


Figure 4: Evaluation Accuracy on Sample Count

The benefits of bucketing are further reflected in model precision (Figure 5), where the bucketed model consistently outperforms the randomized model in low-sample regimes. However, unlike accuracy, precision convergence occurs much earlier—at around 689 samples. This suggests that bucketing accelerates the model ' s ability to establish class boundaries and reduce false positives by providing a semantically well-structured dataset. Since precision depends only on the ratio of true positives to false positives, improvements plateau once these boundaries are learned. In contrast, accuracy, which accounts for both true positives and true negatives, requires larger sample sizes to fully converge, leading to the later stabilization observed.

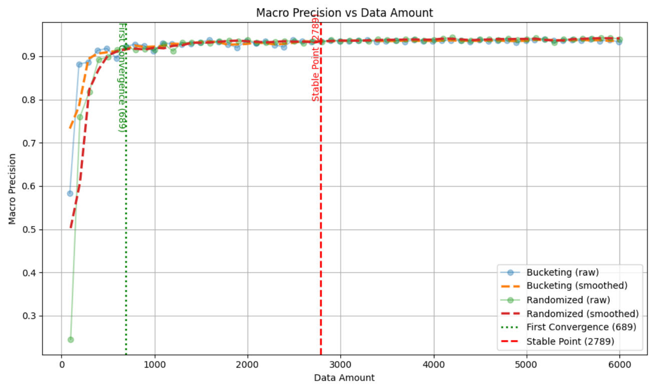


Figure 5: Evaluation Precision on Sample Count

Interestingly, recall converged much earlier than both precision and accuracy, stabilizing at approximately 600 samples. This indicates that the bucketed dataset enabled the model to capture a broad spectrum of positive instances with relatively few examples, thereby reducing false negatives quickly. In contrast, precision required more samples to stabilize, reflecting the additional challenge of avoiding false positives. Accuracy, which depends on both positive and negative classifications, converged latest, requiring the largest sample sizes to balance both aspects.

6.2 Evaluation loss

Examining the overall evaluation loss of both models highlights some potential limitations of the embedding-based bucketing method. While accuracy, precision, recall, and F1 show that the bucketed model consistently outperforms the randomized model in low-sample regimes, its behavior changes as the dataset grows. As shown in Figure 7, the bucketed model initially achieves lower loss until approximately 490 samples, after which it experiences higher loss than the randomized model until roughly 2000 samples. This indicates that, while the bucketed model maintains comparable performance in discrete metrics, it is less confident in its predictions, assigning higher probability mass to incorrect classes.

By the time the losses stabilize and converge (2000 samples), the benefits of bucketing have largely diminished, as also reflected in the evaluation metrics (Figures 4, 5, 6). Bucketing is designed to ensure the model sees representative examples early, but it can create " easy " and " hard " buckets. Easier buckets, containing simpler or more homogeneous samples, are learned quickly, whereas harder buckets, containing more complex or heterogeneous samples, require more data to model accurately. Since each bucket is sampled equally, this can temporarily reduce model confidence for harder samples, resulting in higher loss at intermediate dataset sizes (500 - 2000).

While previous experiments focus on the effect of the implemented bucketing method on the model's performance itself, in this experiment, we will try to evaluate how effectively the implemented method separated our dataset. As previously mentioned in Section 5.1, the implemented bucketing method against the `mteb/amazon_polarity` results in 7 buckets, which includes 1 noise bucket. To properly evaluate whether the bucketing method actually created a meaningful split in the dataset, we will be training the same `bert-base-uncased` model with another bucket and then evaluating using another. The results of which will be compared to a model trained and evaluated with the same bucket (with a training and evaluation split).

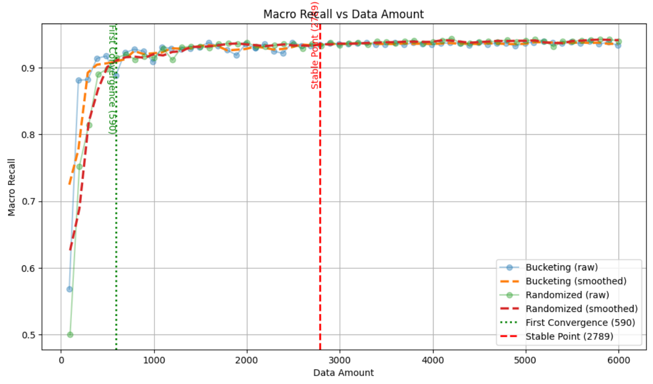


Figure 6: Recall on Dataset Count

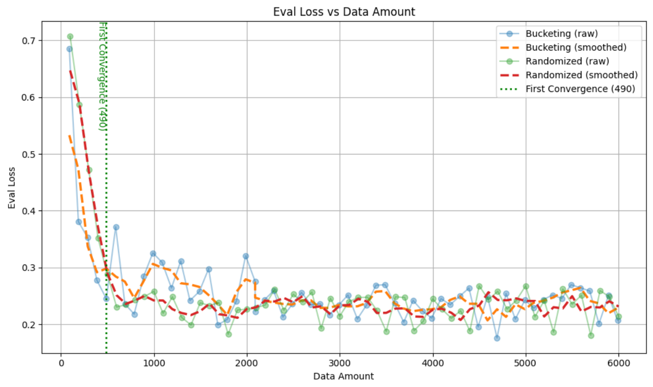


Figure 7: Evaluation loss On Sample Count

6.3 Cross-Bucket Generalization

The results of the buckets created using the previously implemented method are shown in Table 3. The bucket with the index -1 represents the noise bucket, which is discarded and not used for model training. For the cross-bucket generalization experiment, we train a model on the bucket with index 4 and evaluate it on the bucket with index 2.

For comparison, we also train and evaluate a model entirely within bucket 2, using a training/evaluation split of 0.15. The hierarchical split within bucket 2 for this within-bucket experiment is summarized in Table 4.

The results of the experiment can be seen in Table 5. Although training a model on bucket 4 and evaluating it on bucket 2 still achieved relatively high performance (with higher losses), training and evaluating within the same bucket consistently produced better results across all metrics. This indicates that the implemented bucketing method successfully split the original dataset into meaningful strata, potentially allowing the model to learn more effectively from each subset and generalize better on much more heterogeneous datasets.

7 Conclusions

This study shows promising results that embedding-based bucketing can reduce the sample size required to achieve well-performing models in extremely low-resource scenarios. Traditionally, datasets must be predetermined and manually labeled in large quantities to ensure proper generalization, which is time-consuming. By applying the semantic bucketing method, one can select representative samples from unlabeled data, potentially reducing data labeling effort while still maintaining good model generalization. Of course, any method to reduce the amount of data needed won't beat training on as much data as possible.

Table 3: Sizes of clusters after HDBSCAN clustering. The noise bucket, containing points not assigned to any cluster, is indexed as -1.

Bucket Index	**Size**
-1	253
0	246
1	13,444
2	4,767
3	699
4	4,841
5	11,750

Table 4: Hierarchical Data Split for Bucket 2 (4767 samples)

Dataset	**Samples**	**Percentage of Total**
Training Set	3647	76.5%
Training Evaluation (Internal Validation)	405	8.5%
Final Evaluation Set	715	15.0%
Total	4767	100%

Table 5: Cross-Bucket vs Within-Bucket Evaluation Metrics

Method	Eval Loss	Eval Accuracy	Macro F1	Macro Precision
Bucket 4 $\rightarrow$ 2	0.2750	0.9287	0.9221	0.9185
Bucket 2 $\rightarrow$ 2	0.1635	0.9385	0.9315	0.9274

In addition, the niche bucketing method implemented was successful in creating meaningful splits in the dataset. However, the implemented methods still remain under-researched, with open questions regarding the most optimal bucketing algorithms, hyper-parameter selection, efficiency, robustness, and reliability. As previously mentioned, different dataset characteristics may influence the overall effectiveness of the bucketing approach. Nevertheless, the implemented method may prove beneficial in other unexplored areas such as domain adaptation, few-shot learning, or cross-lingual tasks, where structured sample sampling could enhance model generalization on unseen data.

This further emphasizes the need for more rigorous research before embedding-based bucketing can be considered a worthy substitute for more trusted methods such as label-based stratified sampling or embedding-size-based bucketing. Future work should explore these directions to assess whether embedding-based bucketing can mature into a reliable approach for practical applications.

Contribution. We investigate the use of a bucketing strategy that utilizes BERT representations to construct more semantically balanced subsets for training in a low-sample environment, aiming to reduce semantic bias and lower the amount of labeled data required. To our knowledge, this specific application of embedding-based bucketing for bias mitigation in a small-sample environment has not been systematically studied.

References

[1] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, Vol. 3, No. 1, pp. 1–27, 1974.

[2] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pp. 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[3] Tri Dao, Albert Gu, and Alexander Rush. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2406.08128*, 2024.

[4] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, No. 2, pp. 224–227, 1979.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, Vol. abs/1810.04805, , 2018.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, 2019.

[7] Kawin Ethayarajh. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 55–65, Hong Kong, China, November 2019. Association for Computational Linguistics.

[8] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, Vol. 622, pp. 178–210, 2023.

[9] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[10] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[11] Maximillian Merrillees and Lan Du. Stratified sampling for extreme multi-label data, 2021.

[12] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[13] E. Peterfreund. Multidimensional scaling of noisy high-dimensional data. *Computational Statistics & Data Analysis*, Vol. 154, p. 106352, 2021.

[14] Alina Petukhova, João P. Matos-Carvalho, and Nuno Fachada. Text clustering with large language model embeddings. *International Journal of Cognitive Computing in Engineering*, Vol. 6, pp. 100–108, 2025.

[15] Renyi Qu, Ruixuan Tu, and Forrest Bao. Is semantic chunking worth the computational cost? *arXiv preprint arXiv:2410.13070*, 2024.

[16] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, Vol. 20, pp. 53–65, 1987.

[17] Deven Santosh Shah, H. Andrew Schwartz, and Dirk Hovy. Predictive biases in natural language processing models: A conceptual framework and overview. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5248–5264, Online, July 2020. Association for Computational Linguistics.

[18] C. Wongoutong. The impact of neglecting feature scaling in k-means clustering. *PLoS One*, Vol. 19, No. 12, p. e0310839, December 2024.

[19] Dylan Zhang, Justin Wang, and Francois Charton. **Only-IF**:revealing the decisive effect of instruction diversity on generalization, 2024.