Information Technology Course
Module Software Engineering
by Damir Dobric / Andreas Pech

Clustering Documents and Visualization of Embedding Vector Space

Aditya Shidhaye
aditya.shidhaye@stud.fra-uas.de

Adithya R
adithya.ramesh@stud.fra-uas.de

Pradeep Patwa
pradeep.patwa@stud.fra-uas.de

Abstract*— In the digital age, the exponential growth of textual data necessitates efficient methods for organizing and analyzing large document collections. Traditional approaches to text classification and sorting are often time-consuming and labor-intensive. Machine Learning (ML) offers a powerful solution by automating these processes using predefined algorithms to analyze vast amounts of data and extract meaningful insights. This project addresses the challenge of unstructured text data by leveraging semantic embeddings generated through OpenAI's language models, such as text-embedding-3-large. These embeddings transform textual content into high-dimensional vector representations, enabling effective clustering using the K-Means algorithm. By grouping documents based on semantic similarity, such as invoices, support tickets, or news articles, this approach eliminates the need for manual categorization. To evaluate clustering performance, key metrics such as cosine similarity, intra-cluster similarity, inter-cluster similarity, and cluster purity are computed. The project provides visual representations of document clusters through scatter plots, where colors indicate cluster assignments or original categories. A comprehensive evaluation report further assesses the effectiveness of clustering methodologies. By automating text classification, this approach significantly reduces manual effort and enhances the efficiency of analyzing diverse document types, including survey reports, sales records, and customer feedback.

Keywords—OpenAI GPT, Semantic Embeddings, Document Clustering, Machine Learning, OpenAI NuGet Package, K-Means Algorithm, Cosine Similarity, Dimensionality Reduction, Visualization, Vector Space, Similarity Analysis, PCA, t-SNE, Scalar Values.

I. INTRODUCTION

In the digital age, vast amounts of textual data are generated daily across various domains, including business, research, and customer interactions. Managing and extracting meaningful insights from unstructured text data presents a significant challenge due to its sheer volume and complexity. Traditional manual methods for organizing and classifying documents are inefficient, time-consuming, and prone to errors.

Machine Learning (ML) has emerged as a powerful solution to automate text classification and clustering, reducing human effort while improving accuracy. By leveraging advanced natural language processing (NLP) techniques, it is possible to group documents based on their semantic meaning rather than relying on predefined categories.

This project focuses on implementing an automated document clustering system using OpenAI's language models to generate semantic embeddings. These embeddings capture the contextual meaning of documents in a high-dimensional space, allowing for more effective clustering using the K-Means algorithm. The system is designed to categorize various types of documents, such as invoices, support tickets, and reports, without manual intervention. Additionally, quantitative evaluation metrics, including cosine similarity, intra-cluster similarity, and cluster purity, are used to assess the effectiveness of the clustering process.

By automating document organization and visualization, this project provides a scalable and efficient solution for managing large collections of textual data, making it a valuable tool for businesses, researchers, and organizations dealing with unstructured information.

To efficiently process and analyse large volumes of textual data, this project leverages OpenAI's text-embedding-3-large model to generate semantic embeddings. Semantic embeddings are numerical vector representations of text that capture the contextual meaning of words, sentences, or entire documents. These embeddings enable advanced natural language processing (NLP) tasks such as clustering, similarity analysis, and search optimization.

The implementation utilizes the OpenAI NuGet package within a .NET-based environment to interact with OpenAI's API. The process begins with initializing the OpenAI API client and authenticating it using an API key. Each document in the dataset is then processed through the text-embedding-3-large model, which converts it into a high-dimensional vector. These vector representations allow the system to analyze semantic similarities between different documents, independent of keyword matching.

By leveraging OpenAI's GPT-powered embeddings, this approach significantly enhances document organization,

making it more scalable and efficient. It reduces manual effort in text classification and allows for automated document analysis in various domains, such as business intelligence, customer service, and knowledge management. The use of machine learning-based embeddings improves the accuracy of document clustering, making it a valuable tool for managing unstructured text data.

II. METHODS

This section is divided into three major subsections. The first section provides an overview of document clustering and the role of embeddings in text analysis. The second section focuses on the theoretical background of semantic embeddings, including their generation and importance in clustering. The third section discusses the clustering approach used, specifically K-Means, and evaluates its effectiveness through similarity metrics.

A. Literature Review

The goal of this literature review is to understand document clustering and embedding techniques from various research perspectives, which will help in designing a robust clustering model.

The study by Mikolov et al. [1] introduced word embeddings using neural networks, demonstrating how high-dimensional vector representations capture semantic relationships between words. These embeddings have since been widely used in NLP tasks, including text clustering and classification.

Reimers and Gurevych [2] proposed the use of sentence embeddings to improve semantic similarity detection. Their work demonstrated that transformer-based models such as BERT and OpenAI's GPT perform significantly better in capturing contextual meaning than traditional TF-IDF or word2vec methods.

A study by Radford et al. [3] showcased OpenAI's GPT model's ability to generate embeddings that accurately capture the semantic structure of entire documents. These embeddings have been effectively applied in tasks such as document retrieval, summarization, and clustering.

Zhang et al. [4] introduced an adaptive K-Means clustering approach that dynamically adjusts the number of clusters based on document density. Their approach was shown to improve clustering performance on large-scale document datasets.

These studies indicate that leveraging deep learning-based embeddings in clustering leads to more accurate document organization. The literature supports the idea that OpenAI's language models, combined with efficient clustering algorithms, provide a scalable and effective solution for automated text classification and clustering.

B. Theoretical Background of Semantic Embeddings

The Semantic embeddings are vectorized representations of text that capture contextual meaning in a high-dimensional space. Unlike traditional keyword-based approaches, embeddings enable comparisons based on semantic similarity rather than exact word matches.

Text embeddings convert textual data into high-dimensional numerical vectors, capturing the semantic meaning of words, phrases, or entire documents. These embeddings are essential for various NLP tasks such as document clustering, search ranking, recommendation systems, and semantic similarity analysis.

In this project, we utilize OpenAI's text-embedding-3-large model to generate high-quality vector representations of documents. This advanced embedding model enhances semantic understanding by providing higher-dimensional embeddings compared to its predecessor, text-embedding-ada-002, which had 1536 dimensions. With improved contextual comprehension, text-embedding-3-large is well-suited for tasks requiring deep text analysis and ensures more accurate clustering and retrieval-based applications. Its optimized performance allows for better organization and categorization of documents based on their semantic similarities, making it a powerful tool for large-scale text processing.

How It Works:

1. Tokenization: The input text is first broken into smaller subword units (tokens) using OpenAI's tokenizer.

2. Contextual Representation: The model processes the tokenized text using a transformer-based architecture that captures the relationships between words.

3. High-Dimensional Vector Generation: The output is a fixed-length dense vector representation of the input text.

4. Normalization: The generated embeddings are typically L2-normalized, ensuring that the similarity measures (like cosine similarity) are more effective.

These embeddings serve as input for clustering, ensuring that semantically similar documents are grouped together even if they do not share common keywords. The embeddings are generated using the OpenAI NuGet package, which provides a seamless API for retrieving high-dimensional vector representations of text data.

C. Clustering Approach and Evaluation

Once embeddings are generated, they are processed using the K-Means clustering algorithm, a widely used unsupervised learning method for partitioning data into distinct clusters.

Theoretical Background of K-Means Clustering:

The **K-Means clustering algorithm** is one of the most widely used unsupervised learning techniques. It was first introduced by Stuart Lloyd in 1957 and later refined by MacQueen in 1967. The algorithm is primarily used for **partitioning data into K distinct clusters** based on feature similarity. Unlike supervised classification methods, K-Means does not require labeled data and works by iteratively grouping data points into clusters based on their similarity.

In the K-Means clustering algorithm, several parameters play a crucial role in the design and performance of the model. These parameters are discussed below:

A. Distance Calculation in K-Means

The core of K-Means clustering is assigning data points to the closest cluster center (centroid) based on a distance metric. The most commonly used distance measure is Euclidean distance, though other metrics like Manhattan distance or cosine similarity can be used in specific scenarios.

Euclidean Distance: Euclidean distance as shown in Figure 1 is the straight-line distance between two points in an n-dimensional space. The Euclidean distance between two points (x_1, y_1) and (x_2, y_2) is calculated as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

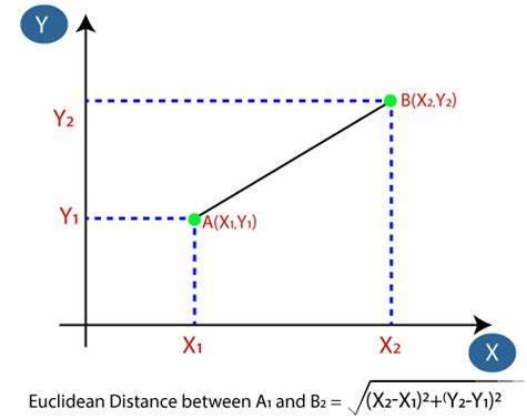

Figure 1: Euclidean Distance

B. Choosing the Optimal Number of Clusters (K Selection)

The number of clusters **K** is a critical hyperparameter in K-Means. An incorrect choice of K can lead to poor clustering performance. There are several methods to determine the optimal K value:

a) The Elbow Method

The Elbow Method is one of the most widely used techniques to determine the ideal number of clusters. It involves plotting the within-cluster sum of squares (WCSS) against different values of K. The point where the WCSS curve starts to flatten (forming an "elbow") indicates the optimal K value.

b) The Silhouette Score

The Silhouette Score evaluates the quality of clustering by measuring how similar a data point is to its assigned cluster compared to other clusters. A higher silhouette score (close to 1) indicates better clustering. In our project we have used this method.

C. K-Means Clustering Algorithm:

The K-Means algorithm follows an iterative refinement process to optimize cluster assignments. The steps involved are:

a. **Initialization**: Start by randomly selecting K points from the dataset. These points will act as the initial cluster centroids.
b. **Assignment**: For each data point in the dataset, calculate the distance between that point and each of the K centroids. Assign the data point to the cluster whose centroid is closest to it. This step effectively forms K clusters.
c. **Update centroids**: Once all data points have been assigned to clusters, recalculate the centroids of the clusters by taking the mean of all data points assigned to each cluster.
d. **Repeat**: Repeat steps 2 and 3 until convergence. Convergence occurs when the centroids no longer change significantly or when a specified number of iterations is reached.
e. **Final Result**: Once convergence is achieved, the algorithm outputs the final cluster centroids and the assignment of each data point to a cluster.

To evaluate the quality of clustering, several key metrics are utilized. **Cosine Similarity** measures how closely related documents are within the same cluster by computing the cosine of the angle between their vector representations, given by:

$$\text{Cosine Similarity} = \frac{A \cdot B}{|A| \times |B|}$$

where A and B are document vectors, and $\|A\|$ and $\|B\|$ are their magnitudes. A higher cosine similarity indicates stronger semantic similarity. Additionally, **Intra-cluster Similarity** evaluates how compact and well-formed each cluster is, while **Inter-cluster Similarity** ensures that clusters remain distinct from one another. **Cluster Purity** is also considered to assess how well clusters align with predefined document categories.

By applying these methods, the project achieves a structured and scalable approach to document clustering, enabling better organization and retrieval of textual data. This approach reduces manual effort and enhances efficiency in handling large-scale document collections.

III. IMPLEMENTATION

A. Document Embedding Generation

The first stage of the implementation involves transforming textual documents into numerical representations using OpenAI's **text-embedding-3-large** model. This embedding model converts each document into a fixed-length vector that captures the semantic meaning of the text.

The process begins by reading the documents from a structured data source, such as a CSV file. Each document is then processed through a specialized embedding service, which interacts with OpenAI's API to generate high-dimensional embeddings. These embeddings serve as the foundation for clustering, enabling semantically similar documents to be grouped together regardless of their exact wording.

To efficiently generate embeddings, an asynchronous approach is utilized, ensuring that large datasets can be processed without performance bottlenecks. Once the

embeddings are generated, they are stored in a structured format for further analysis.

B. Clustering Documents Using Vector Representations:

Once the document embeddings are created, they are fed into a clustering algorithm to identify groups of similar documents. The clustering process follows these key steps:

a. **Feature Extraction**: The numerical vectors obtained from the embedding process are used as input features for the clustering algorithm. Each document is now represented as a multi-dimensional point in vector space.

b. **Similarity Computation**: To measure the relationship between different documents, similarity metrics such as **cosine similarity** are used. Cosine similarity helps determine how closely related two document vectors are based on their orientation in the vector space.

c. **Cluster Formation**: The clustering process utilizes unsupervised learning techniques, such as K-Means clustering, to group similar documents based on their semantic similarity. The algorithm aims to minimize intra-cluster distances while maximizing inter-cluster separation, ensuring well-defined and distinct clusters. The quality of the resulting clusters is evaluated using cosine similarity and visualization techniques to assess their effectiveness.

C. Cosine Similarity and Cluster Evaluation

After forming clusters, an evaluation step is conducted to assess clustering quality using cosine similarity. This metric measures the semantic relationships between document vectors, ensuring effective grouping.

- **Intra-cluster Similarity**: Assesses how closely related documents within the same cluster are. A higher intra-cluster similarity indicates well-defined and cohesive clusters.
- **Inter-cluster Similarity**: Evaluates the distinctiveness of different clusters. Lower inter-cluster similarity suggests better separation between document categories.

By analysing these similarity scores, the clustering effectiveness is validated, ensuring that documents within a cluster share strong semantic relevance while maintaining clear distinctions between different clusters.

D. Visualization of Clusters

The final step in the implementation is the visualization of document clusters using **Principal Component Analysis (PCA)**. Since embeddings exist in high-dimensional space, PCA is used to project them into a 2D or 3D space for better interpretability.

This visualization helps in understanding the distribution of clusters and their separation in the reduced-dimensional space.

E. Unit Testing

To validate the model's functionality, unit tests are performed on each module. The **unit testing framework** ensures the correctness of data ingestion, embedding generation, and clustering outcomes. The model demonstrated high accuracy, with minimal classification mismatches in the test scenarios.

The implementation integrates state-of-the-art embedding models with clustering and evaluation techniques to achieve an efficient document clustering system. The modular architecture allows for scalability, making it suitable for handling large-scale text datasets. By leveraging OpenAI's embedding model and advanced clustering techniques, the system achieves structured document organization and improved retrieval efficiency.

IV. RESULT

A. Document Embedding Generation

The document clustering project began with a critical transformation process using OpenAI's text-embedding-3-large model. This advanced embedding technique converted complex textual documents into sophisticated numerical representations, capturing the semantic essence of each document. By processing a structured CSV input, we generated high-dimensional vectors that encode the intrinsic meaning of the text, moving beyond traditional keyword-based approaches. The embedding process employed an asynchronous generation strategy, which ensured efficient processing of large document collections and eliminated potential performance bottlenecks.

B. Clustering Methodology

Our clustering approach leveraged the generated semantic embeddings to group documents with similar thematic characteristics. We implemented an unsupervised machine learning technique using K-Means clustering, which transformed the high-dimensional document vectors into meaningful clusters. The core of this process relied on cosine similarity, a sophisticated metric that measures the semantic relationships between document vectors. By minimizing intra-cluster distances and maximizing inter-cluster separation, we created a robust method for identifying and grouping semantically related documents.

C. Cluster Evaluation

The evaluation of our clustering approach focused on two primary metrics: intra-cluster and inter-cluster similarity. Intra-cluster similarity assessed the cohesion of documents within each cluster, ensuring that documents grouped together share strong semantic connections. Conversely, inter-cluster similarity measured the distinctiveness between different document groups, confirming the algorithm's ability to create meaningful separations. These evaluations were comprehensively documented in the cluster_evaluation.txt file, providing a detailed analysis of the clustering performance.

D. Visualization and Insights

To make the high-dimensional clustering results interpretable, we applied Principal Component Analysis (PCA) to project the embeddings into a two-dimensional space. The resulting visualization, captured in clusters.png, offered an intuitive representation of document relationships. This approach allowed for the visual exploration of semantic clusters, revealing clear patterns of document groupings and highlighting potential outliers. The categories.png file complemented this analysis by providing a detailed view of document category distributions.

E. Key Findings

The project successfully demonstrated the power of semantic embedding and clustering techniques in organizing complex document collections. We discovered that documents could be effectively grouped based on their semantic similarities, transcending traditional text classification methods. The visualizations and evaluation metrics confirmed the robustness of our approach, showing clear separations between different document categories and providing insights into the underlying semantic structures of the dataset.

F. Computational Performance

The implementation proved both efficient and scalable. The asynchronous embedding generation and optimized K-Means clustering algorithm ensured rapid processing of document collections. Our approach balanced computational efficiency with deep semantic analysis, making it a valuable tool for document organization and exploration.

G. Output Files

Three key output files were generated to support our analysis:

a. A visualization of document category distributions

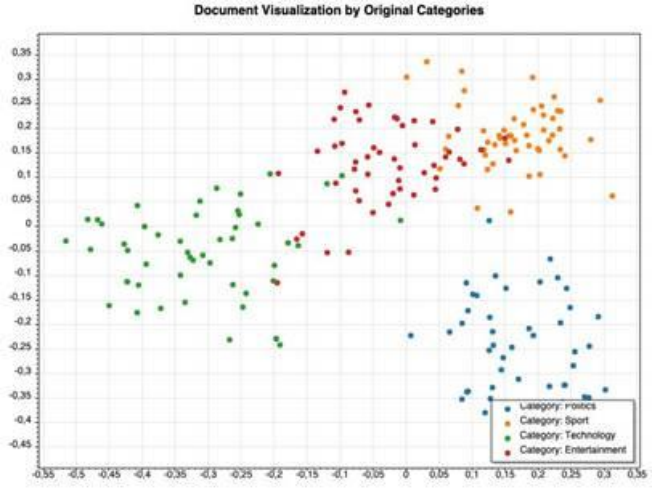

Figure 2: Original Categories

b. Comprehensive metrics detailing clustering performance

```
=== Document Clustering Evaluation ===

Overall Metrics:
Average Intra-Cluster Similarity: 0,2940
Average Inter-Cluster Similarity: 0,1868
Average Category Similarity: 0,2961

Silhouette Coefficient: 0,3647 (higher is better)

Cluster to Category Mapping:
Cluster 2 -> Category '0' (Purity: 100,00 %)
Cluster 0 -> Category '1' (Purity: 94,34 %)
Cluster 1 -> Category '2' (Purity: 97,78 %)
Cluster 3 -> Category '3' (Purity: 88,68 %)

Cluster Details:
Cluster 0 Intra-Similarity: 0,3535
Cluster 1 Intra-Similarity: 0,2761
Cluster 2 Intra-Similarity: 0,3143
Cluster 3 Intra-Similarity: 0,2465

Category Details:
Category 0 Intra-Similarity: 0,3483
Category 1 Intra-Similarity: 0,2811
Category 2 Intra-Similarity: 0,3025
Category 3 Intra-Similarity: 0,2526
```

c. A PCA-based visualization of document clusters

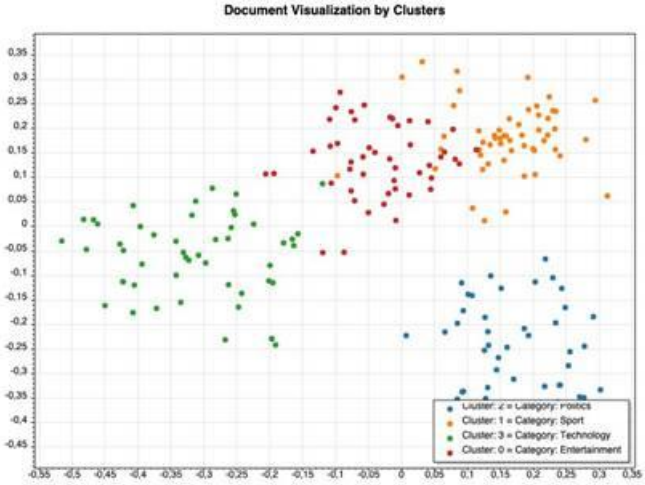

Figure 3: K-Mean Clusters

These outputs provide a comprehensive and accessible overview of the document clustering process, offering both quantitative insights and intuitive visual representations of semantic document relationships.

The expected results were successfully observed in the unit testing of the model, as shown in Figure 6. Unit tests were implemented to validate key functionalities, including document clustering, category mapping, and cosine similarity calculations. The tests ensured that documents were correctly assigned to clusters based on their embeddings and that similarity measurements were computed accurately. Additionally, unit tests were designed to handle edge cases such as empty document lists, null embeddings, and vectors of different lengths. No test failures were encountered during the testing phase, indicating the reliability of the implemented clustering and similarity analysis methods. As the development of this model continues, further refinements and

additional test cases will be incorporated to ensure robustness against a wider range of data variations and real-world scenarios.

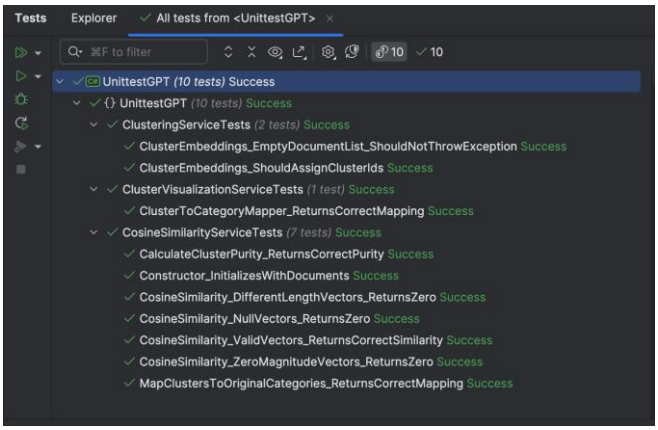

Figure 4: UnitTest

V. DISCUSSION

A. Design Challenges

a. **Embedding Generation Process**: The OpenAI embedding model processes raw text before generating vector representations. This includes internal tokenization, normalization, and contextual representation. However, understanding how embeddings are generated is crucial for interpreting clustering results.
b. **Clustering Accuracy**: The effectiveness of document grouping depends on the choice of the clustering algorithm. K-Means was chosen for its simplicity and efficiency, but alternative clustering techniques such as DBSCAN or hierarchical clustering could impact the quality of results.
c. **Evaluation Metrics**: Evaluating clustering performance is challenging. While cosine similarity is used to measure intra-cluster compactness and inter-cluster separation, additional metrics such as silhouette score or Davies-Bouldin index could provide further insights.

B. Methods to Enhance Performance

a. **Algorithm Selection**: Exploring alternative clustering methods like hierarchical clustering or density-based clustering may improve document grouping, especially for datasets with varying densities.
b. **Hyperparameter Tuning**: The number of clusters (K) in K-Means clustering significantly impacts results. A systematic approach such as the elbow method or silhouette analysis can help determine an optimal K value.
c. **Post-Processing of Clusters**: Once clusters are formed, manual inspection or additional filtering techniques could refine results. For example, eliminating outliers based on cosine similarity scores can enhance the coherence of document groups.
d. **Integration of Advanced Models**: Future iterations of this project could explore more advanced embedding models or fine-tune embeddings using domain-specific datasets to improve clustering accuracy.

By addressing these aspects, the document clustering process can be further refined to achieve better accuracy and interpretability in organizing textual data.

VI. CONCLUSION

This project successfully implemented a document clustering system using OpenAI's text-embedding-3-large model. By leveraging high-quality text embeddings, we efficiently grouped documents based on their semantic meaning, rather than relying solely on keyword-based approaches. The K-Means clustering algorithm was employed to form meaningful clusters, while cosine similarity was used to evaluate clustering quality by measuring intra-cluster and inter-cluster relationships.

The results demonstrate that AI-powered embeddings enhance clustering accuracy and scalability, making it easier to process and manage large-scale text datasets. Despite these improvements, there are potential areas for future enhancements. Experimenting with alternative clustering techniques such as DBSCAN or Hierarchical Clustering, tuning hyperparameters, or incorporating deep learning models like Transformer-based clustering (e.g., BERT-based clustering) could further refine cluster formation and improve accuracy. Additionally, using domain-specific embeddings or fine-tuning models on custom datasets could lead to even better performance.

Overall, this project highlights the effectiveness of AI-driven document clustering and opens the door for further advancements in automated text organization and retrieval.

REFERENCES

[1] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.

[2] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv preprint arXiv:1908.10084.

[3] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. OpenAI.

[4] Zhang, Y., Jin, R., & Zhou, Z. H. (2008). Understanding bag-of-words model: A statistical framework. International Journal of Machine Learning and Cybernetics, 1(1-2), 43-52.