# African Cuisine Text Generation, Semantic Similarity, and Clustering

Cyrille Yelibi*

*Southern New Hampshire University, 2500 N River Rd, Manchester, NH 03106, USA*

This study explores the application of Google's Gemini large language model (LLM) for generating African cuisine recipes and analyzing their semantic relationships. We generate a dataset of short textual descriptions of African recipes. The generated data was embedded into a numerical space which we show makes it appropriate to find potential similarities between food recipes. We show that Gemini can generate semantically meaningful culinary data without fine-tuning.

## I. INTRODUCTION

We are interested in investigating LLM generated content for African cuisine. A LLM is a statistical model[1] trained on a text dataset which uses Machine Learning (ML) techniques to predict the next word in a sequence. The purpose of LLMs is to generate synthetical text which mimics human language. Developing specialized LLMs for African data presents significant challenges due to the lack of high-quality, and diverse datasets. Many generative AI models are trained on predominantly Western-centric data leading to biases and poor generalization when applied to African cultural contexts. The scarcity of large scale text data further limits the ability to fine-tune models effectively, making it difficult to generate accurate and contextually relevant outputs.

Several studies have explored similar challenges in adapting AI models to African data, particularly in areas such as speech recognition and domain-specific generative modeling. Owodunni et al. [12] investigated how machine learning models can be fine-tuned to adapt to African linguistic variations. Their work on automatic speech recognition (ASR) used zero-shot learning to enable models to recognize diverse African accents without explicit training on each accent. A study on speech recognition in clinical settings [11] revealed significant performance gaps in global ASR models when applied to African dialects. The research emphasized that pretrained models perform better when exposed to more diverse datasets, reinforcing the need for region-specific data collection.

Prior work [10] addresses the challenge of aligning items between two distinct food composition databases through their textual descriptions. To achieve this, they developed a word embedding model trained on a comprehensive corpus of 267,071 recipes from various sources, including BBC Food Recipe, Epicurious, Cookstr, and AllRecipes. This model captures semantic relationships between food items, facilitating the identification of equivalent entries across the two databases.

These studies illustrate the broader issue of AI performance on African datasets and the necessity of refining generative models for regional specificity. They also highlights the importance of testing AI models on underrepresented data to assess their capabilities.

Google's Gemini is a family of multi-modal LLMs developed by Google DeepMind, designed for tasks such as text, image, code, and audio generation. Built on an advanced transformer-based neural network architecture, Gemini provides enhanced reasoning and multi-modal capabilities. Different versions of Gemini have been released, each catering to specific use cases[2]

Text embedding is a specialized form of embedding used in Natural Language Processing (NLP) to convert words, sentences, or documents into numerical vectors while preserving their semantic meaning. Text embeddings are generated using specialized models trained on vast amounts of linguistic data to capture contextual relationships. Various models have been developed to optimize this process, including: Word2Vec (Mikolov et al., 2013) – Predicts word relationships based on co-occurrence in large text corpora.[8] GloVe (Pennington et al., 2014) – Captures word associations by factorizing word co-occurrence matrices.[13] BERT (Bidirectional Encoder Representations from Transformers) – Unlike previous models, BERT understands context bidirectionally, making it highly effective for semantic understanding.[15] SBERT (Sentence-BERT) – A modified version of BERT designed for sentence-level similarity tasks.[15]

The Transformer introduced in Vaswani et al.'s 2017 paper "Attention Is All You Need"[18], revolutionized NLP by replacing traditional sequence-based models like RNNs (Recurrent Neural Networks)[6] and LSTMs (Long Short-Term Memory networks)[16].

Because of the Transformer advancements[3], transformer-based models like BERT[15], GPT[3],

* cyelibi@gmail.com

[1] for a review on LLMs see [2]

[2] Gemini-2.0-flash: optimized for next generation features, speed, and multimodal generation. Gemini-2.0-flash-lite: A Gemini 2.0 Flash model optimized for cost efficiency and low latency Gemini-1.5-flash: Fast and versatile performance across a diverse variety of tasks Gemini-1.5-flash-8b: High volume and lower intelligence tasks Gemini-1.5-pro: Complex reasoning tasks requiring more intelligence

[3] The key innovations of the transformer architecture include: Self-Attention Mechanism – Instead of processing words sequentially, transformers analyze relationships between all words in a sentence at once. This allows the model to capture long-range dependencies in text more effectively than RNNs. Positional Encoding – Since transformers do not process input sequentially like

and SBERT[15] achieve state-of-the-art performance in various NLP tasks, including text embeddings, machine translation, and sentiment analysis.

In this project we are using a LLM to generate African recipes, which we then embed into numerical dense data. We then show that by estimating a correlation matrix which is then mapped to a network we are able to partition the dataset into communities. The rest of this paper is structured as follow: in section II we discuss Machine Learning tools we use to accomplish our objective, section III focuses on toy problems, and our african recipes use-case, and finally in section IV we summarize, discuss the study result.

## II. IMPLEMENTATION

### A. Text Generation Process

Text generation in our project involves using Google's Gemini-1.5-Flash model to generate unstructured recipe data. The model follows a multi-step procedure involving encoding, probability estimation, and token sampling before producing the final output. The following steps outline the process of generating text using Gemini:

1. Prompt Encoding: From Text to Numerical Representation We provided a prompt as a text input in the first step to allow the model to encode the input into a format the Transformer model can process. This involves:
   (a) Tokenization: The text is broken down into smaller components called tokens (words, subwords, or characters). Modern LLMs often use subword tokenization techniques like Byte-Pair Encoding (BPE) or SentencePiece, which allow efficient handling of rare and out-of-vocabulary words. For instance: Input: "Give me the detailed recipe for dish including the ingredients and preparation steps under 300 words and be less verbose." Tokens: [”Give”, ” me”, ” the”, ” detailed”, ” recipe”, ” for”, ” ”, ”dish”, ””, ” including”, ” the”, ” ingredients”, ” and”, ” preparation”, ” steps”, ” under”, ” 300”, ” words”, ” and”, ” be”, ” less”, ” verbose”, ”.”]
   (b) Conversion to Numerical Vectors: Each token is mapped to a unique integer ID using a pre-trained vocabulary. This step turns the text into a sequence of numbers that the model can process.
   (c) Embedding Layer: The token IDs are then converted into dense numerical vectors using an embedding matrix. The embedding layer maps each token ID to a high-dimensional numerical representation, allowing the model to understand relationships between words.
2. Probability Estimation: How the Model Predicts the Next Word Once the input tokens are embedded, the model estimates the probability distribution over the entire vocabulary for the next possible word. This is done using self-attention in the Transformer architecture:
   (a) Self-Attention Mechanism: Each token attends to other tokens in the input sequence to determine context. The model calculates attention scores that indicate how much importance each word should have in predicting the next token.
   (b) Probability Computation: The output of the self-attention layers is fed into a soft-max function, which converts the raw scores into probabilities over the model's vocabulary.
3. Token Sampling: Multiple words can have nonzero probabilities, the model must sample the next token intelligently. This is controlled by Top-K and Top-P sampling parameters:
   (a) Top-K Sampling: The model considers only the K most probable words instead of the entire vocabulary. If K=5, only the 5 highest-probability words are considered. This prevents extremely rare or irrelevant words from being chosen.
   (b) Top-P (Nucleus) Sampling: Instead of choosing a fixed number (K), this method selects words dynamically based on cumulative probability. If P=0.9, the model selects the smallest set of words whose total probability exceeds 0.9. This makes the generation more flexible and adaptive to different prompts.
   (c) Final Token Selection: Once the candidate words are filtered, the model picks the next word randomly but weighted by probability. If temperature=0, the model always picks the most probable word. If temperature=1.0, words are picked based on their probability, adding randomness.
4. Iterative Generation: Expanding the Sentence Once the next word is chosen, it is appended to the input sequence, and the process repeats. The model continues generating tokens until it reaches a predefined max token length. A stopping condition is met (a period or newline). A special end-of-sequence token is generated.

RNNs, positional encodings are added to input tokens to retain word order information. Multi-Head Attention – The model can focus on multiple aspects of a sentence simultaneously, enabling richer contextual understanding. Parallel Processing – Unlike RNNs, which process text step by step, transformers compute all token relationships in parallel, leading to faster and more efficient training.

This process balances structure and randomness, ensuring that generated recipes remain diverse while still being semantically meaningful.

### B. Embedding Process

We use Google's text-embedding-004 model[4] to transform the text of the recipe into embeddings. It is the default text embedding model provided by the Google GenAI library. The steps in the creating the embedding are the following:

1. Input Processing: The embedding process begins with an input text. The input is then encoded following a process similar to the tokenization of the text generation step. Embeddings here are used to map text into a high-dimensional space where similar sentences are close together.

2. Contextual Encoding: The text-embedding-004 model processes the token embeddings through a Transformer-based network to capture deeper meaning:

   (a) Self-Attention Mechanism: Determines how much each token relates to others in the sequence.

   (b) Contextual Representation: The model recomputes embeddings based on relationships between words.

3. Probability Estimation and Vector Normalization: The processed embeddings go through vector normalization so that they can be meaningfully compared:

   (a) The model converts the contextualized token embeddings into a single fixed-size vector representing the entire input text.

   (b) Normalization: The vector is scaled so that its values fall within a certain range (e.g., between -1 and 1), making distance computations more reliable.

   Google's `text-embedding-004` model supports different embedding tasks depending on how the vectors are used[4].

### C. Community Detection

Unsupervised learning is a type of machine learning where patterns are discovered from data without explicit labels or supervision. One key application of unsupervised learning is community detection, which involves identifying clusters or groups within a network.

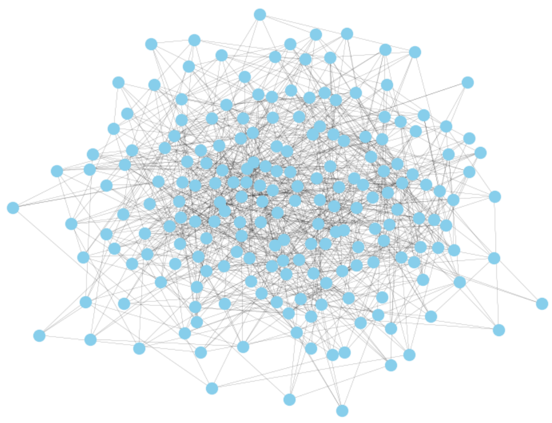
FIG. 1. Erdős–Rényi (ER) Random Graph Model

Graphs are mathematical structures used to model pairwise relations between objects. A graph consists of nodes (or vertices) representing entities and edges representing relationships between them. Graph theory is widely used in various applications, including social networks, air traffic networks, road networks, and epidemic modeling.

A community in a graph is a group of nodes that are more densely connected to each other than to the rest of the network. In the context of this study, communities represent groups of recipes that share similar ingredient compositions or cultural influences. Identifying these communities provides insights into regional culinary structures and ingredient usage patterns.

Graphs can be categorized into random graphs which exhibit no inherent structure and whose edges are assigned probabilistically, and complex networks which contain structured communities, where nodes are densely connected within groups.

The Louvain Algorithm is a hierarchical agglomerative clustering method that detects communities in networks by optimizing a quality metric called modularity[5].

## III. RESULTS

### A. Correlation Graphs

The Erdős–Rényi (ER) model[5] is a fundamental type of random graph, where edges between nodes are assigned

[4] II

[5] It measures the density of edges inside communities compared to a random graph with the same degree distribution.

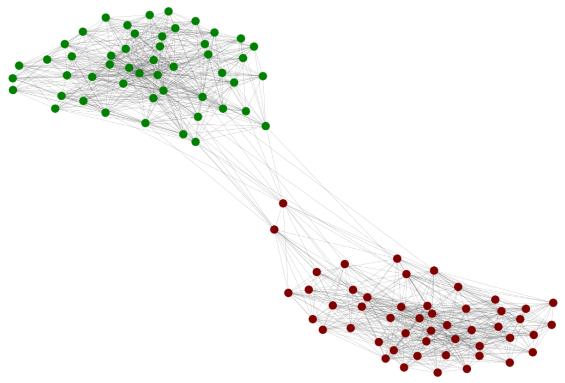

FIG. 2. Network of Weak Correlation

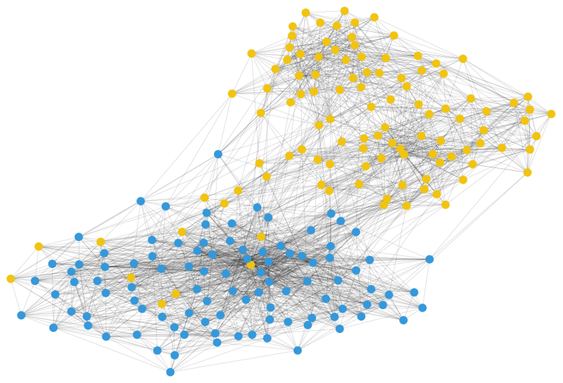

FIG. 3. Network of Medium Correlation

with a given probability. These graphs typically exhibit no apparent structure, as edges are formed independently at random. As the edge probability increases, the likelihood of node connectivity rises, eventually forming what is known as a giant component, where a significant portion of the nodes becomes interconnected.

Figure 1 shows an ER graph generated with probability $p = 0.05$. Unlike modular networks, ER graphs lack clustering structure and do not exhibit community formation. Using our text generator we generate 50 text samples from two independent topics: "Cars", "Public Health" totaling 100 samples, we create embeddings and estimate their correlation matrix which is then mapped to a network. Figure 2 shows two clusters which are well separated with a small number of edges connecting them as expected. We then pick two semantically connected topics: "CPU clock speed" , "Performance Benchmark Score" and generate 200 samples from both topics. Figure 3 illustrates a moderate correlation between two data sets. Clusters are separated but share a non-negligible number of links. Finally we generated 200 text samples using topics: "NVIDIA", "GPU". Figure 4 represents a strong correlation between two data sets. Both clusters

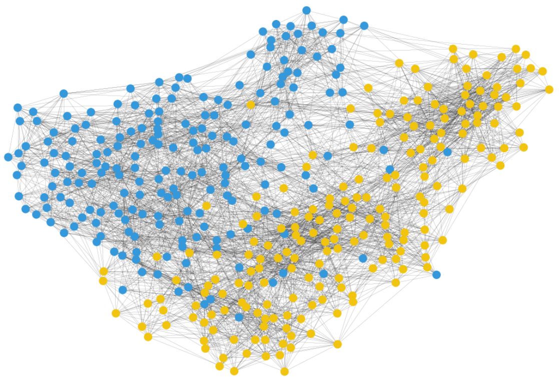

FIG. 4. Network of Strong Correlation

share a very large number of connections making is challenging to cleaning separate them.

## B. African Cuisine

```
**Koussou (approx. 6 servings)**
**Ingredients:**
* 1 cup semolina flour (fine)
* 1/2 cup all-purpose flour
* 1 tsp baking powder
* 1/4 tsp salt
* 1/4 cup melted butter
* 1 cup warm water (approx.)
* 1/4 cup sugar (optional)
* 1 tbsp anise seeds (optional)
**Preparation:**
1.  Combine semolina, all-purpose flour, baking
powder, salt, and sugar (if using) in a large bowl.
2.  Add melted butter and gradually incorporate
warm water, mixing until a soft, slightly sticky
dough forms.  Add more water if needed.
3.  Knead the dough gently for 5 minutes. Let rest
covered for 15 minutes.
4.  Roll the dough thinly (1/8 inch).
5.  Cut into small squares or diamonds.
6.  Heat a lightly oiled griddle or frying pan
over medium heat.
7.  Cook koussou until golden brown on both
sides, flipping once.
8.  Serve warm, optionally dusted with
powdered sugar and/or accompanied by honey
or jam.
**Note:**  Anise seeds can be mixed into the
dough for extra flavor. Adjust water as
needed to reach desired dough consistency.
```

FIG. 5. Traditional Koussou Recipe

The dataset for the recipes was generated in 2 steps: We first prompt the model to generate 20 recipe names for every country, and then prompt it again this time with the dish and country names to generate the actual recipe. An example can be seen in Fig 5 in which the generated text contains the dish name, a detailed list of ingredients and preparation instructions. We hope that embedding models are able to capture the similarities which exists in both the ingredient lists and the cooking steps in a meaningful way.

The generated text data is stored in a list, passed to the encoder-only model which generates the numerical embedding.

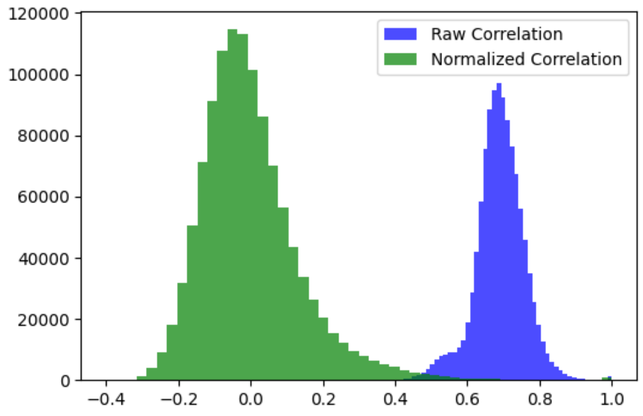

FIG. 6. distribution of Pearson corelation coefficient. In blue the raw data and in green the standardized data

We estimate the correlation matrix of the numerical embedding. In figure 6 we show histograms of two correlation matrices. One for the raw data and the other for the normalized data. We notice the raw data histogram is right-skewed with an average around 0.7 suggesting a significantly highly overall correlation between observations. Normalizing the data shifts the histogram to the left and the mean of the distribution is now 0. Standardizing the data before estimating the correlation matrix has the effect of removing a buffer signal within the data which raises the average correlation, and has the potential to improve the separation of clusters.

### C. Network Representation and Visualization

We created a k-nearest neighbors (KNN) graph with k=15 neighbors. Our expectations is that nodes closer to each other on the graph means they share similarity. In Figure 7 a visual inspection shows dense groups of nodes located at the periphery of the graph signaling the presence of clusters. Visual inspection is insufficient however as the position of the nodes on the graph is essentially due to NetworkX's spring layout. We also observe that unlike the networks in Figures 2 we observe many connections between all nodes which suggests our recipe networks shares similarities with Figure 4 and 1. We execute Louvain on our graph and we obtain 14 clusters.

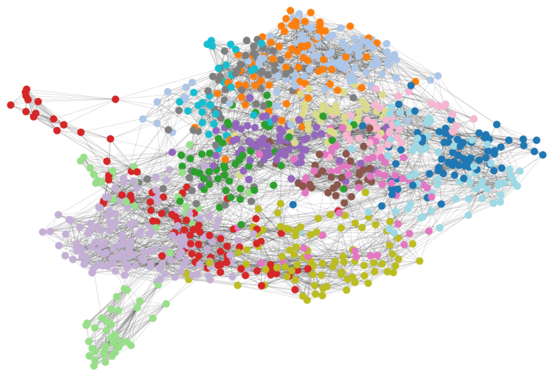

FIG. 7. KNN Graph of LLM-generated African cuisine recipes, cluster colors by Louvain.

The frequency of these labels is 0: 68, 1: 51, 2: 34, 3: 111, 4: 61, 5: 124, 6: 65, 7: 67, 8: 41, 9: 93, 10: 168, 11: 57, 12: 87, 13: 35 where the clusters were assigned an integer label from 0 to 13 and the frequency showed clusters have different sizes. We observe that the number of clusters being 14 is significantly smaller than 54 the number of African countries. This suggests that African cuisine recipes have semantic similarity which goes beyond national and possibly regional borders.

## IV. DISCUSSION AND CONCLUSION

The result of this study on LLMs showed us that LLM are capable of generating recipes that are unstructured data. We were able to generate a dataset of 1062 African recipes. This dataset would be used to generate numerical embeddings. The text-embedding model produced numerical embeddings which we use to estimate correlation matrices. We created knn graphs whose edges are the pairwise correlation coefficients between recipes. We applied the Louvain algorithm to detect clusters in the dataset and obtained 14 clusters showing that African food can be partitioned into large groups of similar recipes.

While Gemini is a powerful model it remains a general purpose language model, the quantity and the quality of generated content could increase if we use either a significantly larger model, or a fine-tuned model that is designed for cuisine. We also worked with the Flash version of the Gemini which is a model appropriate for the fast generation of text but not necessarily the best model of the Gemini family (i.e. Pro). Or more recent models such as Gemini-2.0-flash. The text-embedding model task we used was "semantic similarity". Gemini offers a "clustering" mode which may be more appropriate for clustering.

Multiple directions for future work exist: Non-text data such as pictures and videos can be generated with other generative AI models. An in-depth analysis of the recipes found in the largest clusters for semantic interpretation. Finally while the focus is on Gemini, the methodology can also be applied to other generative AI models, such as LLAMA [17], OpenAI GPT[14], Anthropic Claude[1], Mistral[9] or even DeepSeek [7].

[1] Anthropic (2025). Claude. `https://www.anthropic.com`. Large language model.

[2] Author(s) (2023). A review on large language models: Architectures, applications, taxonomies, open issues and challenges. *IEEE Transactions on Artificial Intelligence*, 4(3):1–20.

[3] Baktash, J. A. and Dawodi, M. (2023). GPT-4: A review on advancements and opportunities in natural language processing. *arXiv preprint arXiv:2305.03195*.

[4] Cloud, G. (Accessed: 2025-02-21). Text embeddings api - vertex ai. Online resource.

[5] Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3–5):75–174.

[6] GeeksforGeeks (2023). Self-attention in nlp. Accessed: 2025-02-21.

[7] Guo, D., Zhu, Q., Yang, D., Xie, Z., Dong, K., Zhang, W., Chen, G., Bi, X., Wu, Y., Li, Y., et al. (2024). Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

[8] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[9] Mistral AI (2025). Le Chat. `https://mistral.ai/news/all-new-le-chat`. AI assistant.

[10] Morales-Garzón, A., Gómez-Romero, J., and Martin-Bautista, M. J. (2020). A word embedding model for mapping food composition databases using fuzzy logic. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 635–647.

[11] Olatunji, T., Afonja, T., Yadavalli, A., Emezue, C. C., Singh, S., Dossou, B. F. P., Osuchukwu, J., Osei, S., Tonja, A. L., Etori, N., and Mbataku, C. (2023). Afrispeech-200: Pan-african accented speech dataset for clinical and general domain asr. *arXiv preprint arXiv:2310.00274*.

[12] Owodunni, A. T., Yadavalli, A., Emezue, C. C., Olatunji, T., and Mbataku, C. C. (2024). Accentfold: A journey through african accents for zero-shot asr adaptation to target accents. In *Findings of the Association for Computational Linguistics: EACL 2024*.

[13] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[14] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *OpenAI*.

[15] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[16] ScienceDirect (Accessed: 2025-02-21). Long short-term memory network. Online resource.

[17] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, , and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

## APPENDIX

TABLE I. Text Generation Parameters

| **Parameter** | **Description** |
|---|---|
| Max Output Tokens | Controls the maximum length of generated text. |
| Temperature | Regulates randomness in generation; lower values yield more deterministic outputs. |
| Top-K Sampling | Restricts token selection to the top $K$ probabilities. |
| Top-P Sampling | Dynamically selects tokens based on cumulative probability until a threshold $P$ is reached. |
| Seed | Ensures reproducibility by initializing the random number generator. |
| Frequency Penalty | Reduces repetition in generated text. |
| Presence Penalty | Encourages introducing new topics in the output. |

TABLE II. Text Processing and Retrieval Task Types

| **Task Type** | **Description** |
|---|---|
| SEMANTIC_SIMILARITY | Assess text similarity |
| CLASSIFICATION | Classify texts according to preset labels |
| CLUSTERING | Cluster texts based on their similarities |
| RETRIEVAL_DOCUMENT, RETRIEVAL_QUERY, QUESTION_ANSWERING, FACT_VERIFICATION | Document search or information retrieval |
| CODE_RETRIEVAL_QUERY | Retrieve a code block based on a natural language query, such as sort an array or reverse a linked list. *Embeddings of the code blocks are computed using* RETRIEVAL_DOCUMENT. |