Smart Tutor Knowledge Hub: An Exploration of Clustering on Academic Abstracts

Jaee Oh, Lu Chen, Ehsan Haghian, Peter Martin,
Sally Johnstone

April 26, 2025

1 Abstract

In the project described in this paper, we use unsupervised machine learning models to create a knowledge base of academic papers and other online resources for the purpose of powering an AI tutor. We begin by scraping data from Google Scholar using a set of predefined keywords to establish a dataset of data science related abstracts. We compared K-Means Clustering with Gaussian Mixture Models for clustering textual data embeddings on the abstracts produced by an autoencoder. Our findings suggest that both K-Means clustering and Gaussian Mixture Models produce clustering outputs of equal quality. The less computationally expensive K-Means clustering is selected for future iterations. We divide groups of papers into semantically similar partitions for the purpose of establishing the most relevant subdomains within the data science field and use these keywords to identify other resources to augment the knowledge base. These clusters are used by an application that we produce to read user input and respond to user inquiries with links to semantically similar online content.

2 Introduction

The amount of academic literature produced increases each year. In 2022 alone, more than 5.14 million articles were published[2]. For a researcher in any field, navigating the amount of relevant work and determining emerging trends is incredibly difficult. A student may spend their entire career playing catch-up with the rapid pace of development, especially in technological fields. AI can help ease the burden of staying current for learners of all levels.

Unsupervised learning techniques provide a multitude of solutions for these problems. With clustering, research trends are revealed by grouping semantically similar content. In addition, clusters can be used to establish deeper trends that exist within the clusters through an iterative process. That is, a single cluster can be divided further into more specific subdomains. This allows

for the building of flexible knowledge paths, with clear order for understanding topics in the data science field.

The primary goal of this project is to evaluate the feasibility of using unsupervised machine learning methods to build a system that guides users, particularly students and independent learners, through the academic field of data science by identifying and recommending semantically relevant content.

3 Literature Review

Recent advancements in Natural Language Processing (NLP) have focused on creating powerful, context-aware representations of text using deep learning. Two key technologies behind this progress are contextual embeddings, particularly those generated by BERT, and dimensionality reduction using autoencoders, both of which are central to our project.

Our embedding process is based on BERT (Bidirectional Encoder Representations from Transformers), introduced by Devlin et al[3]. BERT captures the meaning of words in context using bidirectional attention, setting new performance standards across many NLP tasks. Due to the high computational cost of using the full BERT model, we use a lighter version called all-MiniLM-L6-v2[6]. This distilled transformer strikes a balance between speed and accuracy, producing compact 384-dimensional sentence embeddings using the Sentence Transformers library. These embeddings are highly effective for downstream tasks such as clustering, sentiment analysis, and semantic search.

Before embedding, we preprocess the text using lemmatization with the Stanza toolkit[5]. Stanza provides advanced NLP features such as tokenization, part-of-speech tagging, and morphological analysis. Unlike simple stemmers, Stanza's lemmatization uses syntactic structure to map words like "running," "ran," and "runs" to a common root, "run." This ensures cleaner and more consistent input for embedding and improves the semantic quality of the output.

To maintain high data quality, we filter our dataset to include only the top 30% of papers based on citation counts, taken from the Citations column in our Google Scholar-derived dataset. This threshold, which is set at the 70th percentile, helps us exclude low-impact entries and focus our analysis on more influential works. This method aligns with previous studies that use citation-based filtering to prioritize academic importance[4].

We reduce the dimensionality of the 384-dimensional sentence embeddings using a symmetric autoencoder implemented in PyTorch. This model encodes each vector into a 64-dimensional latent space and reconstructs it by minimizing the mean squared error between the original and reconstructed vectors. Unlike linear methods such as Principal Component Analysis (PCA), autoencoders are capable of capturing non-linear relationships, allowing them to better preserve complex semantic structures within the data while effectively reducing noise. This advantage is particularly important for natural language representations, where semantic similarity is rarely linearly separable. Autoencoders provide a flexible and powerful framework for unsupervised representation learning, with

proven utility across tasks involving dimensionality reduction, feature extraction, and data reconstruction[1].

Finally, the reduced embeddings are clustered using K-Means or Gaussian Mixture Models (GMM). We selected the optimal number of clusters using the Silhouette Score (for K-Means) and Bayesian Information Criterion (BIC) (for GMM), in line with best practices in unsupervised learning. This clustering groups related papers into meaningful topics, supporting applications like academic recommendations.

4 Methodology

4.1 Tools and Frameworks

Reproducibility is a foundation of machine learning research. Since many algorithms depend on probabilistic operations and random number generators, we ensured consistent results by explicitly setting fixed random seeds across all libraries and internal parameters.

Scikit-learn is an easy-to-use machine learning package. Many basic functions for machine learning are available. The functions that are mainly used for this work are K-means, Gaussian Mixture Model, PCA, TF-IDF and various metric scores.

Although Scikit-learn is a very useful machine learning package, it mostly contains basic levels of machine learning techniques. In order to adapt to the Autoencoder function, we had to use additional packages, such as PyTorch. PyTorch was chosen over TensorFlow due to its compatibility with other packages. ReLu was used as an activation function, and Adam optimizer was used for the training of the autoencoder.

For text preprocessing, we used spaCy, a widely adopted NLP library. Its lemmatization capability was essential in reducing inflected words to their root forms (such as "aspects" $\rightarrow$ "aspect," "bounded" $\rightarrow$ "bound"), improving consistency in the text and minimizing redundant semantic representations. This step helps eliminate noise in the TF-IDF keyword extraction process and enhances the clarity of the clustering outcomes.

To generate meaningful sentence-level embeddings, we utilized the Sentence-Transformers library (SBERT), a modern embedding framework that builds on BERT architecture. Unlike traditional word-level models, SBERT generates semantically rich sentence vectors by considering context. We specifically used the all-MiniLM-L6-v2 model, which is a refined transformer with only six layers which provide a balance between performance and speed. This model could generate high-quality 384-dimensional embeddings for over 30,000 academic abstracts in a computationally efficient manner.

For dimensionality reduction, we implemented a symmetric autoencoder using PyTorch. The encoder compresses high-dimensional SBERT embeddings into a 64-dimensional latent space, and the decoder attempts to reconstruct the original vector by minimizing mean squared error. We used ReLU as the

activation function and the Adam optimizer to train the model, ensuring fast convergence and stable training dynamics. PyTorch was selected for its modularity, clear syntax, and perfect integration with GPU resources and other libraries and frameworks.

For clustering and evaluation, we relied on Scikit-learn, a widely used Python library that provides strong implementations of unsupervised learning algorithms. Specifically, we used:

1. K-Means for hard clustering,
2. Gaussian Mixture Models (GMM) for soft probabilistic clustering,
3. TF-IDF for keyword extraction within clusters,
4. Silhouette Score, WCSS, and Bayesian Information Criterion (BIC) for model evaluation and cluster validation.

4.2 Data and Preprocessing

The dataset used in this project consists of 30000 research paper entries, each with the following key attributes:

- Title: The full name of the published paper.
- Abstract: A concise summary outlining the paper's motivation, methodology, results, and conclusions.
- Authors: The names of the primary contributors.
- Year: The publication year of the paper.
- URL: A direct link to the paper
- Source: The origin of the paper (e.g., "arXiv" or "Google Scholar").

Before generating embeddings, the abstract text is preprocessed as follows:

- Missing abstract imputation: For entries missing abstracts, the title was used as a substitute.
- Lemmatization: Applied using spaCy to convert inflected words to their base form (Like "running," "ran" $\rightarrow$ "run"), reducing redundancy and improving embedding quality.
- Lowercasing: All text was converted to lowercase to prevent duplicate representations of the same word (Like "Neural" vs. "neural").
- Numerical data removal: Numbers were removed as they generally add little semantic value in scientific abstracts.
- Punctuation removal: Eliminated to focus solely on linguistic content.
- Whitespace normalization: Ensured consistent spacing for clean formatting.

4.3 Text Embedding Using BERT

To generate high-quality semantic representations of the abstracts, we utilized the all-MiniLM-L6-v2 model from the Sentence-Transformers library. This model, a refined variant of BERT, encodes entire sentences into 384-dimensional vectors that efficiently capture contextual meaning. Its smaller size and faster inference time made it ideal for processing our dataset of over 30000 academic papers without compromising embedding quality.

The use of MiniLM significantly improved downstream performance. Compared to traditional TF-IDF or word-level embeddings, BERT-based vectors enabled more accurate clustering by targeting semantically similar abstracts closer together in the embedding space. This was evident in the tight cluster formations and meaningful keyword groupings observed after applying

dimensionality reduction and K-Means. Overall, the use of BERT embeddings laid a strong foundation for semantic clustering and recommendation within the academic domain.

4.4 Dimensionality Reduction

To reduce the dimensionality of the high-dimensional BERT embeddings, we employed an autoencoder; a type of neural network designed to learn compact, information-rich representations of input data. An autoencoder works by encoding input vectors into a lower-dimensional latent space (the bottleneck layer) and then reconstructing the original input from this compressed representation. This approach forces the model to retain only the most significant features, effectively filtering out noise and redundancy.

In our implementation, the autoencoder compresses the 384-dimensional SBERT embeddings into a 64-dimensional latent representation, which is used for downstream clustering tasks. The reconstruction output is of the same dimensionality as the input, allowing us to calculate reconstruction errors and evaluate how well the latent space captures the original information.

The model was trained for 50 epochs, with each epoch comprising a full forward and backward pass through the dataset. Training was optimized using the Adam optimizer with a learning rate of 0.001. Adam was chosen over traditional optimizers like Stochastic Gradient Descent due to its adaptive learning rate and ability to handle sparse gradients which we can consider as an advantage when working with semantic embeddings from textual data. Its efficiency and strength made it an ideal fit for this task without requiring extensive hyperparameter tuning.

The Mean Squared Error (MSE) loss function was used to evaluate the difference between the original and reconstructed embeddings. MSE is well-suited for this task as it penalizes large deviations and provides a clear quantitative measure of reconstruction quality. Lower MSE values indicate that the latent representation has preserved most of the semantic information from the original embeddings.

The output of the autoencoder (64-dimensional latent vectors) served as

an input to our clustering algorithms. These reduced embeddings significantly improved both computational efficiency and clustering performance, as they retained critical semantic structures while eliminating redundant dimensions. As shown in Figure 1 (Appendix A), the architecture consists of fully connected layers with ReLU activation functions and symmetric encoder-decoder structures.

To further evaluate the effectiveness of dimensionality reduction, we visualized the distribution of reconstruction errors across samples and inspected the quality of the clusters formed using the reduced space. The results showed that the autoencoder was able to reconstruct embeddings with low error while improving separation in cluster boundaries compared to raw embeddings.

5 Algorithm

To identify semantic groupings within the dataset, we applied clustering techniques to the 64-dimensional latent representations produced by the autoencoder. Two unsupervised learning algorithms were investigated: K-Means and Gaussian Mixture Models (GMM).

K-Means was selected as a baseline method due to its simplicity and computational efficiency. It partitions data into non-overlapping clusters by minimizing intra-cluster variance, assigning each sample to the nearest cluster centroid. In contrast, GMM adopts a probabilistic soft clustering approach, assigning each data point a likelihood of belonging to multiple clusters. This is particularly valuable in the academic context, where a single paper may span multiple topics or fields, requiring a more precise clustering approach.

To extract meaningful topics from each cluster, we used TF-IDF (Term Frequency-Inverse Document Frequency) on the original text data. For every cluster, we computed the average TF-IDF score for each word and selected the top 10 terms to characterize the cluster's thematic content. This process helped us label and interpret research themes across groups.

5.1 Determining the Optimal Number of Clusters

Selecting the appropriate number of clusters is a challenging task due to conflicting signals from different evaluation metrics. To guide this decision, we employed multiple methods:

- Silhouette Score reached its highest value at k = 28, indicating strong inter-cluster separation. However, the gradual upward trend of the score with increasing k raised concerns about over-segmentation, where clusters may become too fine-grained for meaningful interpretation.
- The Elbow Method, based on the within-cluster sum of squares (WCSS), revealed an inflection point at k = 13. This suggests that adding more

clusters beyond this point yields only marginal gains in compactness, making k = 13 a reasonable trade-off between model complexity and variance reduction.

- The Bayesian Information Criterion (BIC), used in conjunction with Gaussian Mixture Models (GMM), identified k = 11 as the optimal number of clusters. The BIC reached its lowest value at this point, indicating the best balance between model fit and complexity.

We ultimately selected 28 clusters as the optimal configuration based on the peak observed in the Silhouette Score, which measures how well-separated the clusters are. Although other metrics such as the Elbow Method and BIC suggested smaller values like 11 and 13, we prioritized interpretability and granularity in topic representation. A higher number of clusters allows for more fine-grained semantic differences between research areas, which is particularly valuable in the context of academic exploration where papers often focus on highly specific subtopics. By choosing 28 clusters, we aimed to capture this diversity and provide users with more targeted and meaningful groupings, which can later be hierarchically organized or merged into broader categories as needed

5.2 TF-IDF Keyword Extraction

To interpret and label each cluster with meaningful topics, we applied TF-IDF to the abstracts grouped by cluster. This technique identifies words that are both frequent within a cluster and distinctive across the entire dataset, making it well-suited for uncovering key themes.

For each cluster, we calculated the average TF-IDF score of every term across all documents in that group. The top-ranking words (top 10 per cluster) were extracted to represent the central concepts associated with that cluster. These keywords provide a compact summary of the major topics and enhance interpretability for downstream tasks such as semantic search and topic navigation.

For example, one cluster was characterized by terms like “transformer,” “attention,” “pretraining,” and “language model,” indicating a focus on recent advancements in NLP architectures. Another cluster included keywords such as “graph,” “node,” “embedding,” and “structure,” reflecting research in graph neural networks. These keyword sets not only validated the quality of clustering but also served as labels for the semantic partitions of the dataset. Generative AI tools may be applied to reconfigure these keywords into a single phrase for better readability for human users.

5.3 User Interface and Query Matching

To make the system interactive and user-friendly, we developed a search interface that allows users to retrieve semantically relevant academic papers based on input from natural language. Users can enter either a single keyword or a full question in the search field as shown in Figure 7 (Appendix A). The query is first

embedded into a 384-dimensional vector using the pre-trained BERT model and then reduced to 64 dimensions using the same autoencoder used during training.

Instead of comparing the query vector against the entire dataset of over 30,000 papers, we improve performance and relevance by applying a two-layered clustering and filtering approach. In the first layer, we use K-Means clustering in the paper abstracts to identify which semantic cluster the query most closely belongs to. This significantly narrows the search space by focusing only on the most relevant group of documents.

Within this filtered cluster, we compute the cosine similarity between the query vector and the reduced abstract embeddings. Documents with a similar score above 0.50 are retained for further consideration.

In the second layer, we refine the results further by performing an additional clustering step, this time using only the titles of the selected papers. The query is then re-mapped to this smaller, more focused title-based cluster, enhancing precision and topical alignment. This layered design ensures both semantic depth from abstracts and sharp topical specificity from titles.

Finally, the top-matching papers are ranked and displayed to the user, along with their titles, URLs, and similarity scores as shown in Figure 7 (Appendix A). The system has the capability to return highly relevant papers, even when the exact wording differs between the query and the documents. This is a key strength of BERT-based embeddings: their ability to capture semantic meaning and context, allowing the system to match documents to queries based on meaning rather than just keyword overlap.

This intelligent, two-phase architecture not only improves retrieval accuracy and speed, but also creates a more natural and powerful search experience for students and researchers.

5.4 Iteration

The previous sections in Methodology describe a single run from scraping, clustering and user interaction. This produced a database separated or clustered by subtopics of the initial input keywords. However, some students or users may want to study in more detail the subtopics. Our solution to this is to apply iteration to this whole process to scrape the keywords of the resultant clusters and cluster them again, so we can obtain subtopics within the previous subtopics. This iteration process can be applied multiple times to create hierarchical topics in a tree structure.

There is also a benefit of extracting resources from different search engines, since each iteration involves scraping at the beginning. For example, we can use the output keywords of the first cluster of the initial run with Google Scholar scraping to scrape on YouTube and obtain more user-friendly tutorials.

6 Result

6.1 Reconstruction Error Analysis

To evaluate how well the autoencoder retained the semantic structure of the original BERT embeddings, we analyzed the distribution of reconstruction errors using Mean Squared Error (MSE) as the loss metric. For each sample, we calculated the squared distance between the original and reconstructed embedding vectors.

A histogram of reconstruction errors (see Figure 1 in Appendix A) reveals that the majority of samples exhibit low reconstruction errors, indicating successful compression and preservation of meaningful information. The distribution is right-skewed, with a small number of samples showing higher error, which may correspond to outliers or less semantically consistent abstracts. This analysis confirms that the autoencoder effectively filtered out noise while preserving the key semantic relationships needed for clustering. We also identified the top 5% of samples with the highest reconstruction error to investigate potential model limitations. 95% of the data exhibited low reconstruction error, indicating that the autoencoder effectively captured the underlying semantic structure of the embeddings while minimizing loss of information.

6.2 Training and Validation Loss

To monitor the learning progress of the autoencoder, we tracked training and validation loss across epochs. This allows us to assess the generalization capability of the model and detect issues such as overfitting or underfitting.

As shown in Figure 2 (Appendix A), both the training and validation losses decreased steadily during the first few epochs and then stabilized, indicating that the model effectively minimized the reconstruction error without overfitting to the training data. The close alignment between the two curves suggests strong generalization and a well-balanced training process.

This consistency confirms that the autoencoder learned meaningful compressed representations of the BERT embeddings and was able to reconstruct them accurately without memorizing the input data.

6.3 Cluster

To visually assess the quality and separation of the clusters, we used Principal Component Analysis (PCA) to reduce the 64-dimensional latent vectors (from the autoencoder) to 2D space. PCA, which is a linear dimensional reduction technique, is not used in our actual clustering pipeline. It only serves as an effective method for visualization of high-dimensional data in a human-interpretable format.

Figure 4 (Appendix A) compares the shapes of K-Means clusters using 13 clusters (left) and 28 clusters (right), visualized using PCA for 2D projection. The 13-cluster model forms broader, more compact groups, while the 28-cluster

model reveals finer topic segmentation and greater thematic granularity. The more detailed separation seen in the 28-cluster configuration supports our final choice, as it enables more targeted recommendations and subdomain discovery for users.

In addition to shape comparison, we evaluated clustering quality using Silhouette Score, Inertia (WCSS), and BIC (Bayesian Information Criterion) as shown in Figure 3 and 5 (Appendix A). As detailed earlier, although the elbow method and BIC favored 11 and 13 clusters, the peak silhouette score occurred at k = 28, guiding our decision to prioritize semantic richness and precision in the final application.

6.4 Semantic Search and Recommendation

Figure 7 (Appendix A) demonstrates our semantic search engine in action. When the user enters the query "What is PCA", the system embeds the question using BERT, filters the dataset via K-Means clustering, and calculates cosine similarity between the query and relevant documents.

The top 10 results are ranked by similarity scores, with values ranging from 0.64 to 0.53, showing strong semantic alignment. Notably, the retrieved titles are highly relevant, covering tutorials, reviews, and advanced topics on Principal Component Analysis. This confirms that our system successfully captures the underlying meaning of the query, even when the wording differs from the text in the documents.

7 Discussion

The results of our project support a structured and effective approach for clustering scholarly articles based on their semantic content. We began by transforming abstracts and titles into dense vector representations using pre-trained BERT embeddings, specifically the all-MiniLM-L6-v2 model. This lightweight transformer captures deep contextual meaning while remaining efficient for large-scale applications. The resulting 384-dimensional embeddings effectively encode both the literal content and semantic relationships within the text, forming a strong foundation for downstream clustering.

Preprocessing played a key role in enhancing clustering quality. Lemmatization, applied before embedding, unified word variations (Such as "runs," "ran," "running" $\rightarrow$ "run"), reducing vocabulary fragmentation and aligning related concepts in embedding space. This contributed to more consistent cluster boundaries and clearer TF-IDF keyword outputs.

To address the high-dimensionality and reduce noise, we employed a symmetric autoencoder to compress the embedding into a 64-dimensional latent space. By minimizing mean squared reconstruction error, the autoencoder learned to retain essential semantic features while eliminating redundant variation. This improved the cluster separability and interpretability compared to raw embeddings.

We applied K-Means clustering to the reduced embeddings, chosen for its simplicity, scalability, and strong performance on dense numerical vectors. Clustering configurations were evaluated using silhouette scores, inertia (WCSS), and visual inspection. While early experiments favored 13 clusters based on the elbow method, silhouette score peaked at 28 clusters, revealing finer semantic granularity. This configuration was adopted to support more precise thematic segmentation across subfields.

Clusters were evaluated qualitatively through keyword extraction, which revealed evident and meaningful topics such as transformer architecture, reinforcement learning, and graph neural networks. This thematic coherence confirms that our clustering approach effectively captures the research landscape across AI and data science.

Moreover, we integrated this pipeline into a two-layer semantic search interface, enabling users to query by topic or question and retrieve relevant papers in real-time. The system first narrows search space using cluster proximity, then ranks papers using cosine similarity, ensuring fast and contextually accurate recommendations. For example, the query "What is PCA" returned topically accurate results with similarity scores above 0.64, demonstrating practical utility of the clustering results.

Despite its strengths, the system has limitations. Choosing the number of clusters remains subjective due to conflicting metrics. Also, computational constraints limit our ability to experiment with deeper autoencoder architectures or train on larger datasets.

Overall, the results align strongly with our objective of building an interactive, AI-powered academic exploration tool. The combination of contextual embeddings, dimensionality reduction, and probabilistic clustering forms a robust backbone for scalable semantic understanding. With future refinements in labeling, user interface, and domain adaptation, the system can evolve into a powerful learning assistant for AI and data science education.

8 Conclusion

This project demonstrated the feasibility and value of using semantic clustering to organize and explore academic literature in machine learning and AI. By combining contextual embeddings from a distilled BERT model, dimensionality reduction through an autoencoder, and clustering using K-Means and GMM, we were able to identify meaningful thematic groupings across thousands of research papers. The two-layer pipeline, using abstract-based and title-based clustering significantly improved both performance and precision. Our semantic search interface allowed users to enter natural language queries and retrieve contextually relevant papers, showing the real-world potential of the system as an AI-powered academic assistant.

For future work, we can extend the platform with a hierarchical clustering structure that enables users to navigate from broad research domains to highly specific subtopics. Additionally, integrating a web-based user interface

would make the tool accessible to a wider audience. Further improvements could involve fine-tuning the embedding model for domain-specific vocabulary and experimenting with dynamic or user-personalized clustering to adapt to individual learning goals.

9 Individual Reflections

- **Peter Martin**: I took away from this project how to construct AI powered systems. One of my frustrations with my undergraduate computer science experience was an emphasis on learning and utilizing material only in isolation. There wasn't enough opportunity to practice real software engineering by synthesizing all of what was learned to create something new and interesting. This project proved a valuable experience from both a data science perspective and a software engineering perspective. I hope future iterations of this course take the same approach with the final project.

- **Lu Chen**: I learned how to scrape papers from Google Scholar, which I found to be a valuable skill for accessing online materials, even when faced with obstacles from CAPTCHA. I also gained an understanding of lemmatization in natural language processing, where converting words to their basic forms helps reveal underlying similarities and improves clustering results. Additionally, I found that even a basic clustering algorithm like K-Means can perform well for our tasks, demonstrating that simple methods can be effective in practice.

- **Jaee Oh**: The main takeaway for me of the project and the class was that AI and unsupervised machine learning are extensions of data analysis that I've experienced from experimental study in physics. Some basics of Natural Language Processing that I've learned from this class, such as SBERT and TF-IDF, broadened my understanding on common generative AIs such as ChatGPT and ClaudeAI. Surprising part was that even the simplest clustering technique like K-Means can be very useful, and there is no need to only use the latest clustering techniques that may be very complicated and computationally more expensive.

- **Sally Johnstone**: It was an incredible experience to have the privilege of working with the smartest people in the room. I must admit that the first time I read the Project Group One project description, I was a bit overwhelmed by the unfamiliar language and the suggested analytical approach. Now that I have been part of developing a complete unsupervised machine learning pipeline, from scraping to preprocessing text data, to reconstructing clusters into human-readable text, I feel confident to do it in other contexts. I will be using what I learned with my colleagues about web scraping, lemmatization, SBERT, autoencoders, and TF-IDF in a project I am doing for a friend this summer to analyze changes in public sentiment around their institution.

- **Ehsan Haghian**: Through this project, I gained a valuable understanding of how modern NLP techniques like BERT and autoencoders can be integrated with clustering methods to organize large volumes of text data. More importantly, I learned how to apply these concepts in a practical,

real-world setting while collaborating with a great team. Working together to build and refine a functional system taught me the value of communication, adaptability, and shared problem-solving skills that are just as important as the technical ones in any successful project.

References

[1] Kamal Berahmand et al. “Autoencoders and their applications in machine learning: a survey”. In: *Artificial Intelligence Review* 57.2 (2024), p. 28.

[2] Curcic D. *Number of Academic Papers Published Per Year*. `https://wordsrated.com/number-of-academic-papers-published-per-year/`. 2023. (Visited on 06/01/2023).

[3] Jacob Devlin et al. “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, volume 1 (long and short papers)*. 2019, pp. 4171–4186.

[4] Chanwoo Jeong et al. “A context-aware citation recommendation model with BERT and graph convolutional networks”. In: *Scientometrics* 124 (2020), pp. 1907–1922.

[5] Peng Qi et al. “Stanza: A Python natural language processing toolkit for many human languages”. In: *arXiv preprint arXiv:2003.07082* (2020).

[6] Chen Yin and Zixuan Zhang. “A Study of Sentence Similarity Based on the All-minilm-l6-v2 Model With “Same Semantics, Different Structure” After Fine Tuning”. In: *2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024)*. Atlantis Press. 2024, pp. 677–684.

10 Appendix A: Figures

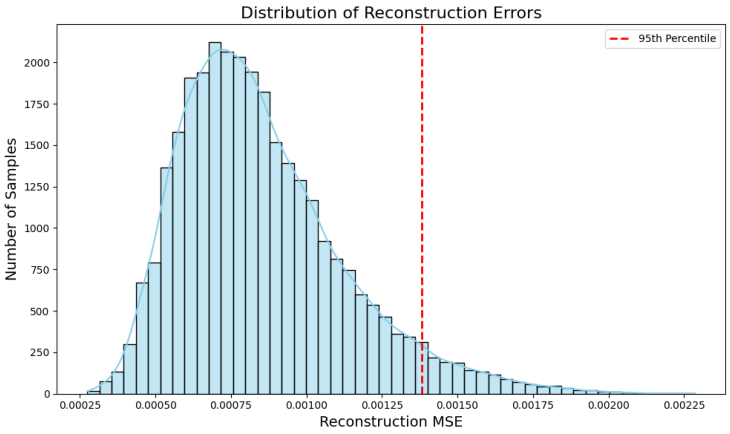

Figure 1: Reconstruction Errors

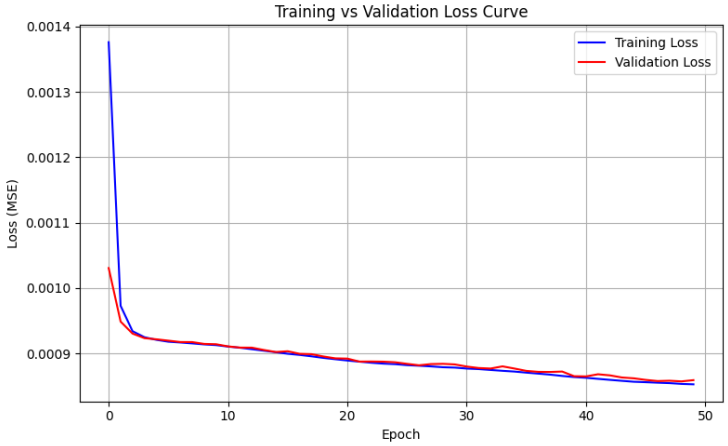

Figure 2: Training vs Validation Loss

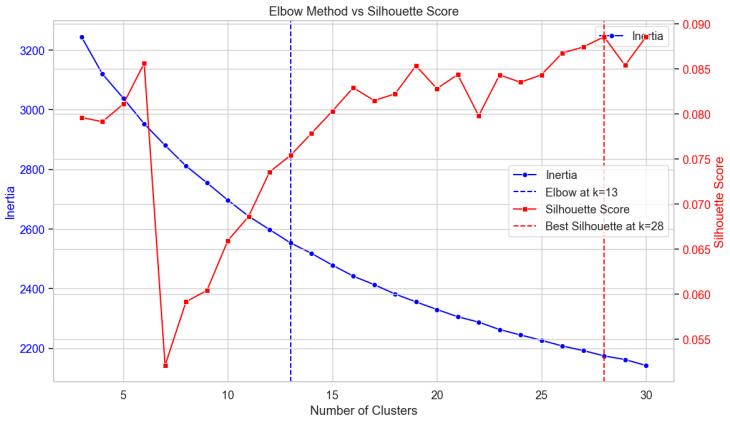

Figure 3: Elbow Method vs Silhouette Score

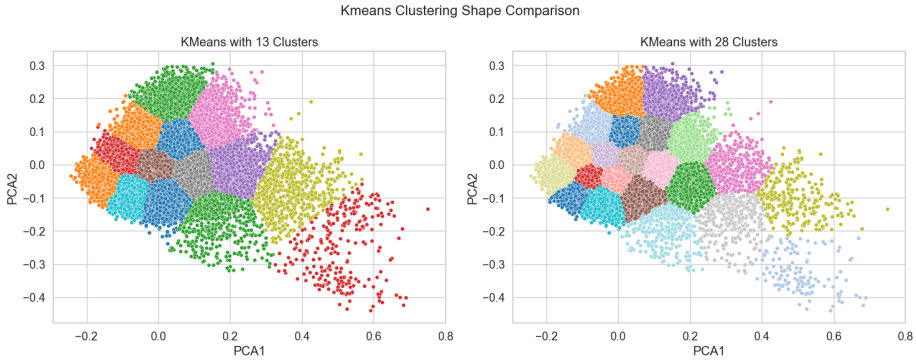

Figure 4: K-Means

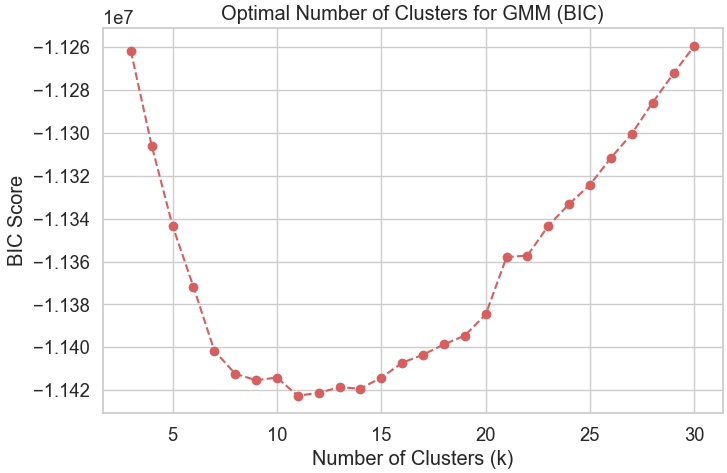

Figure 5: BIC

Figure 6: GMM

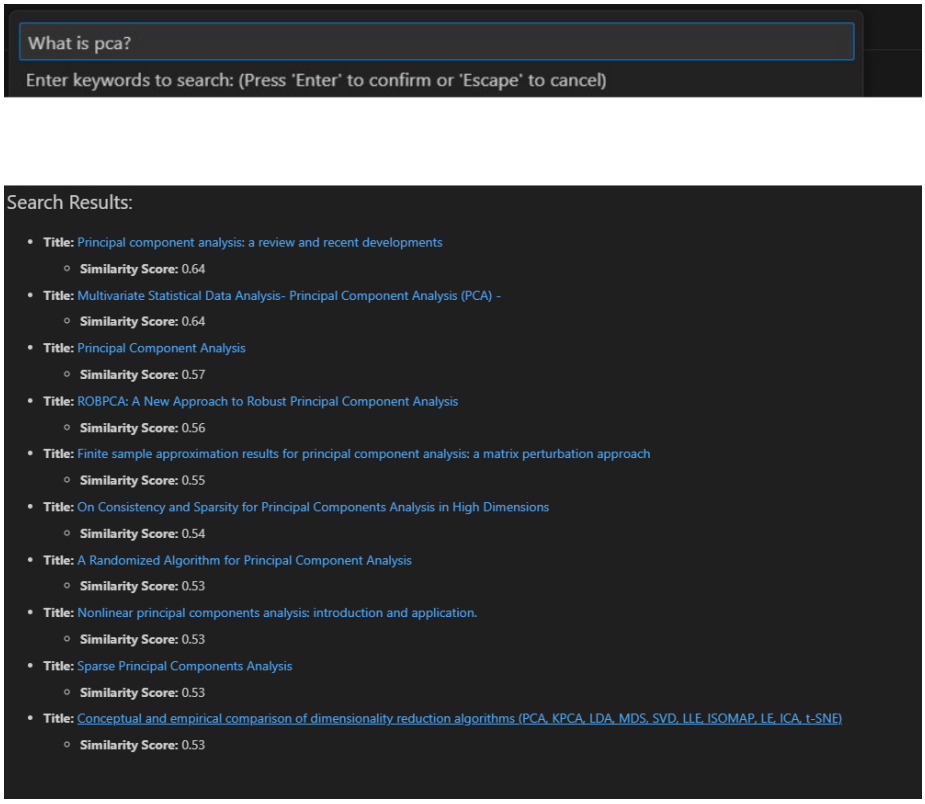

Figure 7: User Interface

11 Appendix B: Cluster Keywords

- Cluster 0 — Top Keywords: reinforcement, learning, policy, agent, rl, learn, task, reward, environment, algorithm
- Cluster 1 — Top Keywords: learning, label, learn, supervised, shot, image, representation, semi, class, datum
- Cluster 2 — Top Keywords: galaxy, star, cluster, mass, model, formation, spectral, density, matter, optical
- Cluster 3 — Top Keywords: vector, support, machine, svm, svms, classification, algorithm, use, prediction, base
- Cluster 4 — Top Keywords: topic, sentiment, text, model, document, word, user, classification, use, twitter
- Cluster 5 — Top Keywords: online, student, learning, education, learn, study, course, teaching, teacher, research
- Cluster 6 — Top Keywords: language, model, task, bert, transformer, llm, large, text, sentence, train
- Cluster 7 — Top Keywords: material, diffusion, energy, model, molecular, chemical, machine, density, molecule, property
- Cluster 8 — Top Keywords: machine, tree, algorithm, learning, classification, decision, learn, classifier, datum, forest
- Cluster 9 — Top Keywords: federate, learning, privacy, fl, device, datum, communication, model, distribute, client
- Cluster 10 — Top Keywords: regression, model, logistic, analysis, effect, variable, use, linear, datum, method
- Cluster 11 — Top Keywords: image, deep, cancer, medical, learning, disease, use, clinical, patient, imaging
- Cluster 12 — Top Keywords: network, neural, deep, learning, recurrent, model, learn, architecture, task, layer
- Cluster 13 — Top Keywords: detection, attack, anomaly, learning, machine, model, network, deep, datum, intrusion
- Cluster 14 — Top Keywords: video, recognition, temporal, action, feature, representation, model, object, detection, network
- Cluster 15 — Top Keywords: gan, generative, adversarial, image, model, network, generate, generator, discriminator, propose
- Cluster 16 — Top Keywords: graph, network, node, representation, learning, gnn, structure, learn, datum, model

- Cluster 17 — Top Keywords: regression, lasso, linear, method, algorithm, problem, sparse, model, selection, kernel
- Cluster 18 — Top Keywords: cluster, clustering, algorithm, datum, mean, method, spectral, base, analysis, hierarchical
- Cluster 19 — Top Keywords: protein, drug, molecular, diffusion, model, discovery, molecule, structure, cell, chemical
- Cluster 20 — Top Keywords: hyperspectral, classification, datum, model, image, use, spectral, remote, spatial, climate
- Cluster 21 — Top Keywords: landslide, use, model, power, susceptibility, fault, transformer, network, control, base
- Cluster 22 — Top Keywords: ai, model, research, business, technology, firm, intelligence, innovation, industry, explain
- Cluster 23 — Top Keywords: forecasting, series, stock, model, time, price, financial, forecast, market, prediction
- Cluster 24 — Top Keywords: image, transformer, network, model, object, vision, convolutional, cnn, feature, attention
- Cluster 25 — Top Keywords: quantum, classical, machine, learning, algorithm, network, neural, computing, quantization, circuit
- Cluster 26 — Top Keywords: ai, intelligence, artificial, human, machine, education, healthcare, research, ethical, technology
- Cluster 27 — Top Keywords: brain, memory, eeg, attention, network, neural, mechanism, cognitive, learning, signal

12 Appendix C: Source Codes

GitHub contains all the codes used for the project.

- In order to run the demo with ArXiv scraping, please follow the instructions on the README of the repo.
- If you are more used to Colab or Jupyter environment, you may check the files in colab_src. This file is meant to be ran at Colab, and the current environment may not be compatible. Please use Colab environment, or use Jupytern with the same package dependencies as Colab.
- The presentation in docs were done based on the analysis in FinalProject(Finalized).ipynb. However, this file did not include lemmatization in preprocessing. As a result, the correct analysis with