Phylogeny-Inspired Sentence Semantic Clustering

Dongliang Zhang / dz17@illinois.edu
Yejun Park / yejunp2@illinois.edu
Mingyi Lin / mingyil3@illinois.edu

1 Introduction

Sentence semantic clustering is a natural language processing task with various potential applications. With the development of large language models and representation learning, one straightforward solution is to embed all sentences in the source dataset into an Euclidean feature space, and then perform clustering algorithms like K-Means and DBScan on it. However, this solution depends heavily on the quality of language models used and is susceptible to occasional failures, because embeddings produced by language models may not always capture the true semantic relationships between sentences, leading to suboptimal clustering results. A better sentence-wise similarity metric should be designed and utilized for the task.

To this end, we aim to incorporate new terms into the similarity metric, inspired by several algorithms in computational biology. One of the most important is measuring structural similarity by sequence alignment. After that, a phylogeny algorithm, namely Neighbor Joining, could be applied, yielding a tree with leaves as original sentences, which represents the hierarchical semantic clustering. This method could provide a more reliable and explainable alternative to conventional language-model-based techniques, with increased adaptability across different domains and linguistic contexts.

Our code is currently available at: https://github.com/Y3JUN/CS-466-Project

2 Method

2.1 Dataset generation

To facilitate our study on sentence semantic clustering inspired by phylogenetic methodologies, we generated a comprehensive dataset grounded in computational biology. This process leveraged the capabilities of large language models (LLMs) to systematically create structured content, ensuring the dataset's relevance and diversity. The dataset generation involved three main steps: outline generation, textbook content creation, and question generation.

Step 1: Outline Generation

We began by prompting an LLM to generate an outline of the computational biology subject in JSON format. The outline encapsulated key topics and subtopics, ensuring comprehensive coverage of the domain. For instance, the section on "sequence analysis" included subtopics like "dynamic programming," "global alignment," "local alignment," and "multi-sequence alignment," among others. This structured framework served as the foundation for the subsequent content creation stages.

Step 2: Textbook Chapter Creation

Using the generated outline, we have developed a hierarchical directory structure to organize the content. For each subtopic, an LLM was tasked to generate a detailed textbook chapter. A templated approach was employed, where placeholders in a pre-defined text generation prompt were dynamically replaced with the corresponding topic names. The resulting chapters were stored in the appropriate directories, maintaining a clear and organized repository of computational biology knowledge. To ensure robustness, multiple retries were implemented to handle potential failures in the content generation process. If a subtopic failed to produce content after five retries, it was logged and excluded from the dataset.

Step 3: Question Generation

To fill the sentence dataset with query-oriented data, we employed another LLM to generate approximately 10 questions for each textbook chapter. These questions were designed to capture the essence of the chapter's content and promote engagement with the material. The question generation process involved incorporating the textual content of each chapter into a predefined prompt template. The LLM processed this input to produce a set of questions tailored to the specific subtopic. As with textbook generation, a retry mechanism was implemented to address potential failures. The generated questions were then collated and stored alongside the corresponding textbook chapters.

Dataset Consolidation

The final dataset combined the structured outline, the generated textbook chapters, and the associated questions (sentences to be clustered). By iteratively running the generation process, we ensured both the depth and breadth of the dataset. This structured dataset provided a rich foundation for evaluating and refining our sentence semantic clustering methodology.

2.2 Basic Setting: Edit-Distance Based, Single Threaded Implementation

In this section, we implemented the foundational implementation of sentence semantic clustering, utilizing an edit distance-based approach combined with the Neighbor Joining (NJ) algorithm. This implementation, developed in Python, operates in a single-threaded environment and serves as the first step in constructing a phylogeny-inspired semantic clustering model.

2.2.1 Distance Metric: Edit Distance

The core of the semantic clustering approach relies on the computation of pairwise sentence similarities. For this purpose, we use the Levenshtein (edit) distance, which quantifies the dissimilarity between two strings by calculating the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into the other. The computed edit distance serves as the primary metric for sentence similarity in the subsequent clustering algorithm.

To calculate the pairwise distances, a distance matrix is constructed, where each entry represents the edit distance between a pair of sentences. The matrix is symmetric, with diagonal elements set to zero since the distance between a sentence and itself is always zero.

2.2.2 Hierarchical Clustering: Neighbor Joining Algorithm

With the distance matrix in hand, we proceed to apply the Neighbor Joining (NJ) algorithm to generate a hierarchical clustering of sentences. The NJ algorithm operates by iteratively merging pairs of sentences (or clusters) that minimize a specific criterion, the Q-matrix, which reflects the distances between elements relative to the rest of the dataset.

The algorithm begins by initializing a distance matrix and then computes the Q-matrix, where each entry represents a potential cost for merging two clusters. At each iteration, the pair with the smallest Q-value is selected for merging. This process continues until only two clusters remain, resulting in a hierarchical tree that can be visualized as a phylogenetic tree.

2.2.3 Visualization

The hierarchical tree produced by the NJ algorithm is output in Newick format, a standard format for representing phylogenetic trees. The tree structure is visualized using the Bio.Phylo library, which supports the rendering of phylogenetic trees. The visualization displays sentence clusters as branches, with labels derived from the original hierarchical topics.

2.2.4 Evaluation: Heatmap and Silhouette Score

To evaluate the quality of the sentence clustering, we generate a heatmap of the distance matrix, which visually represents the pairwise distances between sentences. A grouped heatmap is created, where sentences are sorted according to their topic groupings. This heat map provides an intuitive overview of how well sentences cluster together based on their semantic similarity.

Additionally, we assess the clustering quality using the silhouette score, which measures how well-separated the clusters are. The silhouette score ranges from -1 to 1, with higher values indicating better clustering.

2.3 Optimization 1: Parallel Computing

In our optimization efforts, we aimed to accelerate the computation-intensive steps in the sentence semantic clustering process, particularly focusing on the distance matrix calculation and the Q-matrix computation, both of which involve pairwise operations that can be parallelized. By leveraging multi-core processing, we significantly reduced the runtime of these operations.

2.3.1 Parallelizing the Distance Matrix Calculation

The primary bottleneck in our original implementation was the pairwise calculation of the edit distances between sentences. To address this, we utilized the joblib library's **Parallel** and **delayed** functions to compute the upper triangle of the distance matrix in parallel. In this optimization, the function that computes the distance matrix was modified to compute the pairwise distances across all sentences concurrently using multiple CPU cores. This parallelization was particularly beneficial when dealing with large datasets, as it drastically reduced the time taken to compute the entire distance matrix.

2.3.2 Parallelizing the Q-matrix Computation in the Neighbor Joining Algorithm

The second critical optimization was the parallelization of the Q-matrix computation in the Neighbor Joining (NJ) algorithm. The Q-matrix is an essential component in the hierarchical clustering process, as it helps determine which clusters to merge. Given that the Q-matrix calculation involves iterating over all pairs of sentences or clusters, this step was another performance bottleneck.

To accelerate this, we modified the **compute_q_matrix** method to compute parts of the Q-matrix concurrently. This was achieved by breaking the computation into smaller tasks, each of which calculates a line of the Q-matrix, and then combining the results in a single Q-matrix. The reason why we choose to compute a single line within each function call instead of a single entry of the Q-matrix is that, it takes only a few linear operations to compute each entry, which might be too light-weight compared to a CPU context switch. Therefore, we implemented a batched operation for this task to avoid the great multi-processing overhead..

2.4 Optimization 2: Integrating LLM Euclidean Distance

Other than edited distance, we also want to see how LLM can perform in this situation. We used sentence-transformers_paraphrase-multilingual-MiniLM-L12-v2 to perform feature embedding on our dataset, and use the embedded values to calculate the Euclidean distance between sentences. Then, we want to observe how this way of calculating distance matrix will affect our result. We will use this matrix as part of the linear combination with the edited distance matrix from class, and then explore how linear combination will affect final clustering results.

3. Validation Results

3.1 Neighbor Joining Algorithm Basic Settings Validation

To validate our approach, we applied the Neighbor Joining (NJ) algorithm to a structured dataset consisting of 50 sentences across five distinct topics: sequence alignment, phylogenetics, genomics, machine learning in bioinformatics, and protein structure. The goal was to construct a phylogenetic tree that represents hierarchical semantic clustering and to evaluate the clustering quality using metrics like the silhouette score. The source code corresponding to this section is available at: https://github.com/Y3JUN/CS-466-Project/blob/main/Neighbor_Joining.ipynb

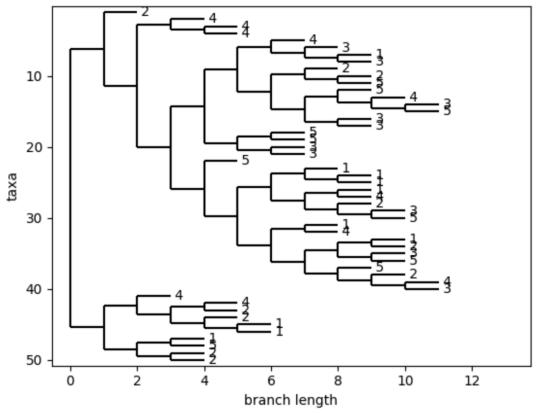

Figure 1

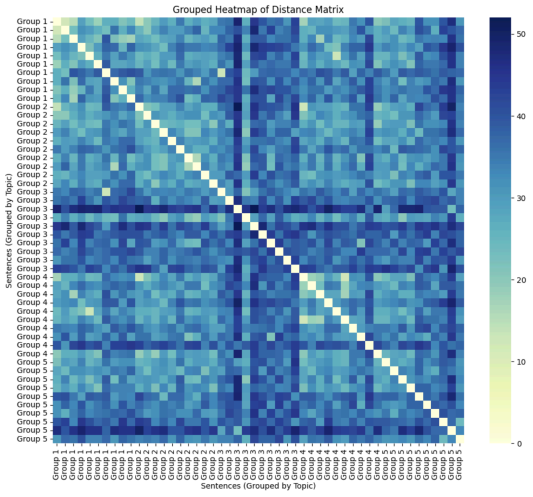

Figure 2

1. **Phylogenetic Tree:**

The resulting tree (Figure 1) visually represents hierarchical relationships among the sentences. Leaves of the tree correspond to the original sentences, grouped based on structural similarity. Branch lengths indicate the relative similarity, with shorter lengths representing closer relationships. For example, sentences within the "sequence alignment" and "phylogenetics" topics formed tighter clusters.

2. **Distance Matrix Heatmap:**

The heatmap (Figure 2) illustrates clear patterns of similarity, with distinct blocks corresponding to sentences from the same topic. However, some overlaps exist, reflecting the limitations of the edit distance metric for capturing nuanced semantic differences.

3. **Clustering Quality:**

The silhouette score of 0.131 indicates weak clustering. While some topics (e.g., "alignment") were tightly grouped, others (e.g., "machine learning in bioinformatics") exhibited significant overlap with related topics.
This score highlights the challenge of relying solely on structural similarity for semantic clustering.

3.2 Clustering Entropy Metric for Optimized Implementations

Notice: For running time purposes and algorithm efficiency, we only choose the first 200 entries from our dataset.

Based on the tree generated from the Neighbor Joining Algorithm, we hence try to measure clustering quality by calculating Entropy for each of the four given clusters. Each group will have entropy of approximately $\log(\frac{1}{6}) \leq H(x) \leq 0$ by information theory, where $H(x) = 0$ indicate

perfect grouping. We tried 3 linear combinations of edited distance matrix and Euclidean weighted distance matrix.

Using purely edited distance results in $H(x) = -0.4099715126259381$. Using purely Euclidean results in $H(x) = -0.34837415990080495$. Using Edited Distance + Euclidean Distance results in $H(x) = -0.40619040140324164$. Looking at all 3 entropy, Euclidean distance has the best clustering result while edited distance has the worst.

4. Conclusion

In this project, we proposed a novel approach to sentence semantic clustering inspired by methods from computational biology, particularly leveraging the Neighbor Joining (NJ) algorithm. Our approach utilized Levenshtein edit distance and Euclidean distance (from sentence embeddings) as distance metrics, incorporating both into a hierarchical clustering framework to achieve more reliable and explainable sentence clustering.

Through the generation of a structured computational biology dataset, we were able to explore the effectiveness of the NJ algorithm in producing hierarchical clustering based on sentence similarities. Our results showed that while the NJ algorithm provided a meaningful hierarchical representation of sentence clusters, the clustering quality was still constrained by the limitations of edit distance as the sole measure of semantic similarity. The silhouette score and clustering entropy calculations further confirmed that the clustering could be improved.

The integration of Euclidean distance through sentence embeddings showed promise in enhancing clustering performance, with Euclidean-based clustering outperforming the edit distance-based clustering in terms of entropy. The use of a linear combination of these two distance metrics further emphasized the potential for more sophisticated distance measures, such as weighted combinations, to improve clustering outcomes.

Despite the optimizations, the results indicated that the current method still faces challenges, particularly in terms of clustering quality for more complex datasets and topics. The current limitations highlight the need for further refinements in both the distance metric design and the optimization process.

5. Future Work

There are several directions from which we might perform further optimizations for this project:

1. In the non-LLM distance metrics, we are currently applying edit distance on original strings of sentences. This could be both beneficial and problematic. On the one hand, applying letter-level algorithms makes it possible to consider word prefixes and suffixes, which allows different words with similar substructures to have less distance with each other. On the other hand, words of different lengths could have the same amount of meaning, so it could be unfair to directly compare the strings. Therefore, we could try to

tokenize the strings before actually calculating the edit distance, and combine the tokenized distance with the letter-level results when possible.
2. When combining Euclidean distance with edit distance, we are currently using a simple addition. Actually, this combination could be weighted, and the factors is still to be explored.
3. In multi-processing implementation, there might still be some internal errors not revealed with our mini test set. For example, we encountered a pickle problem when testing with the full dataset, and this has not been reproduced or addressed due to time limit, since a single full-scale run takes excessive time even with multi-processing.

Acknowledgement

This paper was written based on the best of our knowledge and skill, but it may still contain errors; LLMs could be used for helping knowledge searching, program implementation and paper writeup; This project is purely intended for academic exercises. It should be completely legal once graded by faculties, and the authors are not responsible for any legal consequences.