Report on Autoencoder and Gaussian Mixture Model

Group 2

April 17, 2025

Introduction

In this assignment, we explore the practical application of autoencoder and guassian mixture clustering on the arxiv journal embeddings we generated for the group project. We visualized the results of the autoencoder and guassian mixture model using t-SNE and evaluate their clustering performance. We've found that the autoencoder is able to learn a compact representation of the data. However, non-neural network dimensional reduction method like U-MAP outperforms it in the context of our dataset.

The Arxive Dataset and Embeddings Generation

We obtained the dataset by fetching papers based on multiple curated lists of influential AI/ML papers and perform a manual categorization of the papars. The original categorization are the followings: classic machine learning, optimization, neural network foundation, computer vision, reinforcement learning, representation theory, natural language processing, generative model. However, the number of papers in each general category is not balanced, categories like computer vision and natural language processing have a lot more papers than optimization and reinforcement learning. To fix this, we merged a few categories together. The final categories have the following counts: machine learning general (89), NLP (56), Computer Vision Pattern Recognition (53), Computer Vision Generative Model(53), Recurrent Neural Network (36), Audio (25), Reinforcement Learning (18).

We use pre-trained Sentence-BERT models to generate the embeddings. Based on the MTEB benchmarks [1], we selected the 2 of the top models that score high metrics and have less than 1B parameters. The models are: gte-multilingual-base[3] and jasper-vision-language [2] and the embedding dimension for each model is 768 and 1024 respectively. We've evaluated the performance of the embedding models in our group project and found that the jasper-vision-language slightly outperforms the gte-multilingual-base model in terms of clustering performance, measured with calinski-harabasz index, davis-bouldin index and visual inspection of clusters. Therefore, we will use the jasper-vision-language model for the rest of the assignment.

Autoencoder

We implemented a simple autoencoder with ability to adjust the encoder and decoder layer. We're unsure what the best architecture is, so we tried a few different architectures. We picked 50 as our latent dimension.

Intermediate Dimension
768
768, 512, 256
768, 512, 256, 128

The ratinonale behind the configuration is to have the information being compressed without losing information so we set the middle layers to have a gradual decrease in dimension.

For each architecture, we apply 5-fold cross validation. We trained the autoencoder for 100 epochs with a batch size of 128. We set a early stopping criteria: if the validation loss does not improve for 10 epochs, we stop training. This is to prevent overfitting. We used an ReduceLROnPlateau callback to reduce the learning rate if the validation loss does not improve for 5 epochs. We used Adam optimizer with a learning rate of 0.001. We used Mean Squared Error as our loss function. However, upon training the model, we found that the decoded embeddings are distributed way more closer than the original embeddings. We hypothesize that this is due to the fact that the MSE loss function is not sensitive to the scale of the data. Therefore, we scaled the MSE by 1000 during training.

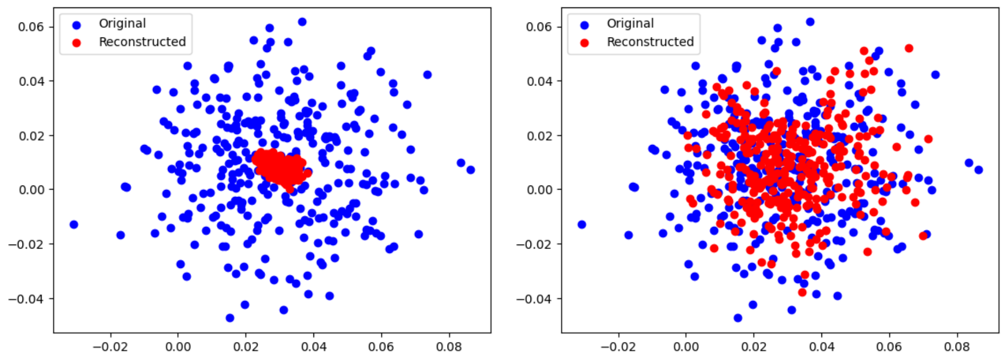

Figure 1: Left: Decoded Embeddings with MSE, Right: Decoded Embeddings with scaled MSE

The validation loss stops decreasing after 30 epochs for all configurations.

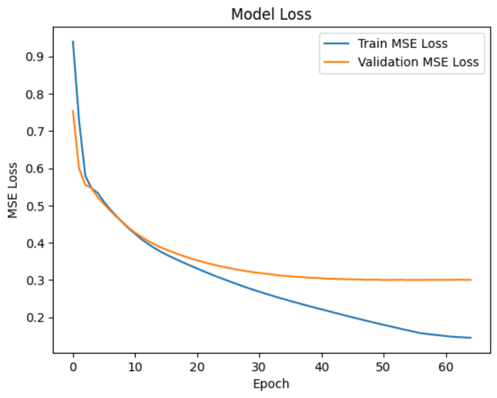

Figure 2: Loss plot for 768

We evaluate the performance of Autoencoder by first compare the clustering performance of the decoded embeddings with the original embeddings. We use KMeans clustering with 7 clusters and evaluate the performance using calinski-harabasz index, davis-bouldin index.

Model	DB score	CH score
KMeans on latent space(50)	1.922802	29.529562
KMeans on original embeddings	2.902679	13.327391
KMeans on UMAP embeddings(50)	0.536476	454.550720
KMeans on UMAP embeddings(25)	0.668492	352.116394
KMeans on UMAP embeddings(25) from latent space(50)	0.719458	402.341339

The table above shows the clustering performance of the autoencoder and UMAP embeddings. We can see that the autoencoder is able to learn a some representation of the data and possibly reduce the noise if we compare the clustering performance of the decoded embeddings with the original embeddings. However, the performance of the autoencoder is not as good as the UMAP embeddings. The UMAP embeddings are able to capture the structure of the data and produce better clustering performance. We also tried to use the UMAP embeddings from the latent space of the autoencoder, but it does not improve the performance.

We then visualize the results of the autoencoder and UMAP embeddings using t-SNE. We use the same parameters for t-SNE as in our project. We set the perplexity to 20 and the number of iterations to 1000. We use the same color scheme as the original dataset. The t-SNE plot shows that the autoencoder is able to learn a compact representation of the data when compared to the original embeddings. However, the UMAP embeddings are able to capture the structure of the data better than the autoencoder. The UMAP embeddings are able to separate the clusters better and produce a more compact representation of the data.

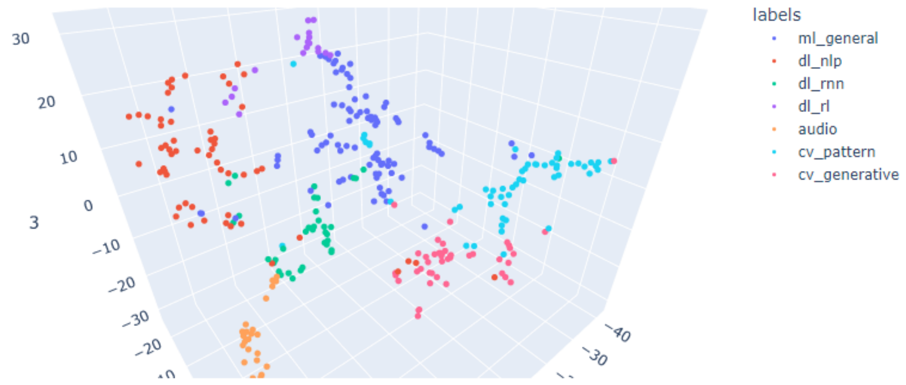

Figure 3: 3D plot of original embeddings with t-SNE, labels are manually assigned based on the original dataset.

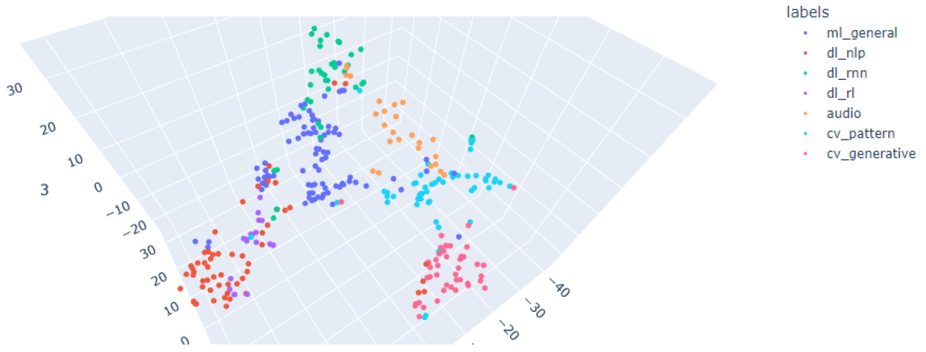

Figure 4: 3D plot of decoded embeddings with t-SNE, labels are manually assigned based on the original dataset.

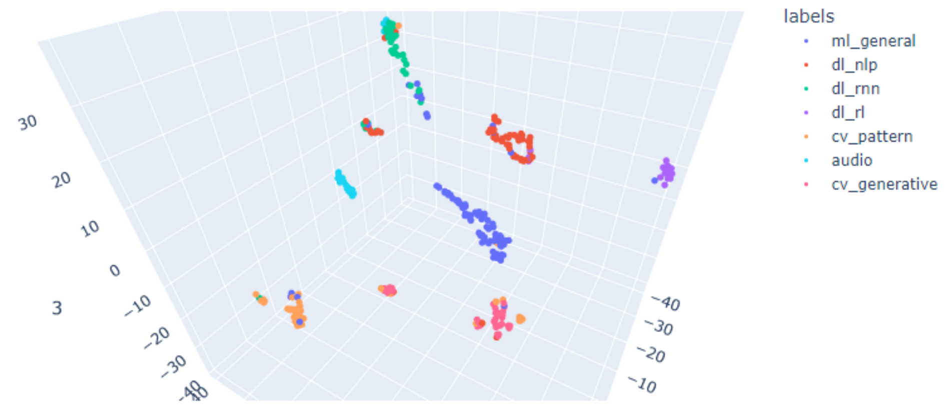

Figure 5: 3D plot of umap-transformed embeddings with t-SNE, labels are manually assigned based on the original dataset.

Gaussian Mixture Model and visualization

We now move on to the dicussion of Gaussian Mixture Model. We apply the GMM to the latent space of the autoencoder. We use the same architecture as the autoencoder. We set the number of clusters to 10. We use the GMM with full covariance matrix. The reason we use 10 clusters instead of 7 is because we want to see if the GMM can capture the structure hiden during the merging operation.

To our surprise, the GMM is able to capture the structure of the data and produce better clustering on certain part of the data than what the tSNE plot shows. For example, in the tSNE plot, the two purple points at the bottom right are visually close to the blue points that are mostly computer vison paper. But a manual check shows that they should be classified as RNN papers. The GMM is able to capture this structure and produce a better clustering performance.

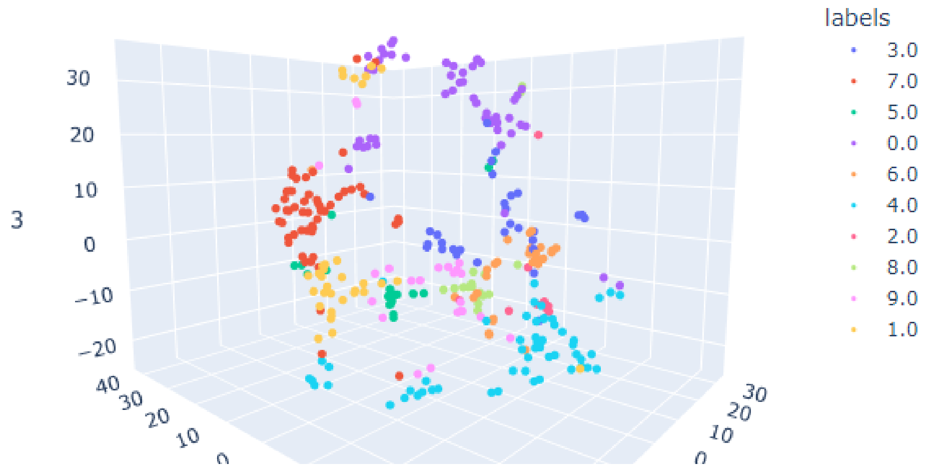

Figure 6: 3D plot of GMM-clustered data with t-SNE, labels are automatically generated by the GMM.

The above table shows the clusters and 2 randomly sampled papers. The GMM successfully separated the representational theory, classic machine learning papers, as well as the volumetric modeling papers which we didn't even considered as a group. And compare with agglomerative clustering we used as the optimal

Cluster	Paper1	Paper2
0	face recognition using eigenfaces	learning hierarchical features for scene labeling
1	learning representations by back propagating errors	highly accurate protein structure prediction with alphafold
2	context dependent pre trained deep neural networks for large vocabulary speech recognition	wavlm large scale self supervised pre training for full stack speech processing
3	tensorf tensorial radiance fields	3d gaussian splatting for real time radiance field rendering
4	nearest neighbor pattern classification	no free lunch theorems for optimization
5	hybrid computing using a neural network with dynamic external memory	photo real talking head with deep bidirectional lstm
6	imagenet classification with deep convolutional neural networks	visual prompt tuning
7	mathematical discoveries from program search with large language models funsearch	zero memory optimizations toward training trillion parameter models
8	human level control through deep reinforcement learning	outracing champion gran turismo drivers with deep reinforcement learning sophy
9	neural word embedding as implicit matrix factorization	self organized formation of topologically correct feature maps

Table 1: Clusters and their representative papers

cluster in the project, the GMM is not any inferior to that.

Reconstruction Analysis

A key feature of autoencoders is their ability to compress data into a low-dimensional latent space and then reconstruct it back to the original dimensionality. In this section, we evaluate the reconstruction capability of our autoencoder and analyze which samples are difficult to accurately reconstruct.

Reconstruction Loss Evaluation

We used our trained autoencoder model to reconstruct all samples and calculated the reconstruction error. The reconstruction error is computed using Mean Squared Error (MSE), representing the difference between the original embedding vectors and the reconstructed vectors.

Our model achieved an average reconstruction error of 0.018250 across the entire dataset, indicating that the autoencoder can reconstruct most samples relatively accurately. The following figure shows the distribution of reconstruction errors:

From the figure, we can see that most samples have reconstruction errors concentrated between 0.010 and 0.025, showing a near-normal distribution, with a few samples having significantly higher than average errors. This suggests that the autoencoder performs well in learning representations for most of the data but has limited reconstruction capability for certain special samples.

Analysis of Poorly Reconstructed Samples

To better understand the limitations of our autoencoder, we identified and analyzed poorly reconstructed samples. We defined samples with reconstruction errors greater than 0.025285 as "poorly reconstructed samples," totaling 32 such samples.

Through analyzing these poorly reconstructed samples, we found the following patterns:

Poorly reconstructed samples are unevenly distributed across different categories. The machine learning general category (ml_general) has the highest proportion, with 17 samples, accounting for 19.1% of all samples

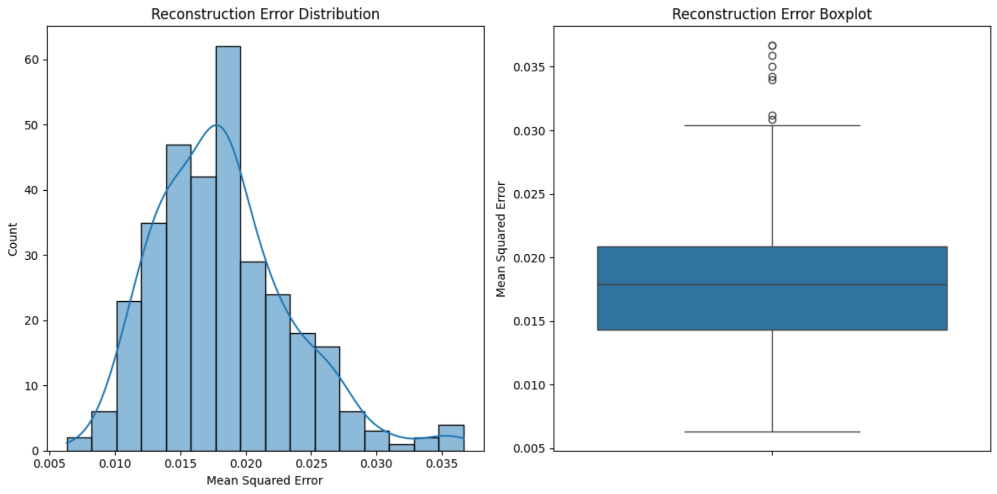

Figure 7: Reconstruction Error Distribution

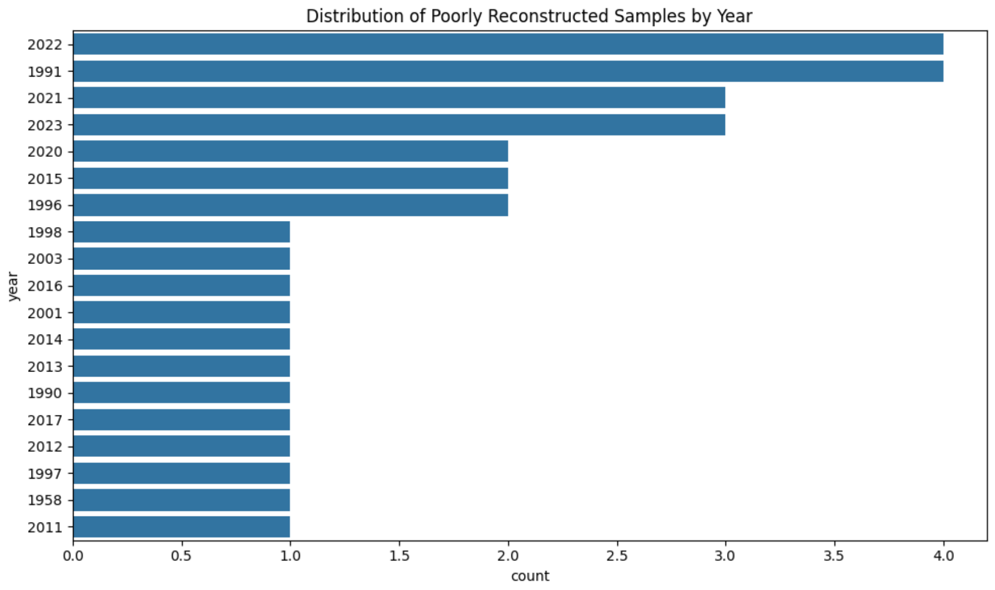

Figure 8: Distribution of Categories in Poorly Reconstructed Samples

in this category. This may be because this category covers a more diverse range of topics, resulting in more complex embedding representations.

In terms of publication year, poorly reconstructed samples are mainly concentrated in the most recent papers (2021-2023) and some classic papers (1991), which may reflect the uniqueness of these papers in research topics or expression styles.

Also, we found that poorly reconstructed samples are often interdisciplinary papers or pioneering works that propose entirely new methods, such as ”learning robust perceptive locomotion for quadrupedal robots in the wild” and ”an introduction to johnson lindenstrauss transforms.”

Table 2: Distribution of poorly reconstructed samples (error ¿ 0.025285)

Category	Poorly Reconstructed Samples	Percentage (%)
ml_general	17	19.1
dl_rnn	4	11.1
dl_rl	3	16.7
cv_pattern	3	5.7
dl_nlp	3	5.4
audio	1	4.0
cv_generative	1	2.3
Total	**32**	

To visually understand the reconstruction process, we selected several samples with the worst reconstruction quality and compared their original embedding vectors with their reconstructed vectors:

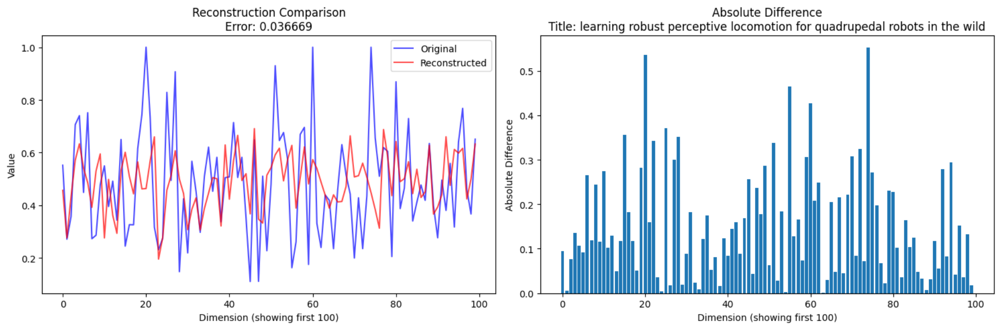

Figure 9: Sample Reconstruction Comparison: Learning Robust Perceptive Locomotion for Quadrupedal Robots Paper

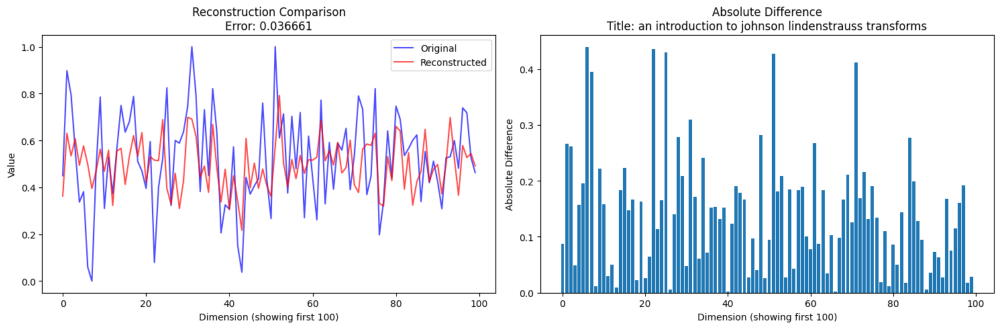

Figure 10: Sample Reconstruction Comparison: Johnson-Lindenstrauss Transforms Introduction Paper

From these comparison figures, we can observe that the autoencoder performs poorly when reconstructing high-frequency features, which may represent unique concepts or innovations in the papers. The reconstruction errors are mainly concentrated in certain specific dimensions of the vectors rather than being uniformly distributed across all dimensions.

Relationship Between Reconstruction Performance and Learned Representations

This analysis reveals the trade-offs in the dimensionality reduction process of autoencoders: they can effectively compress common features of most samples but may lose special information from unique samples.

This also explains why methods like UMAP, specifically designed for dimensionality reduction, may exceed autoencoders in clustering tasks, as they focus more on preserving local and global structures of the data.

Conclusion

The combination of autoencoder and GMM is a good tool for clustering and dimensionality reduction. The autoencoder is able to learn a compact representation of the data and the GMM is able to capture the structure of the data. However, the performance of the autoencoder is not as good as the UMAP embeddings, possibly due to inadquate design of the neural network architecture. In the future, we can try to use a more complex architecture or a different loss function to improve the performance of the autoencoder. We can also try to use regularization techniques to prevent overfitting.

The GMM is able to capture the structure of the data and produce better clustering on certain part of the data than what the tSNE plot shows. And to our surprise, the GMM is able to capture the structure of the data and produce better clustering on certain part of the data than what the tSNE plot shows and is it on par or even better than most of the clustering methods we used in the project. As a result, We should include GMM in our project as well.

References

[1] Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, Ömer Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Mariya Hendriksen, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Šuppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri Krishnakumar, Anna Maksimova, Silvan Wehrli, Maria Tikhonova, Henil Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Hongjin Su, Jimmy Lin, Howard Yen, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. Mmteb: Massive multilingual text embedding benchmark. *arXiv preprint arXiv:2502.13595*, 2025.

[2] Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. Jasper and stella: distillation of sota embedding models, 2025.

[3] Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, et al. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1393–1412, 2024.