Text Clustering Based on Large Language Models

G. Bakagianni

Master Thesis

— ♦ —

Ioannina, November 2024

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Text Clustering Based on Large Language Models

A Thesis

submitted to the designated

by the Assembly

of the Department of Computer Science and Engineering

Examination Committee

by

Georgia Bakagianni

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION

IN DATA SCIENCE AND ENGINEERING

University of Ioannina

School of Engineering

Ioannina 2024

Examining Committee:

- **Aristidis Likas**, Professor, Department of Computer Science and Engineering, University of Ioannina (Supervisor)
- **Konstantinos Blekas**, Professor, Department of Computer Science and Engineering, University of Ioannina
- **Konstantinos Vlachos**, Assistant Professor, Department of Computer Science and Engineering, University of Ioannina

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Aristidis Likas, and to John Pavlopoulos, Assistant Professor at Athens University of Economics and Business, for their invaluable guidance, encouragement, and support during this research. Their expertise and thoughtful feedback have been essential to the completion of this thesis.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Glossary

ACC Clustering Accuracy.

AHC Agglomerative Clustering.

ILP Integer Linear Program.

LLM Large Language Model.

ML Machine Learning.

NLP Natural Language Processing.

NMI Normalized Mutual Information.

TF-IDF Term Frequency - Inverse Document Frequency.

Abstract

Georgia Bakagianni, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, 2024.
Text Clustering Based on Large Language Models.
Advisor: Aristidis Likas, Professor.

Text clustering is a fundamental task in Natural Language Processing (NLP), aimed at organizing documents into groups based on their content. Traditional clustering algorithms, such as k-means and AHC, have been widely used for this purpose, but their performance in text is often limited by the inherent challenges in language understanding and extracting meaningful representations from text. Recent advancements in LLMs have opened new possibilities for improving text clustering, by leveraging their powerful language understanding and generation capabilities. This study explores the role of LLMs in enhancing the text clustering task, both through pre-clustering interventions, where LLM-generated labels are applied to traditional clustering methods, and through direct application of LLMs for text clustering.

We introduce a pre-clustering approach where LLMs generate cluster labels that inform traditional clustering algorithms, significantly boosting performance. Our experiments, conducted on 11 diverse datasets, demonstrate that this approach outperforms all developed clustering methods, including those that rely on traditional clustering algorithms with earlier text representations and more recent contextual embeddings, as well as LLM-based methods that generate key phrases that inform traditional clustering algorithms. Furthermore, we examine the direct clustering potential of LLMs, which, although not always capable of producing the exact number of required clusters, achieves the second-highest average NMI and provides human-readable labels that enhance interpretability, particularly in domains requiring clarity.

Our findings highlight the dual benefits of LLMs in text clustering: improving both clustering performance and the interpretability of results, making LLMs valuable tools for advancing the text clustering task.

Εκτεταμενη Περιληψη

Γεωργία Μπακαγιάννη, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, 2024.
Ομαδοποίηση Κειμένων με Βάση τα Μεγάλα Γλωσσικά Μοντέλα.
Επιβλέπων: Αριστείδης Λύκας, Καθηγητής.

Η Ομαδοποίηση Κειμένων αποτελεί ένα βασικό πεδίο της Επεξεργασίας Φυσικής Γλώσσας (ΕΦΓ), με στόχο την ομαδοποίηση δεδομένων με βάση τις εγγενείς ομοιότητες, χωρίς τη χρήση προκαθορισμένων ετικετών κατηγορίας. Σε αντίθεση με την Κατηγοριοποίηση Κειμένων, η οποία χρησιμοποιεί εποπτευόμενη μάθηση, η ομαδοποίηση είναι ανεπίβλεπτη και αποκαλύπτει κρυμμένες δομές σε μη δομημένα δεδομένα, βοηθώντας στην εξαγωγή προτύπων και πληροφοριών από μεγάλες συλλογές κειμένων. Οι εφαρμογές της περιλαμβάνουν τη δημιουργία περιλήψεων, την ανίχνευση θεμάτων και την κατηγοριοποίηση.

Παραδοσιακά, η διαδικασία αυτή περιλαμβάνει δύο στάδια: την αναπαράσταση του κειμένου ως αριθμητικά διανύσματα που αποτυπώνουν γλωσσικά χαρακτηριστικά, και την εφαρμογή ενός αλγορίθμου ομαδοποίησης για τη δημιουργία των ομάδων. Ωστόσο, οι παραγόμενες ετικέτες δεν είναι κατανοητές από τον άνθρωπο, απαιτώντας επιπλέον προσπάθεια για την εξαγωγή χρήσιμων πληροφοριών από τις ομάδες.

Οι εξελίξεις στα μεγάλα γλωσσικά μοντέλα (LLMs), όπως το GPT-4, ανοίγουν νέες δυνατότητες για την ομαδοποίηση κειμένων, καθώς αυτά τα μοντέλα είναι ικανά όχι μόνο να κατανοούν, αλλά και να παράγουν συνεκτικό κείμενο. Η εκτενής εκπαίδευσή τους σε μεγάλα σώματα κειμένων με δισεκατομμύρια παραμέτρους προσφέρει τη δυνατότητα βελτίωσης της απόδοσης και της ερμηνείας των αποτελεσμάτων της ομαδοποίησης.

Στη μελέτη αυτή, εξερευνούμε πώς μπορούν να αξιοποιηθούν οι δυνατότητες παραγωγής κειμένου των LLMs για την Ομαδοποίηση Κειμένων. Αρχικά, διερευνούμε την ενσωμάτωση των LLMs στην παραδοσιακή μεθοδολογία ομαδοποίησης, με στόχο τη βελτίωση της απόδοσης. Συγκεκριμένα, καθοδηγώντας τα LLMs με ρητές οδηγίες, εξάγεται για κάθε κείμενο μία ετικέτα, που αφορά το κείμενο και είναι σχετική με το κριτήριο με το οποίο γίνεται η ομαδοποίηση, όπως η πρόθεση του χρήστη, το συναίσθημα, το θέμα. Το επόμενο βήμα της μεθόδου είναι η αναπαράσταση των αρχικών κειμένων και των ετικετών ως αριθμητικά διανύσματα, τα οποία εισάγονται σε αλγόριθμους ομαδοποίησης, όπως ο Συσσωρευτικός Ιεραρχικός Αλγόριθμος Ομαδοποίησης AHC και ο k-means. Η υβριδική αυτή μέθοδος η οποία χρησιμοποιεί και τα LLMs και την παραδοσιακή πρακτική στην Ομαδοποίηση Κειμένων όταν εκτελείται με τον AHC επιτυγχάνει την καλύτερη μέση απόδοση στα πειράματα που αναπτύξαμε πάνω σε έντεκα σώματα κειμένων και με δύο μετρικές αξιολόγησης.

Επιπλέον, προτείνουμε μια νέα μέθοδο, στην οποία τα LLMs αναλαμβάνουν απευθείας την ομαδοποίηση, παρακάμπτοντας τις παραδοσιακές μεθόδους Ομαδοποίησης Κειμένων. Η μέθοδος εκτελείται σε δύο βήματα: Αρχικά, καθοδηγούμε το LLM με ρητές οδηγίες να εξάγει για κάθε κείμενο μια ετικέτα σχετική με το κριτήριο ομαδοποίησης, όπως ακριβώς στην προηγούμενη υβριδική μέθοδο. Δίνονται στο LLM οι ήδη παραχθείσες ετικέτες από προηγούμενα κείμενα, ώστε να τις λάβει υπόψη και να παράξει νέες μόνο όταν οι υπάρχουσες δεν συνάδουν με το κείμενο. Στη συνέχεια, το LLM καλείται να ομαδοποιήσει τις παραχθείσες ετικέτες με βάση ένα προκαθορισμένο πλήθος ομάδων. Αυτές οι ετικέτες σχηματίζουν τις ομάδες, προσφέροντας μεγαλύτερη διαφάνεια και καλύτερη ερμηνεία, καθώς οι ομάδες διαμορφώνονται βάσει του περιεχομένου και των χαρακτηριστικών που αναγνωρίζονται από τα LLMs. Η ερμηνεία των ετικετών αποτελεί σημαντικό πλεονέκτημα της άμεσης ομαδοποίησης κειμένων με χρήση LLMs, που απουσιάζει από τις καθιερωμένες μεθόδους. Παρόλα αυτά, αυτή η μέθοδος δεν υπερβαίνει σε απόδοση την υβριδική προσέγγιση που συνδυάζει τα LLMs με παραδοσιακούς αλγορίθμους, και δεν επιστρέφει πάντα το απαιτούμενο πλήθος ετικετών. Συνεπώς, απαιτείται περαιτέρω έρευνα για τη βελτίωση αυτής της μεθόδου.

Chapter 1

Introduction

Text clustering is a core task in NLP, aimed at grouping textual data based on inherent similarities without relying on predefined category labels [1]. Unlike text classification, which relies on supervised learning and prior knowledge, text clustering is unsupervised, revealing hidden structures within unstructured data. This exploratory process seeks to identify meaningful patterns, making it a crucial tool for deriving insights from large text corpora. Applications of text clustering include text summarization, topic detection, and classification [2]. Traditionally, this task is comprised of two components as depicted in Figure 1.1: the text representation module, which represents text as numerical vectors capturing specific linguistic features, and the clustering module, which takes the text representation as input and enables a clustering algorithm to process and group the data. This long-established practice has been foundational in text clustering and data clustering in general [1]. However, the cluster labels generated are not human-interpretable, and additional effort is required to examine the clusters and gain insights [3].

In recent years, advances in NLP, particularly with the rise of LLMs such as GPT-4 [4], have opened new opportunities for rethinking the text clustering task. These models excel not only at understanding language but also at generating coherent text, which significantly enhances NLP's capabilities [5]. Since the introduction of ChatGPT in November 2022,[1] LLMs have demonstrated exceptional performance across various NLP tasks and have evolved into versatile tools that can solve complex tasks in a

[1] https://openai.com/chatgpt/

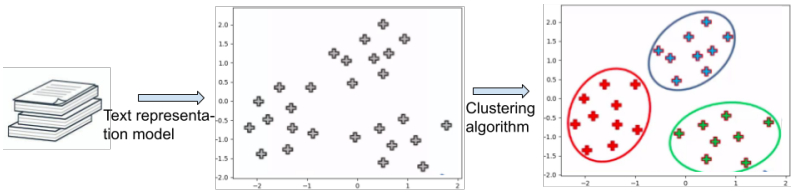

Figure 1.1: Traditional text clustering approach.

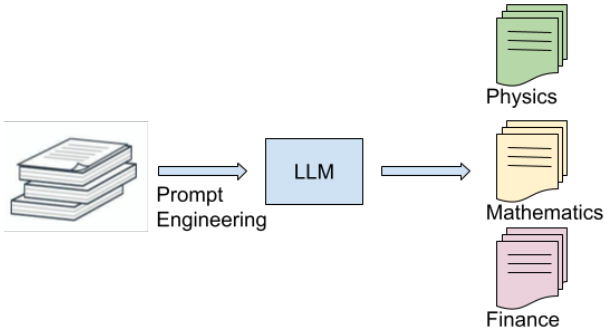

Figure 1.2: Direct LLM-based text clustering approach.

variety of domains [6].[2]. The capabilities of LLMs are enabled by extensive training on vast text corpora using billions of parameters, offering new possibilities for addressing text clustering by improving performance and enhancing the interpretability of clustering outputs.

In this study, we explore how the text generation capabilities of LLMs can enhance text clustering. First, we investigate how LLMs can be integrated into the traditional text clustering workflow to improve performance. Specifically, we examine how these models can enhance traditional clustering methods by enriching the input text prior to clustering. Building on the approach of [6], where key phrases were extracted from LLMs to improve clustering performance, we propose a more targeted method that extracts task-specific information from the texts. By guiding LLMs with explicit instructions about the clustering task and the key characteristics to focus on (e.g., topics, sentiment, or other features relevant to the grouping), we extract task-dependent information that reflects the inherent qualities of the texts. This information is encoded to generate enriched text representations that better capture the task-relevant features. These enriched representations are then passed to traditional clustering algorithms,

[2]Examples of their utility include systems such as https://copilot.microsoft.com/ and https://notebooklm.google/, which leverage the models' ability to follow human instructions, perform multi-step reasoning, and generate accurate outputs across a wide range of tasks.

such as K-MEANS and AHC. This method achieved superior performance across all eleven datasets in our experiments, outperforming both conventional methods and the approach proposed by [6].

Additionally, we introduce a novel method where LLMs directly cluster the texts, bypassing traditional algorithms and eliminating the need for numerical vector transformation. Instead, as depicted in Figure 1.2, the model performs clustering by generating descriptive, human-readable labels for each text. These labels form the clusters. This process provides greater transparency and interpretability, as the clusters are formed based on the content and characteristics identified by the LLMs. The interpretability of labels is a significant advantage of direct LLM-based text clustering, which is lacking in established methods. However, this method does not outperform the hybrid method that combines LLMs with traditional algorithms and requires further investigation to explore potential improvements.

In summary, our contributions are four-fold: (i) We propose a method that enhances traditional clustering by integrating LLMs as a pre-clustering step to provide task-specific information that enriches the input text. This approach enables a more nuanced capture of clustering-relevant features, resulting in well-defined clusters that reflect the inherent qualities of the texts and achieves the best performance among all methods in our benchmarking. (ii) We introduce a novel method where LLMs directly perform text clustering, broadening the scope of clustering methodologies by offering an LLM-based alternative to traditional, numerically-driven techniques. (iii) This direct LLM-based method also addresses the longstanding challenge of interpretability in clustering by generating human-readable labels, making clusters more transparent and accessible, thus facilitating insights without additional analysis. (iv) Finally, we provide a benchmark by comparing our proposed methods with both LLM-based and conventional approaches across diverse datasets, offering a reference for the performance and practical value of LLMs in text clustering and advancing research in unsupervised NLP tasks.

The structure of this study is as follows: Chapter 2 provides the necessary background for text clustering, beginning with traditional methods (Section 2.1) and progressing to an LLM-based approach (Section 2.2). In this chapter we also discuss the related works relevant to our study. Chapter 3 presents the clustering methods, including baselines and our proposed approaches to be assessed. Chapter 4 presents the empirical analysis conducted to assess the clustering methods, detailing

the datasets on which clustering is performed (Section 4.1), the evaluation metrics used (Section 4.2), the experimental setup defining the details of the models and configurations used in the experiments (Section 4.3), and the results of the experiments, discussing the observations made (Section 4.4). Finally, Chapter 5 summarizes the outcomes of our study and outlines future work.

Chapter 2

Background

This chapter provides essential background on text clustering, beginning with traditional methods (Section 2.1), where clustering algorithms process numerical vector representations of textual data. The discussion then progresses to clustering approaches that integrate LLMs prior to clustering to enhance clustering outcomes (Section 2.2). Finally, in Section 2.3 we discuss the related works relevant to our study, providing a taxonomy of the clustering methods, the evolution of text clustering methods, and the current research that performs text clustering using LLMs.

2.1 Traditional Text Clustering

Text clustering traditionally consists of two key components: the text representation module, which transforms text into dense numerical vectors that capture specific linguistic features, and the clustering module, which processes these representations using a clustering algorithm to group the data. In Subsection 2.1.1, we explore the evolution of text representation techniques over the past few decades. In Subsection 2.1.2, we discuss two of the most widely used clustering algorithms, AHC and

2.1.1 Text Representation

Text representation is a fundamental step in text clustering, as it transforms raw text into numerical form, enabling clustering algorithms to process it. Traditional clustering methods, such as AHC and K-MEANS, operate in a fixed feature space, where each document is represented as a vector of real numbers. The approaches used to achieve this transformation have evolved from simpler statistical methods, such as Term Frequency - Inverse Document Frequency (TF-IDF), to more advanced techniques such as general-purpose text embeddings, which capture semantic and contextual nuances of texts and can address a multitude of downstream tasks. This section explores three key text representation methods, illustrating the evolution of text representation in NLP.

Tf-Idf

One of the earliest methods for text representation is TF-IDF, a statistical measure that evaluates the importance of a word within a document relative to its prevalence across the entire corpus. It combines two metrics: Term Frequency (TF), which measures how often a word appears in a document, and Inverse Document Frequency (IDF), introduced by [7], which measures how rare a word is across the corpus. By multiplying these metrics, TF-IDF highlights words that are frequent within specific documents but rare across the corpus, making them particularly relevant to individual documents.

TF-IDF improves clustering performance by emphasizing unique terms while reducing the influence of common words, allowing clustering algorithms to better differentiate between clusters based on thematic similarity [8]. However, a key limitation of TF-IDF is that it does not capture semantic or contextual relationships within a document. Each word is treated independently, so TF-IDF cannot recognize synonyms or related terms. This limitation can cause it to overlook contextually significant words— for instance, if a synonym is used instead of a common word, TF-IDF would not identify its relevance, potentially affecting cluster quality and thematic consistency [9].

Non-Contextual Word Embeddings

Unlike TF-IDF, static word embeddings capture the semantic relationships between words. These embeddings were popularized by the work of [10] and produce dense vector representations of words based on their co-occurrence contexts in large corpora. Notable models include Word2Vec [10], FastText [11], and GloVe [12], which represent each word in a vocabulary as a single, fixed-size vector that reflects the word's semantic properties.

By leveraging these embeddings, clustering algorithms can exploit deeper semantic similarities between words, leading to more accurate text clustering. The key limitation of static embeddings is their inability to capture context-specific meanings. Each word is assigned a single vector representation, regardless of its different meanings in various contexts. For example, the word "bank" in the phrases "river bank" and "financial bank" would have the same representation, making it difficult for clustering algorithms to distinguish between different contextual uses of the word.

Contextual Word and Document Embeddings

The introduction of the Transformer architecture marked a significant advancement in text representation. Transformers [13] use self-attention mechanisms to compute attention scores for each word in a sentence, capturing not only semantic relationships but also the specific contextual relationships of words within their usage. This ability to dynamically adjust word representations depending on the context is a major strength of models built on Transformers.

The first and the most influential model using this architecture is BERT [14], which generates contextualized word embeddings. Unlike static embeddings, each word in BERT is represented in a context-dependent manner, meaning that the same word can have different embeddings based on the sentence in which it appears. This context-aware representation significantly improves performance in tasks such as clustering, where word meanings shift based on usage.

Additionally, while in all previous methods the text is represented by aggregating the embeddings of individual words, there are methods that focus on embedding the entire text (being a phrase or an entire document). These text embeddings capture the meaning of longer sequences (e.g., phrases, sentences, paragraphs, or entire documents) in their specific contexts. Furthermore, models such as INSTRUCTOR [15] take

this concept further by generating task-aware text embeddings. These models embed each input alongside task instructions, allowing for tailored representations without the need for domain-specific fine-tuning. This differs from other approaches, such as BERT or Word2Vec, where fine-tuning is often required to adapt the embeddings to specific downstream tasks and domains. In contrast, task-aware models inherently incorporate such adjustments during the embedding process.

2.1.2 Clustering Algorithms

In the following, we survey two of the most popular clustering algorithms.

Agglomerative Clustering

AHC is one of the simplest and most intuitive approaches to clustering [16], following a bottom-up hierarchical paradigm. The algorithm begins by assigning each data point to its own cluster and then proceeds iteratively, merging the "closest" clusters of the previous clustering at each step. This process reduces the number of clusters with each iteration until all data points are combined into a single cluster. The outcome is a dendrogram, a tree-like structure where individual data points form the leaves, and the final, all-encompassing cluster forms the root. The number of clusters is determined by cutting the dendrogram at a chosen level, allowing for flexible control over the final grouping.

To fully specify the algorithm, three parameters must be determined. First, the distance metric d, which measures the distance between individual points. Second, the linkage criterion D, which defines the distance between clusters and governs how they are merged. The most commonly used linkage criteria between two sets of observations A and B and a distance d are:

- Single Linkage defines the distance between two clusters as the minimum distance between any pair of points from the two sets of observations A and B.

$$D_{\text{single}}(A, B) = \min_{a \in A, b \in B} d(a, b)$$

- Maximum or Complete Linkage measures the distance between two clusters as the maximum distance between any pair of points from the two clusters. It tends to create compact and spherical clusters.

$$D_{\text{complete}}(A, B) = \max_{a \in A, b \in B} d(a, b)$$

- Average Linkage calculates the distance between two clusters as the average of all pairwise distances between points in the two clusters. It provides a balance between single and complete linkage.

$$D_{\text{average}}(A, B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$$

- Ward's Linkage calculates the distance between two clusters based on the increase in within-cluster variance when they are merged. This linkage minimizes the variance between merged clusters.

$$D_{\text{ward}}(A, B) = \frac{|A||B|}{|A| + |B|} \|\mu_A - \mu_B\|^2$$

  where μ_A and μ_B are the centroids (means) of clusters A and B, respectively.

The third parameter is the stopping criterion, which dictates when to stop merging clusters. Common stopping criteria include:

- A fixed number of clusters k: the algorithm stops merging once the number of clusters reaches k,
- A linkage distance threshold $r \in \mathbb{R}^+$: the algorithm stops merging when all the between-clusters distances are larger than r.

K-Means

K-MEANS is the most popular algorithm in the partition-based family of clustering methods [1]. Partition-based clustering methods aim to divide the dataset into a predetermined number of clusters, where each data point is assigned to one cluster based on its similarity to the cluster's centre. Compared to hierarchical clustering algorithms, partition-based clustering algorithms identify all clusters simultaneously without imposing a hierarchical structure.

These algorithms define a cost function over a parametrized set of possible clusterings, and the goal is to find a partitioning (clustering) of minimal cost. Under this paradigm, the clustering task becomes an optimization problem. The K-MEANS algorithm finds a partition such that the squared error between the empirical mean of a cluster and the points within that cluster is minimized. Let μ_k be the mean of cluster c_k. The error between μ_k and the points in cluster c_k is defined using a distance function $d(x, \mu_k)$ as:

$$J(c_k) = \sum_{x \in c_k} d(x, \mu_k)$$

The goal of K-MEANS is to minimize the sum of the errors over all K clusters $C = \{c_k, k = 1, \ldots, K\}$:

$$J(C) = \sum_{k=1}^{K} \sum_{x \in c_k} d(x, \mu_k)$$

Algorithm 2.1 K-Means

Require: Dataset $X = \{x_1, x_2, \ldots, x_n\}$, number of clusters K, cluster initialization method, distance metric d

Ensure: Cluster assignments $\{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ and centroids $\{\mu_1, \mu_2, \ldots, \mu_K\}$

1: **Initialize** cluster centroids $M = \{\mu_1, \mu_2, \ldots, \mu_K\}$ using the specified initialization method
2: **repeat**
3: **Assignment step**:
4: **for** each point $x_i \in X$ **do**
5: **for** $j = 1$ to K **do**
6: Calculate the distance: $d_{ij} \leftarrow d(x_i, \mu_j)$
7: **end for**
8: **Assign** x_i to the cluster with the nearest centroid:

$$\alpha_i \leftarrow \arg\min_j d_{ij}$$

9: **end for**
10: **Update step**:
11: **for** each cluster $j = 1, \ldots, K$ **do**
12: **Update** centroid μ_j as the mean of all points assigned to cluster j:

$$\mu_j \leftarrow \frac{1}{|\{x_i : \alpha_i = j\}|} \sum_{x_i : \alpha_i = j} x_i$$

13: **end for**
14: **until** convergence criterion is met (e.g., centroids no longer change)
15: **return** Final cluster assignments $\{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ and centroids $\{\mu_1, \mu_2, \ldots, \mu_K\}$

Minimizing this objective function is known to be an NP-hard problem (even

for $K = 2$) [1]. As an alternative, the following simple iterative algorithm [17] is often used, so frequently that the term "K-MEANS clustering" refers to the outcome of this algorithm rather than the clustering that minimizes the K-MEANS objective cost. Algorithm 2.1 describes the main steps of the K-MEANS clustering algorithm.

Even though K-MEANS was discovered independently in various scientific fields over 50 years ago by [18], [19] (proposed in 1957 but published in 1982), and [20, 21], it remains one of the most widely used algorithms for clustering. The main reasons behind the algorithm's popularity are its linear complexity, simplicity of implementation, and empirical success [22].

2.2 LLMs for Text Clustering

With the rise of LLMs and their general-purpose language capabilities, new possibilities for addressing the text clustering problem have emerged (see Subsection 2.3.3 for research papers on LLM-based text clustering). In this section, we introduce a text clustering approach that combines traditional clustering algorithms with the text generation capabilities of LLMs, laying the groundwork for one of our proposed LLM-based clustering methods.

2.2.1 LLM-Based Pre-Clustering Approach for Text Clustering

This approach uses the text generation capabilities of LLMs to extract key information from documents before applying a traditional clustering method. Instead of relying solely on traditional algorithms to identify thematic clusters, LLMs can be employed to uncover essential features, concepts, or latent themes within the text. By extracting relevant aspects prior to clustering, LLMs can enhance the overall clustering process. This hybrid model integrates LLMs by generating task-relevant insights that inform traditional clustering methods. By focusing on identifying topics, common themes, or keywords, LLMs help facilitate a more efficient and accurate clustering process. Algorithm 2.2 presents the main steps of this approach.

This method is particularly useful for large datasets or when documents exhibit overlapping themes. By highlighting unique textual characteristics, LLMs help differentiate between similar clusters, ensuring that the clusters reflect nuanced distinctions that might otherwise be overlooked. The enriched representations generated by LLMs

Algorithm 2.2 LLM-Based Pre-Clustering Approach

Require: Dataset $X = \{x_1, x_2, \ldots, x_n\}$, LLM Model, Embedding Model, Clustering Algorithm (e.g., AHC or K-MEANS)

Ensure: Cluster assignments for each text $x \in X$

Pre-clustering step:
Initialize an empty list for task-specific information: `LLM_Gen_Texts` = []
for each text $x \in X$ **do**
    Instruct the LLM Model to generate task-specific information t from x, such as key topics, common themes, or relevant key phrases.
    Append t to `LLM_Gen_Texts`
end for
Text representation module:
Initialize an empty list for enriched text representations: `Enriched_Representations` = []
for each pair (x, t) from X and `LLM_Gen_Texts` **do**
    Encode x using the Embedding Model to obtain $\texttt{Enc}(x)$
    Encode t using the Embedding Model to obtain $\texttt{Enc}(t)$
    Concatenate $\texttt{Enc}(x)$ and $\texttt{Enc}(t)$ to form $\texttt{Enriched\_Rep}(x)$
    Append $\texttt{Enriched\_Rep}(x)$ to `Enriched_Representations`
end for
Clustering module:
Apply the Clustering Algorithm to `Enriched_Representations` to generate cluster assignments:
    `Cluster_Assignments` = Clustering_Algorithm(`Enriched_Representations`)
return `Cluster_Assignments`

improve the performance of traditional clustering algorithms, such as AHC and K-MEANS, by better capturing the inherent qualities of the text.

2.3 Related Work

2.3.1 Clustering

Clustering is a fundamental task in Machine Learning (ML) and therefore is applied to various types of data, including text [23, 24, 25], image [26, 27, 28], audio [29, 30, 31], video [31, 32], graph [33, 34], time series [35, 36], and geospatial data [37, 38]. However, each data modality presents unique challenges that require specialized adaptations in data representation methods to effectively uncover the structure inherent to the data [39].

In [39], the authors categorized clustering approaches based on the interaction between the representation learning module and the clustering module. Multi-Stage Clustering treats the two processes as distinct, where representation learning and clustering are handled separately [33, 28]. Iterative Clustering refines both the representation learning and clustering processes iteratively, improving the quality of both in each iteration [26, 27]. Generative Clustering employs generative models to capture the underlying data distribution, which helps the clustering algorithm form groups based on data density [40, 41]. Lastly, Simultaneous Clustering jointly optimizes the representation learning and clustering objectives in a unified process, aiming to improve both tasks simultaneously [42, 25].

2.3.2 Text Clustering

Text clustering, in particular, presents challenges related to the high dimensionality, sparsity, and semantic ambiguity of textual data. Natural language exhibits complexity, where word meanings often change based on context, making it difficult for traditional algorithms to accurately cluster texts solely based on content. Early methods for text clustering relied on statistical techniques for text representation, including TF-IDF [43, 9] and static word embeddings [44, 45], which provided fixed representations of words, limiting their ability to capture nuanced meaning.

More recent approaches have leveraged context-sensitive, dynamic embeddings, such as BERT, which can be fine-tuned specifically for clustering tasks [46] or used without fine-tuning [47, 2]. Additionally, contextual text embeddings that are universal across tasks and domains, such as INSTRUCTOR and E5, have emerged offering strong performance on diverse clustering tasks [48, 49, 50, 51, 52].

2.3.3 LLM-based Text Clustering

LLMs have revolutionized various NLP tasks, including text clustering. Their ability to generate human-like text and understand complex language patterns offers new opportunities for improving both the accuracy of clusters and the interpretability of results.

LLMs have been integrated into text clustering in diverse ways. A pre-clustering method for intent discovery was proposed by [53], which uses LLMs to generate short summaries for utterances in dialogue systems. The process involves initial clustering using a pre-trained encoder and K-MEANS. From each cluster, a prototype utterance is selected based on its proximity to the centroid, representing the cluster's latent intent. The LLM then generates short summaries for these prototypes, which are used to prompt the LLM in classifying non-prototypical utterances by matching them to one of the descriptive labels. If none of the existing labels are appropriate, the LLM generates new short summaries. These labels are combined with the utterances into a single vector representation, and K-MEANS is applied again to refine the clustering based on these enriched representations.

The study in [50] focused on clustering interpretability by incorporating user-defined goals into the clustering process. Their algorithm takes a user-specified natural language goal (e.g., sentiment or genre) and clusters a text corpus accordingly. First, samples from the dataset are provided to an LLM, which generates potential cluster explanations based on the user's goal. Next, another LLM evaluates whether each sample from the corpus fits a given explanation by returning a "yes" or "no" response. This generates an assignment matrix, indicating which samples correspond to which candidate clusters. Finally, an Integer Linear Program (ILP) selects the optimal set of explanations, ensuring that each sample is assigned to roughly one cluster and minimizing overlap or missed assignments.

LLMs were used to understand user preferences in clustering, as described by [51]. Their approach begins with an initial clustering using contextual embeddings and then applies entropy-based sampling to retrieve triplets of samples with high uncertainty. These triplets are used to prompt the LLM to make similarity judgments, which refine the embedding space to better align with the user's clustering perspective. Further, pairwise queries sampled from a hierarchy of cluster levels are used to determine the optimal granularity of clusters, with the level yielding the high-

est consistency between the LLM's predictions and actual cluster assignments being selected.

In [52], the authors explored three stages where LLMs can enhance clustering: pre-clustering (to improve input features), during clustering (providing constraints), and post-clustering (correcting low-confidence cluster assignments). In their pre-clustering method, they used LLMs to generate key phrases that enriched document representations, making them easier to cluster. During clustering, LLMs acted as a pseudo-oracle, offering pairwise constraints that guided the clustering algorithm. Finally, after clustering, LLMs were used to refine the clusters by correcting low-confidence assignments. Their experiments revealed that pre-clustering method provided the strongest performance boost.

Chapter 3

Proposed Text Clustering Methods

This chapter builds upon the background knowledge of text clustering outlined in Chapter 2 and presents the specific methods developed to address the research task. The structure mirrors that of Chapter 2: we begin by discussing traditional NLP methods that use established clustering algorithms (Section 3.1). We then explore LLM-based clustering approaches that leverage the natural language generation capabilities of LLMs (Section 3.2).

3.1 Traditional NLP-Based Text Clustering Methods

As outlined in Section 2.1, traditional NLP clustering methods, which developed prior to the emergence of LLMs, rely on converting text into numerical vectors. This transformation enables clustering algorithms, such as k-means and AHC, to process the data and perform clustering.

This study investigates the impact of various text representation techniques on clustering performance, focusing on methods that reflect the evolution of text representation. We start with the earlier statistical techniques, selecting TF-IDF, a method that emphasizes term importance in a corpus, although it lacks the ability to capture

semantic or contextual relationships between words, which is a limitation. Next, we turn to static word embedding models that capture semantic relationships, but generate a single embedding for each word, regardless of context, and fail to capture contextual nuances. From this category, we use pre-trained embeddings from three widely recognized embedding models - Word2Vec [10], FastText [11], and GloVe [12].

Finally, we explore recent advancements in text representation, which not only capture both semantic and contextual similarities but also generate task-aware embeddings. We employ the INSTRUCTOR model, which uses task-aware prompts to generate text embeddings and, at the time of our experiments, achieved state-of-the-art performance on the MTEB text embedding benchmark [54], a comprehensive evaluation benchmark for measuring the performance of text embedding models on diverse embedding tasks.

For clustering algorithms, we have narrowed our focus to the most commonly used methods, specifically K-MEANS and AHC. These algorithms belong to different families of clustering techniques, providing a broader coverage for our analysis.

By combining these two algorithms with the five text representation methods, we create ten distinct traditional NLP clustering approaches for our experiments.

3.2 LLM-Based Text Clustering Methods

The LLM-based clustering methods discussed in this section leverage the natural language generation capabilities of LLMs, as outlined in Section 2.2.

3.2.1 Direct LLM-Based Text Clustering Method

LLMs have significantly expanded the range of tasks that can be addressed using natural language, thanks to their general-purpose language understanding and generation capabilities [5]. By accessing LLMs through prompting interfaces (e.g., GPT-4 API), users can format their tasks as natural language instructions that the model can follow, even for complex tasks [55]. One innovative approach to LLM-powered clustering is to directly instruct the LLM to group similar documents, bypassing traditional clustering algorithms entirely.

This direct method eliminates the need for vectorized document representations and instead leverages the LLM's ability to understand and generate human-readable

text. The LLM can generate both the clusters and their labels in a human-interpretable manner, offering explanations for why certain documents are grouped together. This capacity for providing meaningful, human-readable labels adds an extra layer of interpretability that traditional algorithms often lack.

Framework

The direct LLM-based clustering framework can be summarized in the following key steps:

1. **Define Task Objectives**: Clearly articulate the clustering task, specifying the goal of grouping similar documents. This sets a clear direction for the LLM.

2. **Prompt Engineering**: Construct effective prompts that guide the LLM in performing the clustering task. General design principles proposed by several existing papers [56, 57, 5] and websites [58, 59] include:

    - Clear task objectives: Define the task goal concisely.
    - Decomposition of task: Break the task into detailed, manageable subtasks
    - Few-shot demonstrations: Provide LLMs example outputs to improve accuracy.
    - Model-friendly format: Ensure prompts are structured in a way that aligns with LLMs preferences.
    - Role-playing strategies: Use techniques that prompt the LLM to assume a role, improving its response consistency.

3. **Execute Clustering**: Using the constructed prompts, instruct the LLM to analyse the documents and perform the clustering. The LLM processes the input text and assigns cluster labels to each document based on its understanding. The resulting set of label assignments forms the clusters.

4. **Review and Interpret Results**: The LLM produces both the clusters and their corresponding human-readable labels. This interpretability is a significant advantage of using LLM for clustering.

Challenges

While the direct clustering approach offers a novel and intuitive method for grouping documents, it presents unique challenges that require careful prompt design and an understanding of the model's inherent limitations:

- **Few-shot Demonstrations**: Adapting prompt examples can be challenging, as there may not be training material from the same distribution as the input text to provide for few-shot demonstrations.
- **Label Consistency**: The contextual understanding of LLMs may introduce variability in grouping due to the use of synonyms or different phrasing structures for the same cluster labels. This inconsistency can complicate the clustering process and hinder scalability, as the model might generate a different number of clusters than expected or use labels that, while semantically similar, are not explicitly aligned.
- **Cluster Number Constraints**: Clustering tasks often require a predefined number of clusters; however, while LLMs can follow semantic patterns, they are not inherently designed to optimize outcomes based on fixed constraints, such as generating a specified number of clusters labels [60]. Their probabilistic nature [6] allows for flexibility in response generation, making it challenging to consistently produce the same number of clusters.

In summary, while direct LLM-powered clustering offers a novel and intuitive approach to grouping documents, it presents unique challenges that require careful prompt design and an understanding of the model's inherent limitations.

Method Overview

Our proposed direct LLM-based clustering method approaches these issues through a two-step process.

Label Generation Step In the first step, we set aside the requirement to group texts to a predefined number of clusters and focus on generating cluster labels for batches of text. The prompt engineering in this stage involves providing the LLM with batches of data and requesting a cluster label for each instance, reflecting the aspect upon

Table 3.1: Example prompts for each step in the direct LLM-based text clustering method. The example uses a dataset of scientific publication titles, where clustering is based on the scientific category of each title. The colors highlight key design principles (see Section 3.2.1 for details). Blue text indicates the task description, green text represents the detailed breakdown into subtasks, and yellow text denotes the prompt format style.

Label Generation Step - First Batch:
I am clustering scientific publication titles based on their scientific category. To assist with this task, assign each title in a given batch of scientific publication titles to a cluster label that reflects its scientific category.
Generate a JSON dictionary where the keys are the provided IDs of the titles and the values are the corresponding cluster labels.

Label Generation Step - Subsequent Batches:
I am clustering scientific publication titles based on their scientific category. To assist with this task:
1. Assign each title in a given batch of scientific publication titles to a cluster label that reflects its scientific category. 2. Use the existing cluster labels <list of labels generated from all previous batches, e.g., [Physics, Astrophysics, Mathematics, Finance, Economics]> if they match the title. 3. Generate a new label based on the title's category if no existing label matches.
4. Generate a JSON dictionary where the keys are the provided IDs of the titles and the values are the corresponding cluster labels.

Clustering Through Label Consolidation Step:
You are given a dictionary of scientific categories, where the keys are the category IDs and the values are the category names. The objective is to consolidate these categories into exactly 5 broader but still specific scientific fields.
Your task is to analyze the provided categories, determine how they can be grouped together, and assign each original category to one of the final consolidated categories.
The output should be a JSON dictionary where the keys are the original category IDs, and the values are the new, consolidated categories.

which the text should be grouped. For example, in a dataset of scientific publication titles, the clustering may be based on the scientific category each title pertains to.

We randomly sample the batches and perform prompting with zero-shot demonstrations. The first batch establishes an initial set of labels, while subsequent batches refer to the pool of labels generated from all previous batches. Each new batch matches its texts to one of these existing labels if appropriate or assigns a new label if none fit. This iterative matching strategy helps maintain label coherence across batches and aims to reduce the issue of generating labels that exhibit synonyms or slightly different phrasing for similar concepts.

Clustering Through Label Consolidation Step While the first step aids in maintaining label consistency across the generated labels, it may also result in the model producing more unique labels (and thus more clusters) than desired. To address this, the second step introduces a constraint to consolidate the generated labels into a specified number of broader human-readable groupings, merging similar labels based on their thematic content or meaning. In this step, we provide the LLM with the set of labels generated from the previous step and instruct it to merge these labels into broader groupings until the required number of clusters is achieved. This merging process assigns a cluster label to each document in the input datasets, thus completing the clustering task.

It is essential to acknowledge, however, that while this step encourages label merging, it cannot fully overcome the challenge of adhering to a predefined number of clusters. Despite stating the desired number of clusters in the prompt, LLMs may not consistently achieve this due to their reliance on probabilistic text generation (see Section 3.2.1 for more details).

Working Example An example of prompts for the proposed method across all steps is presented in Table 3.1. This example uses a dataset of scientific publication titles, where clustering is based on the scientific category of each title.

3.2.2 LLM-Based Pre-Clustering Methods for Text Clustering

The pre-clustering LLM-based methods use the text generation capabilities of LLMs to extract task-specific information from text, enhancing the clustering process when paired with traditional clustering methods (see Subsection 2.2.1). Specifically, the

input text is enriched with task-relevant details, which is then transformed into a numerical representation and fed into the clustering algorithms.

Baseline

As our baseline, we follow the pre-clustering approach proposed by [52], where LLMs are employed to generate a comprehensive list of key phrases. However, in our experiments, requesting comprehensive lists of key phrases resulted in issues such as the LLMs reaching token length limits, generating repetitive outputs, and becoming computationally intensive. To overcome these, we adapted the prompt to instruct the LLM to generate a concise set of keyphrases, with nine demonstrations provided. An example of the prompt for the baseline method is shown in Table 3.2. As in the direct method (Subsection 3.2.1), this example uses a dataset of scientific publication titles, where clustering is based on each title's scientific category.

Table 3.2: Example prompt for the baseline pre-clustering LLM-based text clustering method. The example uses a dataset of scientific publication titles, where clustering is based on the scientific category of each title. The colors highlight key design principles (see Section 3.2.1 for details). Blue text indicates the task description, the yellow text denotes the prompt format style, and the pink text denotes the demonstrations.

I am clustering scientific publication titles based on their scientific category. To assist with this task, I need to identify the key phrases that capture the scientific category of each title. For a given scientific publication title, provide a list of key phrases that summarize its scientific category.
Generate the set of keyphrases as a python list.
Title: "'Neutrino Velocity and the Variability of Fundamental Constants"' Key phrases: ["Neutrino velocity", "Variability of fundamental constants", "Particle physics", "Fundamental physics", "Cosmology"] <More demonstrations>

After extracting the key phrases, we encoded them using the INSTRUCTOR model, which generates task-aware embeddings (see Subsection 2.1.1) and had state-of-the-art performance on the MTEB benchmark [54]. We then tested two approaches:

- Concatenation of Embeddings: Concatenating keyphrase embeddings with the

original document embeddings (both encoded using the INSTRUCTOR model).

- Keyphrase-Only Embeddings: Using only the keyphrase embeddings as input to the clustering algorithms.

We used the K-MEANS and AHC algorithms, resulting in four distinct baseline pre-clustering methods.

Our Method

The pre-clustering method we propose involves extracting cluster labels that represent the aspect on which the text should be grouped. This step mirrors the first phase of our direct LLM-based clustering method (see Subsection 3.2.1). Instead of proceeding with the direct clustering method, we encode these labels and concatenate the label embeddings with the original document embeddings (both encoded with the INSTRUCTOR model). This new representation serves as the input for K-MEANS and AHC algorithms. A working example of the prompt for this approach is presented in Table 3.1, being the Label Generation step of the direct clustering method.

Chapter 4

Empirical Analysis

In this chapter, we evaluate the performance of various clustering methods across 11 datasets, which are detailed in Section 4.1. The evaluation is conducted using two metrics, outlined in Section 4.2. We aim to assess the effectiveness of the clustering approaches and draw insights based on empirical results. The experimental setup, including the details of the models and configurations used in the experiments, is presented in Section 4.3. The results of these experiments are discussed in Section 4.4, where we highlight key observations and implications of our findings.

4.1 Datasets

We selected 11 short text datasets to capture a range of text clustering challenges. Table 4.1 summarizes the key characteristics of these datasets. Each dataset includes ground truth labels, which serve as benchmarks for evaluating the clustering performance. To manage the computational cost associated with using LLMs, we performed

stratified sampling for each dataset, selecting 100 instances evenly distributed across five distinct categories according to the ground truth labels.

MTEB Clustering Datasets [54] The text embeddings evaluation framework MTEB spans eight embedding tasks covering a total of 58 datasets and 112 languages. From this framework we selected the English short text datasets that pertain to the clustering task. These datasets are:

- ArxivClusteringS2S (`Arxiv`), BiorxivClusteringS2S (`Biorxiv`), MedrxivClusteringS2S (`Medrxiv`) [54]: These datasets derived from arXiv[1] and bioRxiv/medRxiv.[2] The input text is the title of the paper. The cluster labels were generated using categories given to the papers by the authors.
- StackExchangeClustering (`StackEx`), RedditClustering (`Reddit`) [61]: `StackEx` consists of question titles from multiple StackExchange forums,[3] while `Reddit` consists of titles from different subreddit communities on the Reddit platform.[4] Both are clustered based on the topic of discussion.

`Clinc` [62] This dataset consists of user queries categorized by their intent in a task-oriented dialogue system.

`Bank` [63] This dataset comprises user queries related to different intent categories in online banking.

Food Recall Incidents [64] This dataset consists of titles of food recall announcements from public food safety authorities. Most of the texts have been authored after 2010 and they describe recalls of specific food products due to specific hazards. Experts manually classified each text to four distinct levels based on the products involved and the hazards identified:

- Hazard: Specific hazards mentioned in the announcements,
- Hazard Category (`Hazard_Cat`): Broader categories of hazards,

[1] `https://info.arxiv.org/help/api/`
[2] `https://api.biorxiv.org/`
[3] `https://archive.org/download/stackexchange`
[4] `https://www.reddit.com/subreddits/`

- Product: Fine-grained description of the products mentioned in the announcements,
- Product Category (`Product_Cat`): Broader categories of products.

Each classification level forms a separate dataset, with different instances for each dataset.

Table 4.1: Dataset statistics, including vocabulary size, average number of characters and tokens per text, and the criterion used for grouping. All datasets are balanced, with 100 instances distributed across five ground truth clusters.

Dataset	Vocab	Avg. Chars	Avg. Toks	Grouping Criterion
`Medrxiv`	915	115.71	15.18	scientific category
`Biorxiv`	933	109.48	14.12	scientific category
`Arxiv`	658	71.40	9.52	scientific category
`Reddit`	846	68.95	11.54	topic
`StackEx`	623	58.74	10.14	topic
`Clinc`	239	41.19	8.72	intent
`Bank`	270	44.06	9.08	intent
`Product`	478	91.30	14.31	product
`Product_Cat`	600	75.62	10.76	product category
`Hazard`	486	83.37	12.22	hazard
`Hazard_Cat`	746	88.78	13.45	hazard category

The datasets can be categorized into three primary groups based on the nature of the aspect on which the text should be grouped:

- Intent-based: This group includes datasets where the objective is to classify texts according to user intent. Examples include `Clinc` and `Bank`, both of which consist of user queries submitted to conversational agents.
- Topic-based: These datasets focus on grouping texts by their subject matter. For instance, `StackEx` contains question titles from various StackExchange forums, while `Reddit` comprises titles from multiple subreddit communities.
- Domain-specific categories: This group includes datasets where the texts are clustered based on structured domain-related categories, such as scientific fields or specific products and hazards. Datasets such as `Medrxiv`, `Biorxiv`, `Arxiv` are

based on paper titles and their assigned scientific categories. Similarly, the Food Recall Incidents dataset focuses on product and hazard categories derived from food safety announcements.

4.2 Evaluation Metrics

To assess the quality of the clusters generated by different text clustering methods, we employ extrinsic metrics, which compare the predicted clusters to the gold standard labels provided by the datasets [65]. In line with previous studies [25, 52, 51], we use ACC and NMI as our primary evaluation metrics.

ACC measures the alignment between predicted clusters and ground truth labels. It is computed by finding the optimal alignment between the predicted and true labels using the Hungarian algorithm [66]. Given a text x_i, let c_i be the predicted cluster label and t_i the ground truth label. ACC is then defined as:

$$\text{ACC} = \frac{\sum_{i=1}^{n} \delta(t_i, \text{map}(c_i))}{n}, \tag{4.1}$$

where:

- n is the total number of texts,
- $\delta(x, y)$ is the indicator function that equals 1 if $x = y$ and 0 otherwise,
- map(c_i) is the permutation mapping function that maps each cluster label c_i to the corresponding ground truth label from the dataset by the Hungarian algorithm.

The NMI metric calculates the mutual information between the true and predicted label sets, normalized by their individual entropies. It evaluates how much information is shared between the predicted clusters and the ground truth labels.

$$\text{NMI}(T, C) = \frac{2\, I(T; C)}{H(T) + H(C)}$$

where:

- T represents the ground truth labels,
- C represents the predicted cluster labels,,

- $I(T;C)$ is the mutual information between the two sets of labels,
- $H(T)$ and $H(C)$ are the are the entropy of the true labels and predicted clusters, respectively.

While both ACC and NMI are widely used to evaluate clustering quality, they assume that the number of clusters in the prediction matches the number of clusters in the ground truth. ACC, relying on an optimal one-to-one mapping between predicted clusters and true labels, presumes that the number of clusters is identical. When the number of predicted clusters deviates from the ground truth, the Hungarian algorithm used for mapping faces difficulties, penalizing methods that generate a different number of clusters. This can lead to lower accuracy scores when the clustering solution over- or under-clusters the data relative to the ground truth [67].

While ACC penalizes both over- and under-clustering similarly, NMI is less sensitive to over-clustering. NMI evaluates the shared information between the predicted and true clusters without being heavily affected by the number of clusters. The mutual information term in the numerator does not significantly penalize methods that produce excessive clusters (over-clustering) because it measures the overall shared information, not the cluster cardinalities. Although the denominator includes entropy terms that account for the individual cluster distributions, it does not completely mitigate the effect of over-clustering. Therefore, NMI remains less sensitive to over-clustering than accuracy [67]. For instance, if a method over-clusters but assigns many correct labels to the right clusters, the mutual information might still be high, indicating that the predicted clusters capture much of the true distribution, despite the excessive number of clusters.

4.3 Experimental Setup

For the static word embeddings, we use pre-trained embeddings from the Gensim data repository,[5] specifically from three widely recognized models: Word2Vec [10], FastText [11], and GloVe [12]. For the Word2Vec model, we use the "word2vec-google-news-300" embeddings, trained on Google News. The FastText model we use is "fasttext-wiki-news-subwords-300", which is trained on Wikipedia 2017,[6] UMBC

[5] https://github.com/piskvorky/gensim-data

[6] https://dumps.wikimedia.org/enwiki/latest/

webbase corpus,[7] and the statmt.org news dataset.[8] For the GloVe model, we use "glove-wiki-gigaword-300", trained on the 2014 Wikipedia[9] and the Gigaword 5 dataset.[10] All three models generate 300-dimensional embeddings.

For contextual text embeddings, we use the "hkunlp/instructor-large" model,[11] a transformer-based model that provides rich contextual embeddings by using instructions for the embedding process. This model is particularly suitable for generating task-specific representations (see Subsection 2.1.1 for more details).

In our experiments, we performed clustering using the ground truth number of clusters for each dataset. Regarding the clustering algorithms, for K-MEANS we use the KMeans++ [68] algorithm which initializes cluster centres that are distant from each other to improve the convergence of K-MEANS. We use this method with standard hyperparameters, including the ground truth number of clusters. For AHC, we performed extensive hyperparameter tuning (the results are presented in Appendix A) to select the best linkage criterion and distance metric. For the main experiments, we use the Ward linkage with Euclidean distance, as these configurations consistently outperformed others during tuning, producing compact and well-separated clusters.

Finally, we use OpenAI's GPT-3.5 Turbo[12] for text generation tasks related to clustering. Specifically, GPT-3.5 Turbo is used to generate cluster labels and key phrases for the pre-clustering approaches (see Subsection 3.2.2). For each approach, we created prompt templates for all datasets, as depicted in Section 3.2, to ensure consistency across datasets and experiments.

4.4 Experimental Results and Discussion

Table 4.2 presents the empirical results for text clustering using the ACC and NMI metrics, respectively. We make the following observations:

[7] http://ebiquity.umbc.edu/resource/html/id/351

[8] https://data.statmt.org/news-crawl/

[9] https://dumps.wikimedia.org/enwiki/latest/

[10] https://doi.org/10.35111/wk4f-qt80

[11] https://huggingface.co/hkunlp/instructor-large

[12] https://platform.openai.com/docs/models/gpt-3-5-turbo

Table 4.2: Text clustering results based on the ACC and NMI metrics for each dataset. The first three columns represent the combination of input text, text representation, and clustering algorithm used in each method. The last two columns show the average ACC and NMI across all 11 datasets. Best results are highlighted in bold. The method marked with a † indicates that the number of clusters (shown in parentheses for each dataset) differs from the ground truth number of clusters, which is five for all datasets (see Section 4.1 for more details).

Input	Text Repr.	Cl. Algorithm	Arxiv		Biorxiv		Medrxiv		StackEx		Reddit		Clinc		Bank		Hazard		Hazard_Cat		Product		Product_Cat		Avg.	
			ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
Traditional Clustering Methods (Baselines)																										
text	TF-IDF	AHC	0.36	0.11	0.36	0.12	0.40	0.20	0.33	0.12	0.32	0.14	0.68	0.80	0.78	0.66	0.60	0.55	0.46	0.23	0.59	0.61	0.37	0.21	0.477	0.340
	TF-IDF	K-means	0.35	0.13	0.29	0.06	0.37	0.13	0.31	0.07	0.32	0.07	0.67	0.74	0.69	0.55	0.52	0.39	0.41	0.18	0.63	0.63	0.34	0.17	0.445	0.285
	Word2Vec	AHC	0.33	0.10	0.34	0.11	0.32	0.13	0.30	0.16	0.45	0.29	0.97	0.93	0.65	0.54	0.34	0.23	0.35	0.18	0.30	0.15	0.32	0.14	0.425	0.270
	Word2Vec	K-means	0.43	0.24	0.28	0.05	0.29	0.08	0.49	0.32	0.29	0.17	0.75	0.77	0.56	0.39	0.39	0.28	0.44	0.20	0.45	0.29	0.36	0.15	0.430	0.268
	FastText	AHC	0.37	0.16	0.29	0.07	0.28	0.04	0.32	0.12	0.38	0.15	0.62	0.44	0.28	0.07	0.29	0.09	0.32	0.10	0.28	0.11	0.31	0.11	0.340	0.131
	FastText	K-means	0.35	0.12	0.27	0.04	0.28	0.05	0.32	0.06	0.40	0.19	0.67	0.51	0.40	0.18	0.30	0.10	0.30	0.10	0.32	0.10	0.33	0.11	0.358	0.143
	GloVe	AHC	0.29	0.11	0.33	0.08	0.27	0.07	0.31	0.10	0.30	0.11	0.89	0.82	0.51	0.41	0.29	0.10	0.42	0.17	0.31	0.11	0.37	0.16	0.390	0.203
	GloVe	K-means	0.30	0.12	0.30	0.09	0.29	0.06	0.31	0.13	0.30	0.08	0.64	0.61	0.54	0.44	0.33	0.16	0.33	0.09	0.34	0.15	0.32	0.15	0.364	0.190
	Instructor	AHC	0.66	0.54	0.66	0.48	0.70	0.50	0.85	0.74	0.80	0.70	0.99	0.98	0.84	0.84	0.58	0.51	0.52	0.33	0.77	0.77	0.53	0.32	0.718	0.609
	Instructor	K-means	0.64	0.46	0.61	0.44	0.58	0.42	0.84	0.72	0.81	0.71	0.71	0.87	0.92	0.89	0.58	0.49	**0.54**	0.35	0.78	0.78	0.59	0.43	0.691	0.595
LLM-Based Clustering Methods (Baselines and Ours)																										
text + key phrases	Instructor	AHC *(Baseline)*	0.64	0.57	0.57	0.38	0.55	0.45	0.87	0.76	**0.86**	**0.80**	**1.00**	**1.00**	0.86	0.89	0.63	0.52	0.44	0.31	**0.96**	**0.91**	0.81	0.66	0.745	0.658
text + key phrases	Instructor	K-means *(Baseline)*	0.62	0.55	0.50	0.38	0.66	0.50	0.88	0.77	0.76	0.71	0.73	0.87	0.69	0.80	0.64	0.56	0.41	0.20	0.75	0.83	0.71	0.57	0.668	0.611
key phrases	Instructor	AHC *(Baseline)*	0.80	0.63	0.67	0.57	0.76	0.61	0.82	0.71	0.79	0.71	0.97	0.95	0.82	0.79	0.52	0.30	0.43	0.19	0.88	0.80	0.58	0.56	0.731	0.620
key phrases	Instructor	K-means *(Baseline)*	0.68	0.65	0.64	0.49	0.69	0.56	0.86	0.73	0.85	0.73	**1.00**	**1.00**	0.67	0.77	0.51	0.48	0.44	0.22	0.82	0.82	**0.82**	0.68	0.725	0.636
text + label	Instructor	AHC *(Ours)*	**0.82**	**0.67**	0.65	0.48	**0.82**	0.68	**0.93**	**0.86**	0.84	0.76	**1.00**	**1.00**	**0.95**	**0.91**	0.68	0.60	0.51	0.35	**0.96**	**0.91**	0.81	**0.71**	**0.815**	**0.721**
text + label	Instructor	K-means *(Ours)*	0.67	0.56	0.52	0.36	0.52	0.41	0.71	0.82	0.84	0.76	0.99	0.98	**0.95**	0.90	**0.70**	0.56	0.44	0.32	0.71	0.82	0.75	0.58	0.709	0.643
text		Direct *(Ours)*†	0.54	0.58 (6)	**0.68**	**0.60** (9)	0.80	**0.76** (8)	0.79	0.80 (6)	0.74	0.74 (6)	0.96	0.95 (7)	0.68	0.83 (5)	0.65	**0.61** (6)	0.43	**0.36** (8)	0.92	0.83 (5)	0.66	0.66 (9)	0.714	0.702

Pre-clustering Approach Our proposed pre-clustering approach (see Section 3.2.2), where the input text is the original text along with LLM-generated labels, shows superior performance in both average ACC and NMI. While it does not achieve state-of-the-art results for every dataset (falling short in five out of 11 datasets for both metrics), it performs very close to the highest results in those cases. For the remaining six datasets, it significantly outperforms the second-best method, achieving an average ACC that is 9% higher and an average NMI that is 3% higher than the second-best method.

Direct LLM-based Clustering Our direct LLM-based clustering method ranks second when evaluated with the NMI metric, which is less sensitive to over-clustering (achieving state-of-the-art results in four datasets). However, it drops to sixth place in terms of the average ACC score. This suggests that when the number of clusters is not a critical factor, the direct method is a strong option for clustering. Additionally, the interpretability of the cluster labels offers a significant advantage. Table 4.3 presents the ground truth labels and the predicted labels from our proposed Direct LLM-based clustering method for each dataset. By comparing the predicted labels with the ground truth labels, we observe that in many cases, the labels align well (e.g., in the `Reddit` dataset). However, in other cases, there is noticeable divergence. In some cases, the LLM overpredicts by generating more general labels. For instance, in the `Arxiv` dataset, the LLM does not distinguish between closely related scientific fields. In other cases, the LLM overextends to labels that do not exist in the true labels, as seen in the `Biorxiv` dataset.

One possible reason for the mismatches could be that, in the first step of the method, the LLM generates cluster labels that are more fine-grained (see Subsection 3.2.1 for details). In the second step, where the number of clusters is constrained, these fine-grained labels are merged into broader ones, resulting in more general labels. Moreover, labels that do not exist in the ground truth remain unaligned.

While clustering is an unsupervised task and the ground truth is not always available, the human-readable labels generated by the LLM provide a valuable level of interpretability. This ability to assign meaningful labels to clusters is a key advantage of this method, particularly compared to traditional clustering algorithms that typically lack direct label assignment. Instead, those methods often require additional steps such as word clouds, topic modelling, or summarization to achieve interpretabil-

Table 4.3: Comparison of true labels and predicted labels across different datasets.

Dataset	True Labels	Predicted Labels	NMI
Arxiv	Astrophysics, High Energy Physics, Mathematics, Quantitative Finance, Quantum Physics	Biology, Earth Science, Economics, Engineering, Mathematics, Physics	0.58
Biorxiv	Cancer Biology, Cell Biology, Genomics, Microbiology, Neuroscience	Cell Biology, Genetics, Medicine, Microbiology, Neuroscience, Oncology, Plant Biology, Toxicology, Virology	**0.60**
Medrxiv	Cardiovascular Medicine, Infectious Diseases, Neurology, Oncology, Pediatrics	Cardiology, Immunology, Infectious Diseases, Neurology, Neuroscience, Oncology, Pediatrics, Pharmacology	**0.76**
StackEx	Gaming, Graphic Design, Philosophy, Photo, User Experience Design	Art and Design, Design and Creativity, Lifestyle, Science and Academia, Technology, Video Games	0.80
Reddit	Animals, Architecture, Atheism, Beer, Classical Music	Animals, Architecture, Food and Drink, Music, Religion, Unknown	0.74
Clinc	change_ai_name, distance, food_last, oil_change_when, where_are_you_from	Food Inquiry, General Inquiry, Government Inquiry, Location Inquiry, Name Change Request, Travel Inquiry, Vehicle Inquiry	0.95
Bank	card_delivery_estimate, card_not_working, declined_cash_withdrawal, unable_to_verify_identity, why_verify_identity	ATM Issues, Account Verification, Card Delivery, Card Issue, Transaction Issue	0.83
Hazard	Foreign Bodies, Lead, Nuts, Sesame Seeds and Products Thereof, Unauthorised Use of Federal Inspection Mark	Allergen Contamination, Contaminant Contamination, Foreign Matter Contamination, Health Risk, Inspection Violation, Mislabeling	**0.61**
Hazard_Cat	Biological, Chemical, Food Additives and Flavourings, Foreign Bodies, Fraud	Allergen Contamination, Chemical Contamination, Ingredient Issue, Labelling Deficiency, Microbiological Contamination, Other Contamination, Physical Contamination, Uninspected Import	**0.36**
Product	Baby Food, Chocolate Bars, Cookies, Ground Beef, Onions	Baby Food, Bakery Products, Chocolate Bars, Meat and Poultry Products, Vegetables	0.83
Product_Cat	Cocoa and Cocoa Preparations, Coffee and Tea, Food Additives and Flavourings, Food Contact Materials, Nuts and Nut Products and Seeds, Poultry Meat and Poultry Meat Products	Cheese Dips, Chocolate Products, Confectionery, Food Additive, Food Coloring Products, Kitchen Tools, Nuts, Other Supplements, Poultry Products	0.66

ity, making the LLM-generated labels more efficient and intuitive for understanding the clustering results.

Effect of Pre-clustering Methods Pre-clustering methods (see Subsection 3.2.2), which add supplementary information to the clustering process, consistently outperform methods that rely solely on the original text. While our pre-clustering method achieves state-of-the-art results, methods that provide LLM-generated key phrases as input to clustering algorithms also perform well. In particular, the method inspired by [52] (see Section 3.2.2), combined with AHC, ranks second in terms of ACC. This highlights the value of key phrases, as even the method using only generated key phrases as input ranks third in ACC performance.

Comparison of Text Representation Approaches Clustering methods that use earlier text representation techniques, such as TF-IDF and static word embeddings, perform worse across all datasets compared to those using contextual text representations such as Instructor embeddings. This confirms the superiority of contextual embeddings over traditional approaches. However, TF-IDF representations outperform static word embeddings in specialized datasets (e.g., `Biorxiv`, `Medrxiv`, `Bank`, `Hazard`, `Hazard_Cat`, `Product`, and `Product_Cat`), likely because the pre-trained embeddings were derived from different domain-specific datasets than the ones used here.

Comparison of Clustering Algorithms Clustering methods using AHC generally achieve higher ACC and NMI scores than those using K-MEANS, with some exceptions.[13] To validate this trend, we evaluated clustering quality with an intrinsic measure, the silhouette score [69]. The silhouette score assesses the cohesion and separation of clusters by measuring how similar an instance is to its own cluster compared to the nearest neighbouring cluster. Scores range from -1 to 1, where higher values indicate well-separated, cohesive clusters.

For a text instance t_i, the silhouette score $s(t_i)$ is calculated as:

$$s(t_i) = \frac{b(t_i) - a(t_i)}{max(a(t_i), b(t_i))} \tag{4.2}$$

where $a(t_i)$ is the mean intra-cluster distance, and $b(t_i)$ is the minimum mean distance to any other cluster.

[13]Specifically, methods using Word2Vec and FastText representations, as well as the pre-clustering approach with key phrases, exhibit different trends as shown in Table 4.2.

As shown in Table 4.4, the silhouette scores averaged across all 11 datasets reveal that AHC consistently outperforms K-MEANS, confirming the ACC and NMI trends. This indicates that AHC produces more compact, well-separated clusters, effectively capturing the underlying structure of the data.

Table 4.4: Average silhouette scores across all 11 datasets for the same clustering methods, with variations in the clustering algorithm (AHC and K-MEANS).

Input	Text Representation	Avg. Silhouette	
		AHC	K-MEANS
text	TF-IDF	**0.058**	0.033
	Word2Vec	**0.106**	0.088
	FastText	**0.149**	0.137
	GloVe	**0.145**	0.144
	Instructor	**0.122**	0.104
text + key phrases	Instructor	**0.130**	0.111
key phrases	Instructor	**0.219**	0.215
text + label	Instructor	**0.148**	0.136

Chapter 5

Epilogue

5.1 Conclusions

In conclusion, this study highlights the crucial role of LLMs in improving the text clustering task, both through pre-clustering interventions to the task with traditional clustering methods and through direct clustering. Our pre-clustering approach, which leverages LLM-generated labels, achieved superior results on average across evaluation metrics. This method significantly enhances clustering performance and interpretability. Another important contribution is the exploration of LLMs for direct text clustering. Although this approach does not always generate the exact required number of clusters, it is the second-best method on average in terms of NMI. Furthermore, the human-readable labels generated by the LLM add substantial interpretative value, particularly in domains where clarity and understanding are essential. Overall, our experimental findings highlight the dual advantages of LLMs in text clustering: improved performance and enhanced interpretability.

5.2 Future Work

Several directions can be explored in future work to further enhance the use of LLMs in text clustering.

Scalability The scalability of the proposed methods can be addressed more rigorously. While the current study explores clustering on small-sized datasets, testing LLM-based clustering on larger datasets could provide insights into the practical applications and limitations in real-world scenarios. This would also help identify the computational efficiency of LLMs in large-scale clustering tasks.

Smaller-Sized or Open-Source LLMs In this study, we experimented with the closed-source GPT-3.5 turbo. Exploring how to effectively use smaller-sized or open-source LLMs for clustering could address the cost limitation and make LLM-based clustering more accessible.

Few-shot Demonstrations Our proposed methods incorporated LLMs in zero-shot prompting. Providing few-shot demonstrations to LLMs instructions could potentially leverage in-context learning.

Optimizing Label Propagation in Direct LLM-Based Clustering In our proposed direct LLM-based clustering method, cluster labels are generated iteratively in randomly sampled batches. The first batch establishes an initial set of labels, and subsequent batches refer to the pool of labels generated from all previous batches: each new batch matches its texts to one of these existing labels if appropriate, or assigns a new label if none fit. Exploring different batch ordering strategies may improve label consistency and clustering performance by optimizing how initial labels are applied across batches.

Direct LLM-Based Clustering and Predefined Cluster Numbers A key limitation in the current direct LLM-based clustering method is its inability to generate a predefined number of clusters. Addressing this challenge can open several directions for future work. One potential approach involves incorporating hierarchical clustering methods in combination with LLMs to allow for merging or splitting clusters at different levels until the target number is reached. Additionally, improvements in

prompt engineering could help explicitly instruct LLMs to align with fixed clustering goals, with few-shot learning that could train the model to generalize optimal behaviours across cluster numbers. Post-clustering approaches, such as cluster merging or splitting, could also ensure that the final number of clusters aligns with the predefined requirements, refining results without losing the LLMs' semantic insights.

Bibliography

[1] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[2] A. Subakti, H. Murfi, and N. Hariadi, "The performance of bert as data representation of text clustering," *Journal of big Data*, vol. 9, no. 1, pp. 1–21, 2022.

[3] J. Chang, S. Gerrish, C. Wang, J. Boyd-Graber, and D. Blei, "Reading tea leaves: How humans interpret topic models," *Advances in neural information processing systems*, vol. 22, 2009.

[4] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[5] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[6] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large language models: A survey," *arXiv preprint arXiv:2402.06196*, 2024.

[7] S. E. Robertson and K. S. Jones, "Relevance weighting of search terms," *Journal of the American Society for Information science*, vol. 27, no. 3, pp. 129–146, 1976.

[8] L. H. Patil and M. Atique, "A novel approach for feature selection method tf-idf in document clustering," in *2013 3rd IEEE international advance computing conference (IACC)*. IEEE, 2013, pp. 858–862.

[9] P. Bafna, D. Pramod, and A. Vaidya, "Document clustering: Tf-idf approach," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE, 2016, pp. 61–66.

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

[11] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[12] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[15] H. Su, W. Shi, J. Kasai, Y. Wang, Y. Hu, M. Ostendorf, W.-t. Yih, N. A. Smith, L. Zettlemoyer, and T. Yu, "One embedder, any task: Instruction-finetuned text embeddings," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 1102–1121.

[16] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[17] A. Jain, "Algorithms for clustering data," 1988.

[18] H. Steinhaus, "Sur la division des corps matériels en parties," *Bull. Acad. Polon. Sci*, vol. 1, no. 804, p. 801, 1956.

[19] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[20] G. H. Ball and D. J. Hall, "Isodata, a novel method of data analysis and pattern classification," *stanford research institute*, pp. AD–699 616, 1965.

[21] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*, 1967.

[22] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine learning*, vol. 42, pp. 143–175, 2001.

[23] J. Xu, P. Wang, G. Tian, B. Xu, J. Zhao, F. Wang, and H. Hao, "Short text clustering via convolutional neural networks," in *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, 2015, pp. 62–69.

[24] A. Hadifar, L. Sterckx, T. Demeester, and C. Develder, "A self-training approach for short text clustering," in *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, I. Augenstein, S. Gella, S. Ruder, K. Kann, B. Can, J. Welbl, A. Conneau, X. Ren, and M. Rei, Eds. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 194–199. [Online]. Available: https://aclanthology.org/W19-4322

[25] D. Zhang, F. Nan, X. Wei, S.-W. Li, H. Zhu, K. McKeown, R. Nallapati, A. O. Arnold, and B. Xiang, "Supporting clustering with contrastive learning," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds. Online: Association for Computational Linguistics, Jun. 2021, pp. 5419–5430. [Online]. Available: https://aclanthology.org/2021.naacl-main.427

[26] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5147–5156.

[27] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 132–149.

[28] Y. Tao, K. Takagi, and K. Nakata, "Clustering-friendly representation learning via instance discrimination and feature decorrelation," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=e12NDM7wkEY

[29] M. A. Siegler, U. Jain, B. Raj, and R. M. Stern, "Automatic segmentation, classification and clustering of broadcast news audio," in *Proc. DARPA speech recognition workshop*, vol. 1997, 1997.

[30] M. Levy and M. Sandler, "Structural segmentation of musical audio by constrained clustering," *IEEE transactions on audio, speech, and language processing*, vol. 16, no. 2, pp. 318–326, 2008.

[31] H. Alwassel, D. Mahajan, B. Korbar, L. Torresani, B. Ghanem, and D. Tran, "Self-supervised learning by cross-modal audio-video clustering," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9758–9770, 2020.

[32] B. Peng, J. Lei, H. Fu, Y. Jia, Z. Zhang, and Y. Li, "Deep video action clustering via spatio-temporal feature learning," *Neurocomputing*, vol. 456, pp. 519–527, 2021.

[33] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *2014 22nd International conference on pattern recognition*. IEEE, 2014, pp. 1532–1537.

[34] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 257–266.

[35] Q. Zhang, J. Wu, P. Zhang, G. Long, and C. Zhang, "Salient subsequence learning for time series clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2193–2207, 2018.

[36] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clustering–a decade review," *Information systems*, vol. 53, pp. 16–38, 2015.

[37] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[38] X. Wang and J. Wang, "Using clustering methods in geospatial information systems," *Geomatica*, vol. 64, no. 3, pp. 347–361, 2010.

[39] S. Zhou, H. Xu, Z. Zheng, J. Chen, Z. Li, J. Bu, J. Wu, X. Wang, W. Zhu, and M. Ester, "A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions," *ACM Computing Surveys*, 2022.

[40] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *arXiv preprint arXiv:1611.02648*, 2016.

[41] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "Clustergan: Latent space clustering in generative adversarial networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4610–4617.

[42] A. Hadifar, L. Sterckx, T. Demeester, and C. Develder, "A self-training approach for short text clustering," in *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, 2019, pp. 194–199.

[43] M. Steinbach, "A comparison of document clustering techniques," Technical Report# 00_034/University of Minnesota, Tech. Rep., 2000.

[44] P. Wang, B. Xu, J. Xu, G. Tian, C.-L. Liu, and H. Hao, "Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification," *Neurocomputing*, vol. 174, pp. 806–814, 2016.

[45] Ö. Çoban and G. T. Özyer, "Word2vec and clustering based twitter sentiment analysis," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. IEEE, 2018, pp. 1–5.

[46] S. Huang, F. Wei, L. Cui, X. Zhang, and M. Zhou, "Unsupervised fine-tuning for text clustering," in *Proceedings of the 28th International Conference on*

Computational Linguistics, D. Scott, N. Bel, and C. Zong, Eds. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 5530–5534. [Online]. Available: https://aclanthology.org/2020.coling-main.482

[47] Y. Li, J. Cai, and J. Wang, "A text document clustering method based on weighted bert model," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1. IEEE, 2020, pp. 1426–1430.

[48] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *arXiv preprint arXiv:2212.03533*, 2022.

[49] H. Su, W. Shi, J. Kasai, Y. Wang, Y. Hu, M. Ostendorf, W.-t. Yih, N. A. Smith, L. Zettlemoyer, and T. Yu, "One embedder, any task: Instruction-finetuned text embeddings," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 1102–1121.

[50] Z. Wang, J. Shang, and R. Zhong, "Goal-driven explainable clustering via language descriptions," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 10 626–10 649.

[51] Y. Zhang, Z. Wang, and J. Shang, "Clusterllm: Large language models as a guide for text clustering," *arXiv preprint arXiv:2305.14871*, 2023.

[52] V. Viswanathan, K. Gashteovski, C. Lawrence, T. Wu, and G. Neubig, "Large language models enable few-shot clustering," *arXiv preprint arXiv:2307.00524*, 2023.

[53] M. De Raedt, F. Godin, T. Demeester, C. Develder, and S. Chatlayer, "Idas: Intent discovery with abstractive summarization," in *The 5th Workshop on NLP for Conversational AI*, 2023, p. 71.

[54] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "Mteb: Massive text embedding benchmark," *arXiv preprint arXiv:2210.07316*, 2022.

[55] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[56] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, 2023.

[57] S. K. K. Santu and D. Feng, "Teler: A general taxonomy of llm prompts for benchmarking complex tasks," *arXiv preprint arXiv:2305.11430*, 2023.

[58] OpenAI. (2023) Gpt best practices. [Online]. Available: https://platform.openai.com/docs/guides/prompt-engineering

[59] (2024) Awesome chatgpt prompts. [Online]. Available: https://github.com/f/awesome-chatgpt-prompts/

[60] M. X. Liu, F. Liu, A. J. Fiannaca, T. Koo, L. Dixon, M. Terry, and C. J. Cai, "" we need structured output": Towards user-centered constraints on large language model output," in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–9.

[61] G. Geigle, N. Reimers, A. Rücklé, and I. Gurevych, "Tweac: transformer with extendable qa agent classifiers," *arXiv preprint arXiv:2104.07081*, 2021.

[62] S. Larson, A. Mahendran, J. J. Peper, C. Clarke, A. Lee, P. Hill, J. K. Kummerfeld, K. Leach, M. A. Laurenzano, L. Tang *et al.*, "An evaluation dataset for intent classification and out-of-scope prediction," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 1311–1316.

[63] I. Casanueva, T. Temcinas, D. Gerz, M. Henderson, and I. Vulic, "Efficient intent detection with dual sentence encoders," *ACL 2020*, p. 38, 2020.

[64] K. Randl, J. Pavlopoulos, A. Henriksson, and T. Lindgren, "Cicle: Conformal in-context learning for largescale multi-class food risk classification," *arXiv preprint arXiv:2403.11904*, 2024.

[65] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Information retrieval*, vol. 12, pp. 461–486, 2009.

[66] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[67] M. Ronen, S. E. Finder, and O. Freifeld, "Deepdpm: Deep clustering with an unknown number of clusters," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9861–9870.

[68] D. Arthur, S. Vassilvitskii *et al.*, "k-means++: The advantages of careful seeding," in *Soda*, vol. 7, 2007, pp. 1027–1035.

[69] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

Appendix A

Hyperparameter Tuning for AHC

Table A.1 presents the average ACC and average NMI metrics across all 11 datasets from the hyperparameter tuning process for AHC. With the stopping criterion fixed to the number of ground truth labels, the two remaining hyperparameters in AHC are the linkage criterion and the distance metric (see Section 2.1.2 for details). The best clustering performance of AHC is achieved using the combination of the ward linkage criterion and the euclidean distance metric. These two hyperparameters were selected for AHC in the main experiments, as they were the most suitable for our datasets.

Table A.1: Results showing the average ACC and average NMI metrics across all 11 datasets from the hyperparameter tuning process for AHC. Best results are highlighted in bold.

Linkage Criterion	Distance Metric	Avg. ACC	Avg. NMI
single	euclidean	0.320	0.217
single	cosine	0.320	0.217
single	manhattan	0.320	0.217
single	l1	0.320	0.217
single	l2	0.320	0.217
complete	euclidean	0.593	0.515
complete	cosine	0.320	0.217
average	manhattan	0.447	0.401
complete	l1	0.320	0.217
complete	l2	0.320	0.217
average	euclidean	0.437	0.386
average	cosine	0.430	0.38
complete	manhattan	0.320	0.217
average	l1	0.447	0.401
average	l2	0.437	0.386
ward	euclidean	**0.718**	**0.609**

Short Biography

Georgia Bakagianni is a Researcher at Agroknow and her work is focused on NLP and machine learning. Previously, she served as a Senior Research Associate at ILSP/Athena RC, engaged in numerous European and national projects (META-SHARE, QTLaunchPad, CEF-ELRC, CRACKER, Clarin:EL, Apollonis, ELE, and LDS) related to language resource infrastructures. She completed her undergraduate studies at Athens University of Economics and Business (AUEB), earning degrees from both the Department of Informatics and the Department of Business Administration.