Large Language Models Enable Few-Shot Clustering

Vijay Viswanathan[1], Kiril Gashteovski[2,3], Carolin Lawrence[2], Tongshuang Wu[1], Graham Neubig[1]

[1]Carnegie Mellon University, USA, [2]NEC Laboratories Europe, Germany
[3]Center for Advanced Interdisciplinary Research, Ss. Cyril and Methodius Uni. of Skopje, Germany

Abstract

Unlike traditional unsupervised clustering, semi-supervised clustering allows users to provide meaningful structure to the data, which helps the clustering algorithm to match the user's intent. Existing approaches to semi-supervised clustering require a significant amount of feedback from an expert to improve the clusters. In this paper, we ask whether a large language model (LLM) can amplify an expert's guidance to enable query-efficient, few-shot semi-supervised text clustering. We show that LLMs are surprisingly effective at improving clustering. We explore three stages where LLMs can be incorporated into clustering: before clustering (improving input features), during clustering (by providing constraints to the clusterer), and after clustering (using LLMs post-correction). We find that incorporating LLMs in the first two stages routinely provides significant improvements in cluster quality, and that LLMs enable a user to make trade-offs between cost and accuracy to produce desired clusters. We release our code and LLM prompts for the public to use.[1]

1 Introduction

Unsupervised clustering aims to do an impossible task: Organize data in a way that satisfies a user's needs without any specification of what those needs are. Clustering, by its nature, is fundamentally an *underspecified* problem. According to Caruana (2013), this underspecification makes clustering ''probably approximately useless.''

Semi-supervised clustering, on the other hand, aims to solve this problem by enabling the domain expert to guide the clustering algorithm (Bae et al., 2020) by providing feedback. Prior work has introduced different types of interaction between an expert and a clustering algorithm, such as hand-picked seed points (Basu et al., 2002) or pairwise constraints between points (Basu et al., 2004). These interfaces have all been shown to give experts control of the final clusters. However, for large, real-world datasets with a large number of possible clusters, the feedback cost required by interactive clustering algorithms can be immense.

Building on a body of recent work that uses large language models (LLMs) as noisy simulations of human decision-making (Fu et al., 2023; Horton, 2023; Park et al., 2023), we propose a different approach for semi-supervised text clustering, illustrated in Figure 1. In particular, we answer the following research question: *Can an expert provide a few demonstrations of their desired interaction (e.g., pairwise constraints) to a large language model, then let the LLM direct the clustering algorithm?*

We explore three places in the text clustering process where an LLM could be leveraged: before clustering, during clustering, and after clustering. We leverage an LLM *before clustering* by augmenting the textual representation. For each example, we generate keyphrases with an LLM, encode these keyphrases, and add them to the base representation. We incorporate an LLM *during clustering* by adding cluster constraints. Adopting a classical algorithm for semi-supervised clustering, we use an LLM as a pairwise constraint pseudo-oracle. We finally explore using an LLM *after clustering* by correcting low-confidence cluster assignments using the pairwise constraint pseudo-oracle. In every case, the interaction between a user and the clustering algorithm is enabled by a prompt written by the user and provided to a large language model.

We test these three methods on five datasets across three tasks: canonicalizing entities, clustering queries by intent, and grouping tweets by topic. We find that, compared to traditional K-Means clustering on document embeddings, using an LLM to enrich each document's representation empirically improves cluster quality on every

[1]https://github.com/viswavi/few-shot-clustering.

Transactions of the Association for Computational Linguistics, vol. 12, pp. 321–333, 2024. https://doi.org/10.1162/tacl_a_00648
Action Editor: Noah Smith. Submission batch: 8/2023; Revision batch: 12/2023; Published 4/2024.

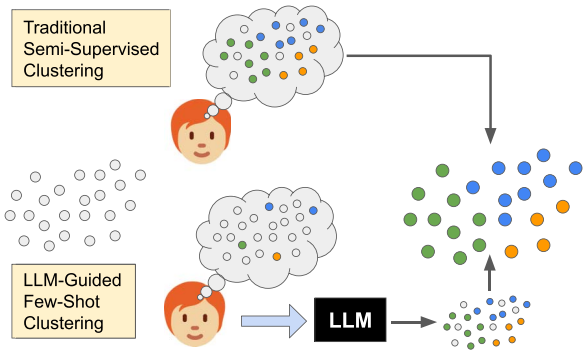

Figure 1: In traditional semi-supervised clustering, a user provides a large amount of feedback to the clusterer. In our approach, the user prompts an LLM with a small amount of feedback. The LLM then generates a large amount of pseudo-feedback for the clusterer.

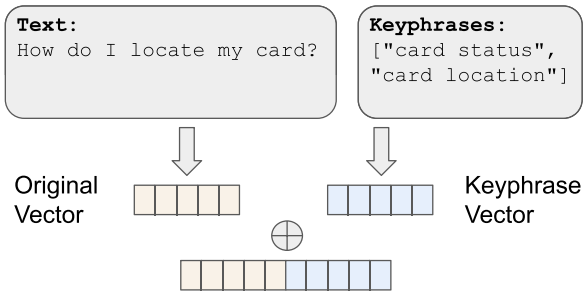

Figure 2: We expand document representations by concatenating them with keyphrase embeddings. The keyphrases are generated by a large language model.

metric for all datasets we consider. Using an LLM as a pairwise constraint pseudo-oracle can also be highly effective when the LLM is capable of providing pairwise similarity judgements but requires a larger number of LLM queries to be effective. However, LLM post-correction provides limited upside. Importantly, LLMs can also approach the performance of *traditional semi-supervised clustering with a human oracle* at a fraction of the cost.

Our work stands out from recent deep-learning-based text clustering methods (Zhang et al., 2021, 2023) in its simplicity. Two of our three methods—using an LLM to expand documents' representation or using an LLM to correct clustering outputs—can be added as a plug-in to *any text clustering algorithm* using *any set of text features*.[2] In our investigation of what aspect of the LLM prompt is most responsible for the clustering behavior, we find that just using an instruction alone (with no demonstrations) adds significant value. This can motivate future research directions for integrating natural language instructions with a clustering algorithm.

2 Methods to Incorporate LLMs

In this section, we describe the methods that we use to incorporate LLMs into clustering.

2.1 Clustering via LLM Keyphrase Expansion

Before any cluster is produced, experts typically know what aspects of each document they wish to capture during clustering. Instead of forcing clustering algorithms to mine such key factors from scratch, it could be valuable to globally highlight these aspects (and thereby specify the task emphases) beforehand. To do so, we use an LLM to make every document's textual representation *task-dependent*, by enriching and expanding it with evidence relevant to the clustering need. Specifically, each document is passed through an LLM which generates keyphrases. These keyphrases are encoded by an embedding model, and the keyphrase embedding is then concatenated to the original document embedding.

We generate keyphrases using GPT-3 (specifically, `gpt-3.5-turbo-0301`). We provide a short prompt to the LLM, starting with an instruction (e.g., *"I am trying to cluster online banking queries based on whether they express the same intent. For each query, generate a comprehensive set of keyphrases that could describe its intent, as a JSON-formatted list."*). The instruction is followed by four demonstrations of keyphrases, which resemble the example on the upper half of Figure 2.

We then encode the generated keyphrases into a single vector, and concatenate this vector with the original document's text representation. To disentangle the knowledge from an LLM with the benefits of a better encoder, we encode the keyphrases using the same encoder as the original text.[3] This approach is similar to Raedt et al. (2023), who generate keyphrases for unsupervised intent discovery.

[2] On the other hand, pairwise constraint clustering requires using K-Means as the underlying clustering algorithm.

[3] An exception to this is entity clustering. There, the BERT encoder has been specialized for clustering Wikipedia sentences, so we use DistilBERT to support keyphrase clustering.

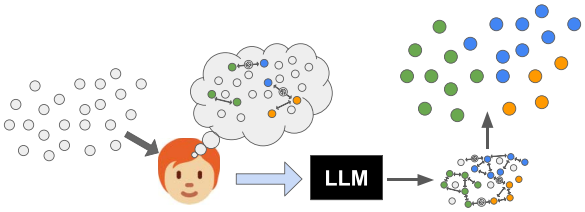

Figure 3: We use an LLM to generate pairwise constraints for a given dataset, given up to four examples of valid pairwise constraints. The pairwise constraint K-Means (‘‘PCKMeans’’) algorithm then consumes these ‘‘pseudo-oracle’’ constraints to produce clusters.

2.2 Pseudo-Oracle Pairwise Constraint Clustering

Arguably, the most popular approach to semi-supervised clustering is *pairwise constraint clustering*, where an oracle (e.g., a domain expert) selects pairs of points that *must* be linked or *cannot* be linked (Wagstaff and Cardie, 2000), such that the abstract clustering intentions of a user can be implicitly induced from their concrete feedback. In other words, a user conceptually describes which kinds of points to group together and wants to ensure the final clusters follow this grouping. We use this paradigm to investigate the potential of LLMs to amplify expert guidance during clustering by using an LLM as a *pseudo-oracle*, shown in Figure 3.

To select pairs to classify, we take different strategies for entity canonicalization and for other text clustering tasks. For text clustering, we adapt the Explore-Consolidate algorithm (Basu et al., 2004) to first collect a diverse set of pairs from embedding space (to identify pairs of points that cannot be linked), then collect points that are nearby to already-chosen points (to find pairs of points that must be linked). For entity canonicalization, where there are so many clusters that very few pairs of points belong to the same cluster, we simply sample the closest pairs of points in embedding space.

We prompt an LLM with a brief domain-specific instruction, followed by up to 4 demonstrations of pairwise constraints obtained from test set labels, to generate 20,000 pairwise constraints.[4] We use these pairwise constraints to generate clusters with the PCKMeans algorithm of Basu et al. (2004). This algorithm applies penalties for cluster assignments that violate any constraints, weighted by a hyperparameter w. Following Vashishth et al. (2018), we tune this parameter on each dataset's validation split. Due to the potential unreliability of pseudo-oracle pairwise constraints, we initialize our clusters using k-means++ (Arthur and Vassilvitskii, 2007) rather than directly using the pairwise constraint neighborhood structure as in prior work (Basu et al., 2004).

[4]We chose this number based on our available budget for LLM usage, after we empirically observed that pairwise constraint K-Means using noisy pairwise constraints worked better with an increasing number of constraints provided.

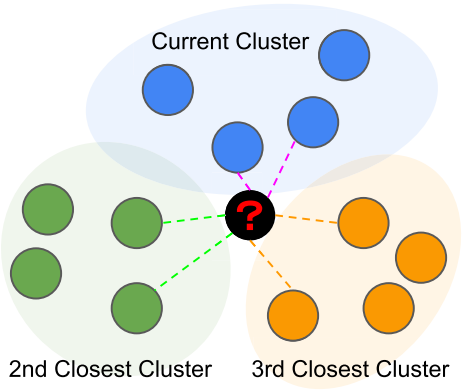

Figure 4: After performing clustering, we identify low-confidence points. For these points, we ask an LLM whether the current cluster assignment is correct. If the LLM responds negatively, we ask the LLM whether this point should instead be linked to any of the top-5 nearest clusters, and correct the clustering accordingly.

2.3 Using an LLM to Correct a Clustering

We finally consider the setting where one has an existing set of clusters, but wants to improve their quality with minimal local changes. We use the same pairwise constraint pseudo-oracle as in Section 2.2 to achieve this (see illustration in Figure 4).

We identify the *low-confidence points* by finding the k points with the least margin between the nearest and second-nearest clusters (setting $k = 500$ for our experiments). We textually represent each cluster by the entity nearest to the centroid of that cluster in embedding space. For each low-confidence point, we first ask the LLM whether this point is correctly linked to any of the representative points in its currently assigned cluster. If the LLM predicts that this point should not be linked to the current cluster, we consider the 4 next-closest clusters in embedding space as candidates for reranking, sorted by proximity. To rerank the current point, we ask the LLM whether

this point should be linked to the representative points in each candidate cluster. If the LLM responds positively, then we reassign the point to this new cluster. If the LLM responds negatively for all alternatives, we maintain the existing cluster assignment.

3 Tasks

3.1 Entity Canonicalization

Task. In *entity canonicalization*, we must group a collection of noun phrases $M = \{m_i\}_1^N$ into subgroups $\{C_j\}_1^K$ such that $m_1, m_2 \in C_j$ if and only if m_1 and m_2 refer to the same entity. For example, the noun phrases *President Biden* (m_1), *Joe Biden* (m_2), and *the 46th U.S. President* (m_3) should be clustered in one group (e.g., C_1). The set of noun phrases M are usually the nodes of an ''open knowledge graph'' produced by an Open Information Extraction (OIE) system.[5] Unlike the related task of entity linking (Bunescu and Pasca, 2006; Milne and Witten, 2008), we do not assume that any curated knowledge graph, gazetteer, or encyclopedia contains all the entities of interests.

Entity canonicalization is valuable for motivating the challenges of semi-supervised clustering. Here, there are hundreds or thousands of clusters and relatively few points per cluster, making this a difficult clustering task.

Datasets. We experiment with two datasets:

- *OPIEC59k* (Shen et al., 2022) contains 22K noun phrases (with 2,138 unique entity surface forms) belonging to 490 ground truth clusters. The noun phrases are extracted by MinIE (Gashteovski et al., 2017, 2019), and the ground truth entity clusters are anchor texts from Wikipedia that link to the same Wikipedia article.
- *ReVerb45k* (Vashishth et al., 2018) contains 15.5K mentions (with 12,295 unique entity surface forms) belonging to 6,700 ground truth clusters. The noun phrases are the output of the ReVerb (Fader et al., 2011) system, and the ''ground truth'' entity clusters come from automatically linking entities to the Freebase knowledge graph. We use the version of this dataset from Shen et al. (2022), who manually filtered it to remove labeling errors.

[5]OIE is the task of extracting surface-form *(subject; relation; object)*-triples from natural language text in a schema-free manner (Banko et al., 2007).

Canonicalization Metrics. We follow the standard metrics used by Shen et al. (2022):

- *Macro Precision and Recall*
  - Prec: For what fraction of predicted clusters is every element in the same gold cluster?
  - Rec: For what fraction of gold clusters is every element in the same predicted cluster?
- *Micro Precision and Recall*
  - Prec: How many points are in the same gold cluster as the majority of their predicted cluster?
  - Rec: How many points are in the same predicted cluster as the majority of their gold cluster?
- *Pairwise Precision and Recall*
  - Prec: How many pairs of points predicted to be linked are truly linked by a gold cluster?
  - Rec: How many pairs of points linked by a gold cluster are also predicted to be linked?

We finally compute the harmonic mean of each pair to obtain *Macro F1*, *Micro F1*, and *Pairwise F1*.

3.2 Text Clustering

Task. We then consider the case of clustering short textual documents. This clustering task has been extensively studied in the literature (Aggarwal and Zhai, 2012).

Datasets. We use three datasets in this setting:

- *Bank77* (Casanueva et al., 2020) contains 3,080 user queries for an online banking assistant from 77 intent categories.
- *CLINC* (Larson et al., 2019) contains 4,500 user queries for a task-oriented dialog system from 150 intent categories, after removing ''out-of-scope'' queries as in Zhang et al. (2023).
- *Tweet* (Yin and Wang, 2016) contains 2,472 tweets from 89 categories.

Metrics. Following prior work (Zhang et al., 2021), we compare our text clusters to the ground truth using normalized mutual information and

accuracy, which are obtained by finding the best alignment between ground truth and predicted clusters using the Hungarian algorithm (Kuhn, 1955).

4 Baselines

4.1 K-Means on Embeddings

We build our methods on top of a baseline of K-Means clustering (Lloyd, 1982) over encoded data with k-means++ cluster initialization (Arthur and Vassilvitskii, 2007). We choose the features and number of cluster centers that we use by task, largely following previous work. As an additional baseline, we also include hierarchical agglomerative clustering with a fixed number of clusters (Day and Edelsbrunner, 1984).

Entity Canonicalization. Following prior work (Vashishth et al., 2018; Shen et al., 2022), we cluster individual entity mentions (e.g., ''ever since the ancient Greeks founded the city of *Marseille* in 600 BC.'') by representing unique surface forms (e.g., ''Marseille'') globally, irrespective of their particular mention context. After clustering unique surface forms, we compose this cluster mapping onto the individual mentions (extracted from individual sentences) to obtain mention-level clusters.

We build on the ''multi-view clustering'' approach of Shen et al. (2022), and represent each noun phrase using textual mentions from the Internet and the ''open'' knowledge graph extracted from an OIE system. They use a BERT encoder (Devlin et al., 2019) to represent the textual context where an entity occurs (called the ''context view''), and a TransE knowledge graph encoder (Bordes et al., 2013) to represent nodes in the open knowledge graph (called the ''fact view''). They improve these encoders by finetuning the BERT encoder using weak supervision from coreferent entities and improving the knowledge graph representations using data augmentation on the knowledge graph. These two views of each entity are then combined to produce a representation.

In their original paper, they propose an alternating multi-view K-Means procedure where cluster assignments that are computed in one view are used to initialize cluster centroids in the other view. After a certain number of iterations, if the per-view clusterings do not agree, they perform a ''conflict resolution'' procedure to find a final clustering with low inertia in both views. One of our secondary contributions is a simplification of this algorithm. We find that by simply using their finetuned encoders, concatenating the representations from each view, and performing K-Means clustering with K-Means++ initialization (Arthur and Vassilvitskii, 2007) in a shared vector space, we can match their reported performance.

To select the number of cluster centers, following the Log-Jump method of Shen et al. (2022), we choose 490 and 6,687 clusters for OPIEC59k and ReVerb45k, respectively.

Intent Clustering. For the Bank77 and CLINC datasets, we follow Zhang et al. (2023) and encode each user query using the Instructor encoder. We use a simple prompt to guide the encoder: ''Represent utterances for intent classification''. Again following previous work, we choose 150 and 77 clusters for CLINC and Bank77, respectively.

Tweet Clustering. Following Zhang et al. (2021), we encode each tweet using a version of DistilBERT (Sanh et al., 2019) finetuned for sentence similarity classification[6] (Reimers and Gurevych, 2019). We use 89 clusters (Zhang et al., 2021).

4.2 Specialized Pretrained Encoders

Zhou et al. (2022) introduced a self-supervised pretrained encoder named Dialogue Sentence Embedding (*DSE*). It is explicitly trained to group together related utterances from dialogues. This model has been reported to achieve state-of-the-art results when used in a few-shot manner (where at least one seed point is provided for each cluster). Since our study focuses on the setting where the user provides substantially less feedback (3-4 instances of feedback), we consider this encoder, used in a zero-shot manner, combined with K-Means as another baseline for short text clustering tasks.

4.3 Clustering via Contrastive Learning

In addition to the methods described in Section 2, we also include two other methods for text clustering: SCCL (Zhang et al., 2021) and ClusterLLM (Zhang et al., 2023). These two methods both finetune an encoder to obtain better text clusters (in contrast to our method that can be used

[6]This model's name is `distilbert-base-nli-stsb-mean-tokens` on HuggingFace.

	Dataset / Method	OPIEC59k				ReVerb45k			
		Macro F1	**Micro F1**	**Pair F1**	***Avg***	**Macro F1**	**Micro F1**	**Pair F1**	***Avg***
	Optimal Clust.	80.3	97.0	95.5	90.9	84.8	93.5	92.1	90.1
	CMVC	52.8	90.7	84.7	76.1	66.1	87.9	89.4	81.1
	KMeans	53.5±0.0	91.0±0.0	85.6±0.0	76.7	69.6±0.0	89.1±0.0	89.3±0.0	82.7
	Agglomerative Clustering	32.8±0.0	87.2±0.0	83.4±0.0	67.8	70.0±0.0	89.5±0.0	89.8±0.0	83.1
	SCCL	56.4±0.0	88.4±0.0	78.7±0.0	74.5	–	–	–	–
ours	PCKMeans	58.7±0.0	91.5±0.0	86.1±0.0	78.7	72.0±0.0	88.5±0.0	87.0±0.0	82.5
ours	LLM Correction	58.7	91.5	85.2	78.4	69.9	89.2	88.4	82.5
ours	Keyph. Clust. (w/ KMeans)	**60.3**±0.0	**92.5**±0.0	**87.3**±0.0	**80.0**	**72.3**±0.0	**90.2**±0.0	**90.0**±0.0	**84.2**
ours	Keyph. Clust. (w/ Agglom.)	47.8±0.0	90.2±0.0	86.0±0.0	74.7	67.2±0.0	88.4±0.0	88.3±0.0	81.3

Table 1: Comparing methods for integrating LLMs into entity canonicalization. CMVC: multi-view clustering method of Shen et al. (2022). KMeans: our simplified reimplementation of the same method. Where applicable, standard deviations are obtained by running clustering 5 times with different seeds. We note that the standard deviations being displayed as 0.0 does not mean there was no variance; there was a nonzero standard deviation in most settings but this was less than 0.05 for all experiments.

	Dataset / Method	Bank77		CLINC		Tweet	
		Acc	**NMI**	**Acc**	**NMI**	**Acc**	**NMI**
	SCCL (Zhang et al., 2021)	–	–	–	–	78.2	89.2
	ClusterLLM (Zhang et al., 2023)	71.2	–	83.8	–	–	–
	KMeans	65.2±0.0	83.3±0.0	80.9±0.0	92.5±0.0	59.1±0.0	80.6±0.0
	Agglomerative Clustering	64.0±0.0	81.7±0.0	77.7±0.0	91.5±0.0	57.5±0.0	80.6±0.0
	DSE (w/ KMeans)	49.2±0.0	71.0±0.0	62.1±0.0	85.5±0.0	50.8±0.0	78.4±0.0
ours	PCKMeans	59.6±0.0	79.6±0.0	79.6±0.0	92.1±0.0	65.3±0.0	85.1±0.0
ours	LLM Correction	64.1	81.9	77.8	91.3	59.0	81.5
ours	Keyphrase Clust. (w/ KMeans)	65.3±0.0	82.4±0.0	79.4±0.0	92.6±0.0	62.0±0.0	83.8±0.0
ours	Keyphrase Clust. (w/ Agglom.)	**66.4**±0.0	**84.1**±0.0	**81.3**±0.0	**93.1**±0.0	**69.8**±0.0	**86.6**±0.0

Table 2: Comparing methods for integrating LLMs into text clustering. ‘‘DSE’’ refers to Zhou et al. (2022). We use the same base encoders as those methods in our experiments. Where applicable, standard deviations are obtained by running clustering 5 times with different seeds.

with a frozen encoder). To compare effectively with these approaches, we use the same base encoders reported for SCCL and ClusterLLM in prior works: Instructor (Su et al., 2022) for Bank77 and CLINC and DistilBERT[7] (Sanh et al., 2019; Reimers and Gurevych, 2019) for Tweet.

5 Results

5.1 Summary of Results

We summarize empirical results for entity canonicalization in Table 1 and text clustering in Table 2. As discussed in Section 4, when performing entity canonicalization, we assign mentions to the same cluster if they contain the same entity surface form (e.g., ‘‘*Marseille*’’), following prior work (Vashishth et al., 2018; Shen et al., 2022). This approach leads to irreducible errors for polysemous noun phrases (e.g., ‘‘Marseille’’ may refer to the athletic club Olympique de Marseille or the city Marseille). To the best of our knowledge, we are the first to highlight the limitations of this ‘‘surface form clustering’’ approach. We present the optimal performance under this assumption in Table 1, finding that the baseline of Shen et al. (2022) is already near-optimal on some metrics, particularly for ReVerb45k.

We find that using the LLM to expand textual representations is the most effective, achieving state-of-the-art results on both canonicalization datasets and significantly outperforming a K-Means baseline for all text clustering datasets.

[7]We used a version of DistilBERT finetuned for sentence similarity classification, following Zhang et al. (2023).

Pairwise constraint K-Means, when provided with 20K pairwise constraints pseudo-labeled by an LLM, achieves strong performance on 3 of 5 datasets (beating the current state-of-the-art on OPIEC59k).

For text clustering, our methods do not match the state-of-the-art results reported by SCCL (Zhang et al., 2021) and ClusterLLM (Zhang et al., 2023): both deep learning-based clustering approaches. We posit that this difference is largely due to these methods finetuning the encoder model to learn better representations. However, our work comes within 5-10 points of these state-of-the art results on every dataset. Our proposed keyphrase augmentation method also provide improvements over both K-Means and agglomerative clustering in most settings, demonstrating the modularity of this approach. Our proposed approach enables greater flexibility than the aforementioned state-of-the-art methods by supporting fixed features before clustering (e.g., those obtained via an embedding generation API). This enables our method to cluster non-textual data in addition to text. In contrast, finetuning encoders in multiple modalities is not supported with the aforementioned state-of-the-art methods. Moreover, unlike SCCL, our method works well on datasets with a very large number of clusters, as shown in Table 1. As a downside, our methods require more queries to the LLM than other methods like SCCL and ClusterLLM (see details in Sec. 5.4).

Below, we conduct more in-depth analyses on what makes each method (in-)effective.

5.2 Illustrative Examples & Key Factors

To qualitatively examine the impact of each LLM-based modification on the clustering process, we use the OPIEC59k dataset to compare the clusters obtained from our various clustering strategies with the clusters obtained from the K-Means baseline.

After aligning each clustering against the ground-truth using the Hungarian algorithm (Kuhn, 1955), we compute the Jaccard similarity between each predicted cluster and its corresponding ground truth cluster. Comparing the clusters obtained through our LLM-based interventions against baseline K-Means clusters, we identify clusters where each intervention provides the greatest improvement and the clusters where the intervention causes the greatest degradation.[8]

Method	# Improved	# Degraded
KMeans.	0	0
Keyphrase Clust.	168	83
PCKMeans	155	82
LLM Correction	102	51

Table 3: After aligning the output of each clustering algorithm with the ground truth, we report the number of clusters on OPIEC59k that were improved or worsened (measured by Jaccard similarity with the corresponding ground truth cluster). Each algorithm produced 490 clusters.

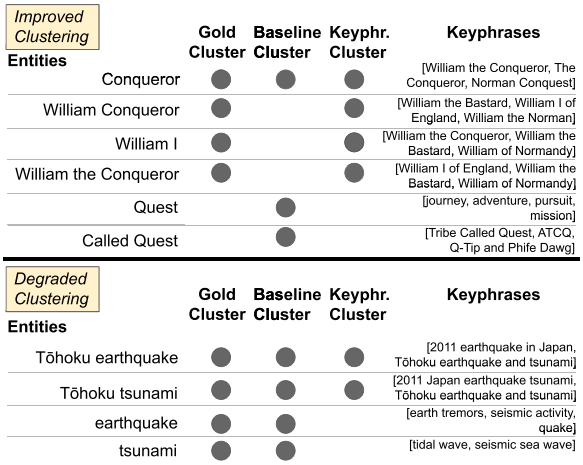

Figure 5: We identify clusters that changed after encoding and clustering keyphrases for each entity. Note that while we provide both the entity name and textual context about the entity to the clusterer, here we omit the textual context for display purposes.

While we show one improved cluster and one degraded cluster (relative to the K-Means baseline), these do not occur in equal proportions. In Table 3, we show the number of improved and degraded clusters for each method. In Figures 5, 6, and 7, we show examples of clusters after *keyphrase expansion*, *incorporating pairwise constraints*, and *LLM post correction*, and use them to provide intuitions for the key factors affecting each algorithm. On OPIEC59k, it is clear that all our LLM-based interventions mostly lead to improved clusters.

[8]We ignore clusters whose output from either algorithm has zero overlap with the corresponding ground truth cluster, since these may be due to cluster misalignment during evaluation.

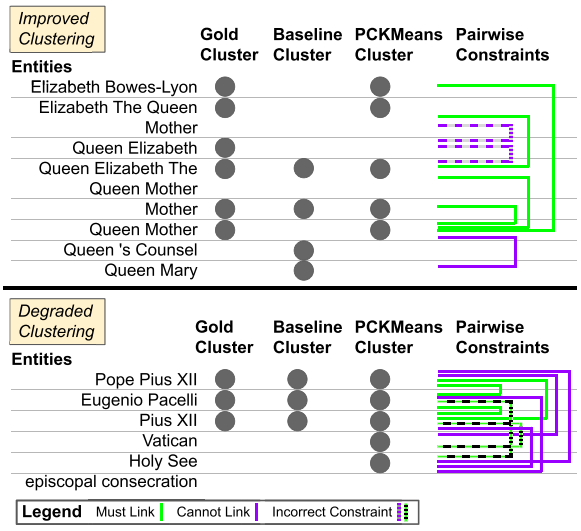

Figure 6: We identify clusters that changed after incorporating pairwise constraints and display the relevant pairwise constraints generated by the pseudo-oracle.

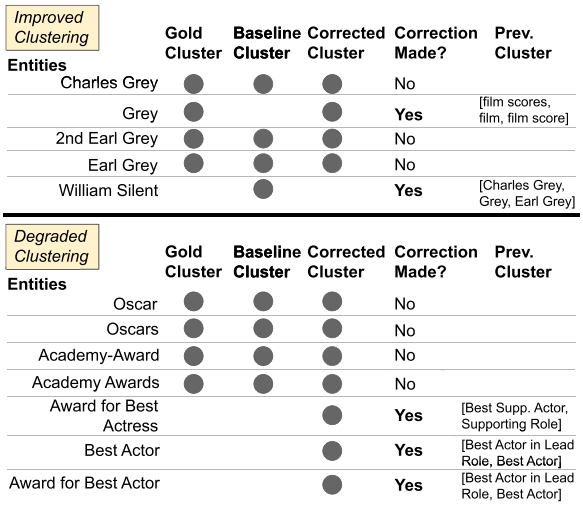

Figure 7: We identify clusters that changed after post-correcting cluster assignments with an LLM.

Keyphrase Clustering: Providing the Right Granularity for Disambiguation. In Figure 5, we see that LLM-generated keyphrases can disambiguate entities effectively (e.g., generating very different keyphrases for ‘‘Conqueror’’ and ‘‘Quest’’, while the embedding-based baseline clustering incorrectly groups these two). In the degraded example, we also see that these keyphrases may overly focus on each entity’s surface form rather than their textual context. This suggests room for more precise modeling and prompt engineering for leveraging keyphrases for complex documents.

PCKMeans: Incorrect and Conflicting Constraints Can Have Too Much Impact. As shown in Figure 6, in the improved case, the LLM accurately identifies relationships between some points (e.g., ‘‘Mother’’ and ‘‘Queen Mother’’) which were not grouped together by K-Means clustering on embeddings. In the degraded case, we see a case where the LLM generates conflicting constraints, leading to false positives. While the LLM correctly predicts that ‘‘Eugenio Pacelli’’ and ‘‘Pius XII’’ must be linked and ‘‘Pius XII’’ and ‘‘Holy See’’ cannot be linked, it incorrectly predicts a link between ‘‘Eugenio Pacelli’’ and ‘‘Holy See’’. As a result of these conflicting constraints, the PCKMeans algorithm incorrectly groups additional points into the cluster. Table 4 provides the accuracy for the pairwise constraints for some datasets, including OPIEC59k.

Datasets / Metrics	OPIEC59k	Tweet	Bank77
Data Size	2,138	4,500	3,080
Total Acc. of Pair. Constraints	86.7	96.8	81.7
# of LLM Reassignmnts	109	78	108
Acc. of Reassignments	55.0	89.7	41.7

Table 4: When re-ranking the top 500 points in each dataset, the LLM rarely disagrees from the original clustering. When it does, it is frequently wrong.

LLM Correction: Final, Hard Constraints Can Lead to Over-correction. In the degraded cluster in Figure 7, we see that the LLM fails to understand the granularity of this cluster, which should focus on *The Academy Awards* in general rather than a particular award presented at that ceremony. Despite the overall effectiveness of LLM correction for OPIEC59k (seen in Table 3), this example highlights a downside of this approach: We accept an absolute decision from the LLM for each point.

This finality impacts the effectiveness of LLM post-correction. In Table 1 and Table 2, the method consistently provides small gains on datasets over all metrics—between 0.1 and 5.2 absolute points of improvement. In Table 4, we see that when we provide the top 500 most-uncertain cluster assignments to the LLM to reconsider, the LLM only reassigns points in a small minority of cases. Though the LLM pairwise oracle is usually accurate, the LLM is disproportionately inaccurate for

points where the original clustering already had low confidence.

5.3 Ablation Study: Why do LLMs Excel at Text Expansion?

In Table 1 and Table 2, we see that *Keyphrase Clustering* is our strongest approach, achieving the best results on 3 of 5 datasets (and giving comparable performance to the next strongest method, *pseudo-oracle PCKMeans*, on the other 2 datasets). This suggests that LLMs are useful for expanding the contents of text to facilitate clustering.

What makes LLMs useful in this capacity? Is it the ability to specify task-specific modeling instructions, the ability to implicitly specify a similarity function via demonstrations, or do LLMs contain knowledge that smaller neural encoders lack? We answer these questions with an ablation study. For OPIEC59k and CLINC, we consider the Keyphrase Clustering technique but omit either the instruction or the demonstration examples from the prompt. For CLINC, we also compare with K-Means clustering on features from the Instructor model, which allows us to specify a short instruction to a small encoder.

Instructions and Demonstrations Have Complementary Gains. Empirically, we find that providing either instructions or demonstrations in the prompt to the LLM enables the LLM to improve cluster quality, but that providing both gives the most consistent positive effect. Qualitatively, we observe that providing instructions but omitting demonstrations leads to a larger set of keyphrases with less consistency, while providing demonstrations without any instructions leads to a more focused group of keyphrases that sometimes fail to reflect the desired aspect (e.g., topic vs. intent).

Instruction-finetuned Encoders Cannot Supply Enough Knowledge Why is keyphrase clustering using GPT-3.5 in the instruction-only (‘‘without demonstrations’’) setting better than using `Instructor` (an instruction-finetuned encoder)? The modest scaling curve suggests that scale is not solely responsible: GPT-3.5 likely contains similar or more parameters than GPT-3 (175B), while Su et al.’s (2022) `Instructor-base/large/XL` contain 110M, 335M parameters, and 1.5B parameters, respectively.

Dataset / Method	OPIEC59k	CLINC	
	Avg F1	**Acc**	**NMI**
Keyphrase Clust.	**80.0**	**79.4**±0.0	**92.6**±0.0
w/o Instructions	79.1	78.4±0.0	92.7±0.0
w/o Demonstrations	79.8	78.7±0.0	91.8±0.0
Instructor-base	–	74.8±0.0	90.7±0.0
Instructor-large	–	77.7±0.0	91.5±0.0
Instructor-XL (Su et al., 2022)	–	77.2±0.0	91.9±0.0
Instructor-XL (GPT-3.5 prompt)	–	70.8±0.0	88.6±0.0

Table 5: We compare the effect of LLM intervention without demonstrations or without instructions. We see that GPT-3.5-based Keyphrase Clustering outperforms instruction-finetuned encoders of different sizes, even when we provide the same prompt.

Note that we used two types of prompts: While our prompts for GPT-3.5 are very detailed, we used brief prompts for Instructor following their original design (e.g., ‘‘Represent utterances for intent classification’’). In addition, we experiment with giving the GPT-3.5 prompt to `Instructor-XL` (the bottom row of Table 5). We see that `Instructor-XL` performs more poorly on the prompt we give to GPT-3.5. We speculate that today’s instruction-finetuned encoders are insufficient to support the detailed, task-specific prompts that facilitate few-shot clustering.

5.4 Using an LLM as a Pseudo-oracle is Cost-effective

We have shown that using an LLM to guide the clustering process can improve cluster quality. However, large language models can be expensive; using a commercial LLM API during clustering imposes additional costs to the clustering process.

In Table 6, we summarize the pseudo-labeling cost of collecting LLM feedback using our three approaches. Among our three proposed approaches, pseudo-labeling pairwise constraints using an LLM (where the LLM must classify 20K pairs of points) incurs the greatest LLM API cost (using OpenAI’s `gpt-3.5-turbo-0301`, which cost $1 per million tokens when we ran our experiments). While PCKMeans and LLM Correction both query the LLM the same number

Method	Data Size	Cost in USD		
		PCKMeans	Correction	Keyphrase
OPIEC59k	2,138	$42.03	$12.73	$2.24
ReVerb45k	12,295	$33.81	$10.24	$10.66
Bank77	3,080	$10.25	$3.38	$1.23
CLINC	4,500	$9.77	$2.80	$0.95
Tweet	2,472	$11.28	$3.72	$0.99

Table 6: We compare the pseudo-labeling costs of different LLM-guided clustering approaches. We used OpenAI's `gpt-3.5-turbo-0301` API in June 2023.

of times for each dataset, Keyphrase Correction's cost scales linearly with the size of the dataset, making keyphrase correction infeasible for clustering very large corpora.

As such, this clustering method may require more LLM queries to support clustering than SCCL (which does not use an LLM at all) or ClusterLLM (which was shown to be effective using 1,000 queries to an LLM). More work is required to make our methods scale cheaply to very large-scale document collections, but we believe that the algorithmic simplicity of our method offers a promising starting point.

Does the improved performance justify this cost? Can we achieve better results at a comparable cost if we employed a human expert to guide the clustering process instead of an LLM? Since pseudo-labeling pairwise constraints requires the greatest API cost in our experiments, we take this approach as a case study. Given a sufficient amount of pseudo-oracle feedback, we see in Figure 8 that pairwise constraint K-Means is able to yield an improvement in Macro F1 (suggesting better purity of clusters) without dramatically reducing Pairwise or Micro F1.

Is this cost reasonable? For the $41 spent on the OpenAI API for OPIEC59k (as shown in Table 6), one could hire a worker for 3.7 hours of labeling time, assuming an $11-per-hour wage (Hara et al., 2017). We observe that an annotator can label roughly 3 pairs per minute. Then, $41 in worker wages would generate <700 human labels at the same cost as 20K GPT-3.5 labels.

Based on the feedback curve in Figure 8, we see that GPT-3.5 is remarkably more effective than a true oracle pairwise constraint oracle at this price point; unless at least 2,500 pairs labeled by a true oracle are provided, pairwise constraint K-Means fails to deliver any value for entity

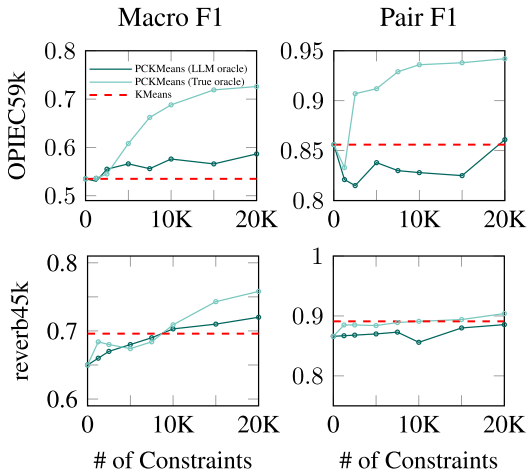

Figure 8: Collecting more pseudo-oracle feedback for pairwise constraint K-Means on OPIEC59k improves the Macro F1 metric without reducing other metrics. Compared to the same algorithm with true oracle constraints, we see the sensitivity of this algorithm to a noisy oracle.

canonicalization. This suggests that if the goal is maximizing empirical performance, querying an LLM is more cost-effective than employing a human labeler.

6 Related Work

Semi-supervised Clustering. Clustering with partial supervision has been extensively studied in the literature (Bae et al., 2020). These works include various types of interaction between an expert and a clustering algorithm, such as initializing clusters with cluster seeds (Basu et al., 2002), specifying pairwise constraints (Basu et al., 2004; Zhang et al., 2019), providing feature feedback (Dasgupta and Ng, 2010), splitting or merging clusters (Awasthi et al., 2013), or locking one cluster and refining the rest (Coden et al., 2017). In a simulation that used split/merge, pairwise constraints, and lock/refine interactions for clustering a small toy dataset, Coden et al. (2017) showed that it took between 20 and 100 human-machine interactions to get *any* clustering algorithm to obtain the desired clusters. Our work is distinct from these prior works in that we aim to obtain improved clusters from just a handful of feedback instances.

Entity Canonicalization. In addition to strong results shown for short text clustering (Yin and Wang, 2016; Casanueva et al., 2020; Larson et al., 2019), we achieve state-of-the-art results

on entity canonicalization. Entity canonicalization is typically viewed as an unsupervised clustering problem, but many prior works have incorporated data obtained via manual annotation, such as the outputs of a trained entity linker (Galárraga et al., 2014) or a manually annotated entity gazettes (Vashishth et al., 2018; Dash et al., 2020). In our work, following Shen et al. (2022), we avoid reliance on any such large-scale resources and only use the extracted open knowledge graph and text corpus (both used universally in entity canonicalization).

Clustering with LLMs. ClusterLLM (Zhang et al., 2023) introduced the use of LLMs for text clustering. It uses constrastive learning of deep encoders to improve clusters, using an LLM to improve the learned features. After running hierarchical clustering, they also use triplet feedback from the LLM[9] to decide the cluster granularity from the cluster hierarchy and generate a flat set of clusters. Unlike our proposed work, this method requires finetuning of the encoder and explicitly requires the use of agglomerative clustering. In contrast, we show that some of our proposed approaches can be modularly added to any clustering algorithm.

7 Conclusion

We find that using LLMs in simple ways can provide consistent improvements to the quality of clusters for a variety of text clustering tasks. We find that LLMs are most consistently useful as a means of enriching document representations, and we believe that our simple proof-of-concept should motivate more elaborate approaches for document expansion via LLMs.

Acknowledgments

This work was supported by a fellowship from NEC Laboratories Europe. We are grateful to Wiem Ben Rim, Saujas Vaduguru, and Jill Fain Lehman for their guidance. We also thank Chenyang Zhao for providing valuable feedback on this work.

References

Charu C. Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In *Mining Text Data*. https://doi.org/10.1007/978-1-4614-3223-4_4

David Arthur and Sergei Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*.

Pranjal Awasthi, Maria-Florina Balcan, and Konstantin Voevodski. 2013. Local algorithms for interactive clustering. *Journal of Machine Learning Research*, 18:3:1–3:35.

Juhee Bae, Tove Helldin, Maria Riveiro, Sławomir Nowaczyk, Mohamed-Rafik Bouguelia, and Göran Falkman. 2020. Interactive clustering: A comprehensive review. *ACM Computing Surveys*, 53(1):1–39. https://doi.org/10.1145/3340960

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *CACM*.

Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. 2002. Semi-supervised clustering by seeding. In *International Conference on Machine Learning*.

Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. 2004. Active semi-supervision for pairwise constrained clustering. In *SDM*. https://doi.org/10.1137/1.9781611972740.31

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2787–2795, Red Hook, NY, USA. Curran Associates Inc.

Razvan C. Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Rich Caruana. 2013. Clustering: Probably approximately useless? In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 1259–1260, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/2505515.2514692

[9] *"is point A more similar to point B or point C?"*.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.nlp4convai-1.5

Anni Coden, Marina Danilevsky, Daniel F. Gruhl, Linda Kato, and Meena Nagarajan. 2017. A method to accelerate human in the loop clustering. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. https://doi.org/10.1137/1.9781611974973.27

Sajib Dasgupta and Vincent Ng. 2010. Which clustering do you want? Inducing your ideal clustering with minimal feedback. *Journal of Artificial Intelligence Research*, 39:581–632. https://doi.org/10.1613/jair.3003

Sarthak Dash, Gaetano Rossiello, Nandana Mihindukulasooriya, Sugato Bagchi, and A. Gliozzo. 2020. Open knowledge graphs canonicalization using variational autoencoders. In *Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/2021.emnlp-main.811

William H. E. Day and Herbert Edelsbrunner. 1984. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1:7–24. https://doi.org/10.1007/BF01890115

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Conference on Empirical Methods in Natural Language Processing*.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. GPTscore: Evaluate as you desire. *ArXiv*, abs/2302.04166.

Luis Galárraga, Geremy Heitz, Kevin P. Murphy, and Fabian M. Suchanek. 2014. Canonicalizing open knowledge bases. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. https://doi.org/10.1145/2661829.2662073

Kiril Gashteovski, Rainer Gemulla, and Luciano Del Corro. 2017. Minie: Minimizing facts in open information extraction. In *Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/D17-1278

Kiril Gashteovski, Sebastian Wanner, Sven Hertling, Samuel Broscheit, and Rainer Gemulla. 2019. Opiec: An open information extraction corpus. In *Proceedings of the Conference on Automatic Knowledge Base Construction (AKBC)*.

Kotaro Hara, Abigail Adams, Kristy Milland, Saiph Savage, Chris Callison-Burch, and Jeffrey P. Bigham. 2017. A data-driven analysis of workers' earnings on Amazon Mechanical Turk. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3173574.3174023

John J. Horton. 2023. Large language models as simulated economic agents: What can we learn from homo silicus? Working Paper 31122, National Bureau of Economic Research. https://doi.org/10.3386/w31122

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. 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)*, pages 1311–1316,

Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1131

S. Lloyd. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137. https://doi.org/10.1109/TIT.1982.1056489

David N. Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *International Conference on Information and Knowledge Management*. https://doi.org/10.1145/1458082.1458150

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*.

Maarten De Raedt, Fréderic Godin, Thomas Demeester, and Chris Develder. 2023. Idas: Intent discovery with abstractive summarization. *ArXiv*, abs/2305.19783. https://doi.org/10.18653/v1/2023.nlp4convai-1.7

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Wei Shen, Yang Yang, and Yinan Liu. 2022. Multi-view clustering for open knowledge base canonicalization. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pages 1578–1588, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3534678.3539449

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. One embedder, any task: Instruction-finetuned text embeddings. In *arXiv*.

Shikhar Vashishth, Prince Jain, and Partha Talukdar. 2018. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference*, pages 1317–1327. https://doi.org/10.1145/3178876.3186030

Kiri L. Wagstaff and Claire Cardie. 2000. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*.

Jianhua Yin and Jianyong Wang. 2016. A model-based approach for text clustering with outlier detection. *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 625–636. https://doi.org/10.1109/ICDE.2016.7498276

Dejiao Zhang, Feng Nan, Xiaokai Wei, Shang-Wen Li, Henghui Zhu, Kathleen McKeown, Ramesh Nallapati, Andrew O. Arnold, and Bing Xiang. 2021. 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*, pages 5419–5430, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.427

Hongjing Zhang, Sugato Basu, and Ian Davidson. 2019. A framework for deep constrained clustering - algorithms and advances. In *ECML/PKDD*. https://doi.org/10.1007/978-3-030-46150-8_4

Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023. Clusterllm: Large language models as a guide for text clustering. *ArXiv*, abs/2305.14871. https://doi.org/10.18653/v1/2023.emnlp-main.858

Zhihan Zhou, Dejiao Zhang, Wei Xiao, Nicholas Dingwall, Xiaofei Ma, Andrew Arnold, and Bing Xiang. 2022. Learning dialogue representations from consecutive utterances. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 754–768, Seattle, United States. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.naacl-main.55