A Generic Method for Fine-grained Category Discovery in Natural Language Texts

Chang Tian[§] **Matthew B. Blaschko**[§] **Wenpeng Yin**[†]
Mingzhe Xing[△*] **Yinliang Yue**[△] **Marie-Francine Moens**[§]
[§] KU Leuven [†]Penn State University [△] Zhongguancun Laboratory
chang.tian@kuleuven.be xingmz@zgclab.edu.cn

Abstract

Fine-grained category discovery using only coarse-grained supervision is a cost-effective yet challenging task. Previous training methods focus on aligning query samples with positive samples and distancing them from negatives. They often neglect intra-category and inter-category semantic similarities of fine-grained categories when navigating sample distributions in the embedding space. Furthermore, some evaluation techniques that rely on pre-collected test samples are inadequate for real-time applications. To address these shortcomings, we introduce a method that successfully detects fine-grained clusters of semantically similar texts guided by a novel objective function. The method uses semantic similarities in a logarithmic space to guide sample distributions in the Euclidean space and to form distinct clusters that represent fine-grained categories. We also propose a centroid inference mechanism to support real-time applications. The efficacy of the method is both theoretically justified and empirically confirmed on three benchmark tasks. The proposed objective function is integrated in multiple contrastive learning based neural models. Its results surpass existing state-of-the-art approaches in terms of Accuracy, Adjusted Rand Index and Normalized Mutual Information of the detected fine-grained categories. Code and data are publicly available at https://github.com/changtianluckyforever/F-grained-STAR.

1 Introduction

Fine-grained analysis has drawn much attention in many artificial intelligence fields, e.g., Computer Vision (Chen et al., 2018; Li et al., 2021; Wang et al., 2024a; Park and Ryu, 2024) and Natural Language Processing (Ma et al., 2023; Tian et al., 2024; An et al., 2024), because it can provide more detailed features than coarse-grained data. For instance, as illustrated in Figure 1, solely

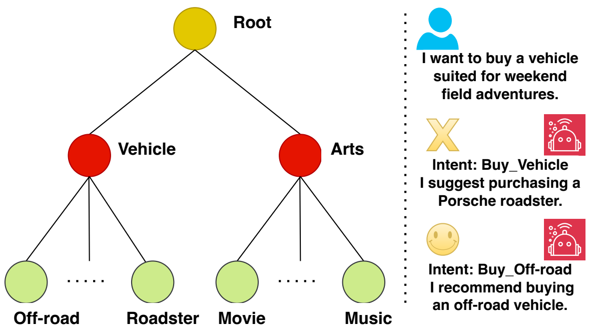

Root
Vehicle
Arts
Off-road
Roadster
Movie
Music
I want to buy a vehicle suited for weekend field adventures.
Intent: Buy_Vehicle
I suggest purchasing a Porsche roadster.
Intent: Buy_Off-road
I recommend buying an off-road vehicle.

Figure 1: A fine-grained intent detection example. **Left**: This panel illustrates the label hierarchy, transitioning from coarse-grained to fine-grained granularity. **Right**: This example demonstrates intent detection in a conversation about car choices, showing how coarse-grained analysis alone can lead to incorrect recommendations by a life assistant due to a lack of fine-grained analysis.

based on coarse-grained analysis, the chatbot might incorrectly recommend a roadster, which is unsuitable for field adventures. Detecting the fine-grained intent would allow the chatbot to recommend an off-road vehicle that aligns with the user's requirements. However, annotating fine-grained categories can be labor-intensive, as it demands precise expert knowledge specific to each domain and involved dataset. Addressing this challenge, An et al. (2022) recently introduced Fine-grained Category Discovery under Coarse-grained Supervision (**FCDC**) for language classification tasks (details in Section 3). Solving FCDC tasks can significantly benefit numerous practical applications, for example, fine-grained classification of enterprise documents (Chen et al., 2023; Vellmer et al., 2023), fine-grained dialogue intent detection tasks (Tian et al., 2022; Lichouri et al., 2024), product labeling on online shopping websites based on text descriptions (Ghani and Fano, 2002; Parekh et al., 2021), and so on. FCDC aims to reduce annotation costs by leveraging the relative ease of obtaining coarse-grained annotations, without re-

[*]Corresponding author.

quiring fine-grained supervisory information. This approach has sparked significant research interest in the automatic discovery of fine-grained language categories (Ma et al., 2023; An et al., 2023a; Vaze et al., 2024; Lian et al., 2024).

Existing methods for addressing FCDC are typically grouped into three groups (An et al., 2024): language models, self-training methods, and contrastive learning methods. Language models (Devlin et al., 2019a; Touvron et al., 2023), including their fine-tuned versions with coarse labels, generally perform poorly on this task due to a lack of fine-grained supervision. Self-training methods (Caron et al., 2018; Zhang et al., 2021) and their variants often employ clustering assignments as fine-grained pseudo-labels, filtering out some noisy pseudo-labels, and training with these labels. Dominant contrastive learning methods (Chen et al., 2020; Mekala et al., 2021; An et al., 2022, 2023a) typically identify positive and negative samples for a given query by measuring their semantic distances. The contrastive loss ensures that the query sample moves closer to positive samples and further away from negative samples. So these methods form clusters of samples in the embedding space, with each cluster representing a discovered fine-grained category, without requiring fine-grained category supervision.

However, past methods did not utilize **comprehensive semantic similarities** (**CSS**) in the logarithmic space to guide sample distributions in the Euclidean space. We define CSS as the fine-grained semantic similarities measured by bidirectional Kullback-Leibler (KL) divergence in the logarithmic space between the query sample and each available positive or negative sample. Although An et al. (2024) recently explored similarities measured by rank order between the query sample and positive samples, they ignore similarities with negative samples.

We propose a method (**STAR**) for detecting fine-grained clu**st**ers of semantically simil**ar** texts through a novel objective function, with the core component considering CSS. This component guides sample distributions in the Euclidean space based on the magnitude of CSS in the logarithmic space. Large semantic differences (low similarity) in the logarithmic space between the query sample and an available sample push the query sample further away in the Euclidean space, while small semantic differences bring the query sample closer to the available sample. Thus, samples form distinguishable fine-grained clusters in the Euclidean space, with each cluster representing a discovered category.

Additionally, clustering inference used by previous works (An et al., 2022, 2023a, 2024) cannot support real-time scenarios, so we propose a variant inference mechanism utilizing approximated fine-grained cluster centroids, delivering competitive results for the tasks considered.

Our main contributions in this work can be summarized as follows:

- **Method:** STAR enhances existing contrastive learning methods by leveraging comprehensive semantic similarities in a logarithmic space to guide sample distributions in the Euclidean space, thereby making fine-grained categories more distinguishable.
- **Theory:** We interpret STAR from the perspectives of clustering and generalized Expectation Maximization (EM). Also, we conduct loss and gradient analyses to explain the effectiveness of using CSS for category discovery.
- **Experiments:** Experiments on three text classification tasks (intent detection (Larson et al., 2019), scientific abstract classification (Kowsari et al., 2017), and chatbot query (Liu et al., 2021)) demonstrate new state-of-the-art (SOTA) performance compared to 22 baselines, validating the theoretical method.

2 Related Work

2.1 Fine-grained Category Discovery

Fine-grained data analysis is crucial in Natural Language Processing (Guo et al., 2021; Ma et al., 2023; Tian et al., 2024) and Computer Vision (Pan et al., 2023; Wang et al., 2024b). However, effectively discovering fine-grained categories from coarse-grained ones remains challenging (Mekala et al., 2021). Traditional category discovery methods often assume that known and discovered categories are at the same granularity level (An et al., 2023b; Vaze et al., 2024).

To discover fine-grained categories under the supervision of coarse-grained categories, An et al. (2022) introduces the FCDC task. Self-training approaches, such as Deep Cluster (Caron et al., 2018; An et al., 2023a), use clustering algorithms to detect the fine-grained categories, assign pseudo-labels to the clusters and their samples, and then train a classification model with these pseudo-labels. Its variant, Deep Aligned Clustering (Zhang et al., 2021), devises a strategy to filter out inconsistent pseudo-labels during clustering. Contrastive learning has become prevalent in FCDC tasks; Bukchin et al. (2021) and An et al. (2022) develops angular contrastive learning tailored for fine-grained classification. An et al. (2022) proposes a weighted self-contrastive framework to enhance the model's discriminative capacity for coarse-grained samples. Ma et al. (2023) and An et al. (2023a) uses noisy fine-grained centroids and retrieves neighbors as positive pairs, respectively, applying constraints to filter noise. An et al. (2024) advances this approach with neighbors that are manually weighted as positive pairs. However, previous efforts have not leveraged comprehensive semantic similarities to guide sample distributions and thereby to enhance fine-grained category discovery.

2.2 Neighborhood Contrastive Learning

Contrastive learning enhances representation learning by bringing the query sample closer to positive samples and distancing it from negative samples (Chen et al., 2020). Prior research has focused on constructing high-quality positive pairs. He et al. (2020) utilizes two different transformations of the same input as query and positive sample, respectively. Li et al. (2020) introduces the use of prototypes, derived through clustering, as positive instances. Additionally, An et al. (2022) employs shallow-layer features from BERT as positive samples and introduces a weighted contrastive loss. This approach primarily differentiates data at a coarse-grained level, and the manually set weights limit its broader applicability.

To circumvent complex data augmentation, neighborhood contrastive learning (NCL) is developed, treating the nearest neighbors of queries as positive samples (Dwibedi et al., 2021). Zhong et al. (2021) extends this by utilizing k-nearest neighbors to identify hard negative samples, while Zhang et al. (2022) selects a positive key from the k-nearest neighbors for contrastive representation learning. However, these approaches often deal with noisy nearest neighbors that include false-positive samples. An et al. (2023a) addresses this by proposing three constraints to filter out uncertain neighbors, yet they overlooks semantic similarities between query sample and each available sample. An et al. (2024) represents semantic similarities using rank order among positive samples but neglects similarities among negative samples. In contrast, STAR uses comprehensive semantic similarities to guide sample distributions in the Euclidean space, offering richer features and a superior approach to pure contrastive learning.

3 Problem Formulation

Given a set of coarse-grained categories $Y_{coarse} = \{C_1, C_2, \dots, C_M\}$ and a coarsely labeled training set $D_{train} = \{(x_i, c_i) \mid c_i \in Y_{coarse}\}_{i=1}^{N}$, where N denotes the number of training samples, the task of FCDC involves developing a feature encoder F_θ. This encoder maps samples into a feature space, further segmenting them into distinct fine-grained categories $Y_{fine} = \{F_1, F_2, \dots, F_K\}$, without any fine-grained supervisory information. Here, Y_{fine} represents sub-classes of Y_{coarse}. Model effectiveness is evaluated on a testing set $D_{test} = \{(x_i, y_i) \mid y_i \in Y_{fine}\}_{i=1}^{L}$, with L as the number of test samples, utilizing features extracted by F_θ. For evaluation consistency and fairness, only the number of fine-grained categories K is used, aligning with methodologies established in previous research (Ma et al., 2023; An et al., 2022, 2023a).

4 Method

STAR leverages comprehensive semantic similarities and integrates seamlessly with contrastive learning baselines by modifying the objective function. We have developed variants for three baselines: PseudoPrototypicalNet (PPNet) (Boney and Ilin, 2017; Ji et al., 2020), DNA (An et al., 2023a), and DOWN (An et al., 2024). This section focuses on STAR-DOWN because DOWN outperforms other baselines, with additional method variants detailed in Appendix A.3.

DOWN involves three steps: pre-training with coarse-grained labels (Section 4.1), retrieving and weighting nearest neighbors (Section 4.2), and training with a contrastive loss. STAR-DOWN follows the same first two steps but replaces the third with a novel objective function (Section 4.3). Like DOWN, STAR-DOWN iterates the last two steps until the unsupervised metric, the silhouette score of the clustering into fine-grained clusters, does not improve for five consecutive epochs. The detailed algorithm is provided in Appendix A.1.7.

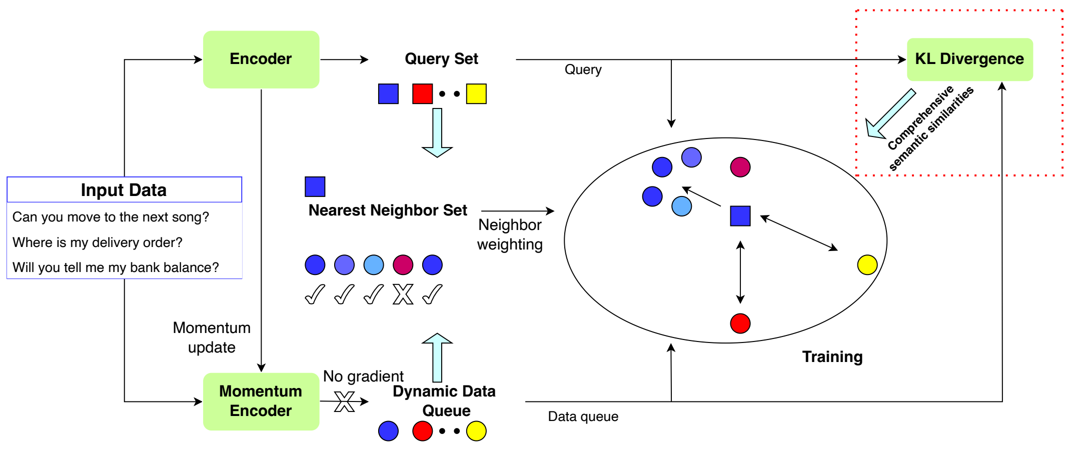


Figure 2: STAR-DOWN integrates the baseline DOWN with the STAR method (shown in the red dashed box). In the visual representation, colors differentiate samples, squares represent features extracted by the Encoder, and circles denote features extracted by the Momentum Encoder. Unidirectional arrows indicate proximity, while bidirectional arrows signify distance between samples.

4.1 Multi-task Pre-training

As illustrated in Figure 2, the baseline DOWN (An et al., 2024) utilizes the BERT Encoder F_θ to extract normalized feature embeddings $q_i = F_\theta(x_i)$ for input x_i, where θ represents the Encoder parameters. To ensure effective initialization for fine-grained training, DOWN pre-trains the Encoder on the coarsely labeled train set D_{train} with labels Y_{coarse}. DOWN utilizes the sum of a cross-entropy loss L_{ce} and a masked language modeling loss L_{mlm} for multi-task pre-training of the Encoder (detailed in Appendix A.1.1).

4.2 Neighbors Retrieval and Weighting

The Momentum Encoder is a slowly evolving version of the Encoder, commonly employed in self-supervised learning (He et al., 2020; An et al., 2023a). DOWN integrates the Momentum Encoder to generate more consistent, stable, and better representations over time (An et al., 2024).

In Figure 2, the Momentum Encoder F_{θ_k} with parameters θ_k extracts and stores gradient-free normalized neighbor features $h_i = F_{\theta_k}(x_i)$ in a dynamic data queue Q. To ensure consistency between the outputs of F_{θ_k} and F_θ, F_{θ_k}'s parameters are updated via a moving-average method (He et al., 2020): $\theta_k \leftarrow m\theta_k + (1-m)\theta$, where m is the momentum coefficient. For each query feature q_i, in order to facilitate semantic similarity capture and fine-grained clustering, its top-k nearest neighbors N_i are determined from Q using cosine similarity (Sim): $N_i = \{h_j \mid h_j \in \operatorname{argtop}_{h_l \in Q} {}_k(\text{Sim}(q_i, h_l))\}$, where $\text{Sim}(q_i, h_l) = \frac{q_i^{\mathrm{T}} h_l}{\|q_i\| \cdot \|h_l\|}$ is the cosine similarity function.

To counteract potential false positives in N_i, DOWN utilizes a soft weighting mechanism based on neighbor rank to balance information utility against noise, with weights ω_j of neighbor h_j calculated as: $\omega_j = \phi \cdot \alpha^{-\frac{l_{ij}}{k}}$, where ϕ is a normalizing constant for weights, α serves as the exponential base, k is the retrieved neighbor count, and l_{ij} denotes the rank of h_j as a neighbor to q_i.

To align with the model's evolving accuracy in neighbor retrieval during training, DOWN periodically decreases α every five epochs, the values for α in ω_j are: $\alpha_{\text{set}} = \{150, 10, 5, 2\}$. The ω_j of each positive sample h_j is used in Eqs. 3 and 4.

4.3 Training

4.3.1 Objective Function

Given a training batch $N_{train} \in D_{train}$, where Y_c is the set of coarse-grained labels of N_{train}, DOWN trains the model using the loss:

$$L_{\text{train}} = L_{\text{ce}} + L_{\text{DOWN}}, \tag{1}$$

$$L_{\text{DOWN}} = \frac{1}{|N_{train}|} \sum_{q_i \in N_{train}} L_1^i. \tag{2}$$

As shown in Eq. 3, DOWN uses a conventional contrastive objective function in the Euclidean space, while STAR-DOWN introduces a novel objective function in Eq. 4, leveraging CSS in the logarithmic space to guide sample distributions in the Euclidean space, the temperature τ is a fixed constant in Eq. 3 and Eq. 4:

$$L_1^i = -\sum_{h_j \in N_i} \omega_j \cdot \log \frac{\exp(q_i^{\mathrm{T}} h_j/\tau)}{\sum\limits_{h_k \in Q} \exp(q_i^{\mathrm{T}} h_k/\tau)}. \quad (3)$$

$$\begin{aligned} L_2^i = -\gamma \sum_{h_j \in N_i} \omega_j \cdot \log \frac{\exp(-d_{KL}(q_i, h_j)/\tau)}{\sum\limits_{h_k \in Q} \exp(-d_{KL}(q_i, h_k)/\tau)} \\ - \sum_{h_j \in N_i} \omega_j \cdot \log \frac{\exp(q_i^{\mathrm{T}} h_j/\tau)}{\sum\limits_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau)}. \end{aligned} \quad (4)$$

During training, STAR-DOWN optimizes the following objective function:

$$L_{\text{train}} = L_{\text{ce}} + L_{\text{STAR}}, \quad (5)$$

$$L_{\text{STAR}} = \frac{1}{|N_{train}|} \sum_{q_i \in N_{train}} L_2^i. \quad (6)$$

As shown in Eq. 4, the term $d_{KL}(q_i, h_k)$ in L_2^i represents the bidirectional KL divergence in a logarithmic space between the query sample embedding q_i and the data queue sample embedding h_k (detailed in Appendix A.1.2). B is a trainable scalar representing the exponential base.

The first term in L_2^i minimizes the KL divergence between query samples and positive samples (the retrieved top-k nearest neighbors N_i in Section 4.2) while increasing it for negative samples (the samples in data queue Q apart from the positive samples) in the logarithmic space, with γ as a balancing hyperparameter. The second term in L_2^i uses CSS in the logarithmic space, denoted by $B^{d_{KL}(q_i,h_k)}$, to guide query sample distribution in the Euclidean space. $q_i^{\mathrm{T}} h_k$ quantifies the cosine similarity between normalized q_i and h_k, equivalent to the negative Euclidean distance (detailed in Appendix A.1.4). The value of the trainable scalar B is updated during loss backpropagation, so $B^{d_{KL}(q_i,h_k)}$ is fully trainable and can integrate with contrastive learning methods, making the STAR method *generic*.

4.3.2 Loss Analysis

The loss L_2^i consists of two terms. The first is a contrastive loss that optimizes sample distribution in logarithmic space, ensuring that similar samples have a small KL divergence $d_{KL}(q_i, h_k)$, while dissimilar samples exhibit a large $d_{KL}(q_i, h_k)$. These semantic similarities are then used as weights in the second contrastive loss term $q_i^{\mathrm{T}} h_k$, optimizing the sample distribution in Euclidean space.

KL divergences grow logarithmically, their scale increases slowly, making it challenging to differentiate semantic differences. In contrast, exponentiation scales rapidly. To address this, we apply exponentiation to amplify semantic distinctions, using a trainable scalar base B and an exponent $d_{KL}(q_i, h_k)$ from the logarithmic space. This results in weights of $B^{d_{KL}(q_i,h_k)}$ for $q_i^{\mathrm{T}} h_k$.

Since STAR-DOWN discovers fine-grained categories in the Euclidean space, we analyze the second term L_{2-2}^i of the loss L_2^i, which optimizes sample distributions in the Euclidean space:

$$\begin{aligned} L_{2-2}^i &= -\sum_{h_j \in N_i} \omega_j \cdot \log \frac{\exp(q_i^{\mathrm{T}} h_j/\tau)}{\sum\limits_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau)} \\ &= \sum_{h_j \in N_i} \omega_j \cdot (\log \sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau) \\ &\quad - (q_i^{\mathrm{T}} h_j/\tau)). \end{aligned} \quad (7)$$

In the loss L_{2-2}^i, $B^{d_{KL}(q_i,h_k)}$ uses CSS in the logarithmic space to guide sample distributions in the Euclidean space. A large $d_{KL}(q_i, h_k)$ (low semantic similarity) causes q_i to distance itself from h_k in the Euclidean space, reducing $q_i^{\mathrm{T}} h_k$, while a small $d_{KL}(q_i, h_k)$ allows q_i to remain relatively close to h_k compared to negative samples. This results in the formation of compact fine-grained clusters, with each cluster representing a discovered category.

Unlike traditional contrastive loss, which multiplies $\exp\left(\frac{q_i^{\mathrm{T}} h_k}{\tau}\right)$ by 1, our STAR method incorporates logarithmic space semantic differences, $B^{d_{KL}(q_i,h_k)}$, as weights[1] for each sample pair. This is expressed as $B^{d_{KL}(q_i,h_k)} \cdot \exp\left(\frac{q_i^{\mathrm{T}} h_k}{\tau}\right)$. As a result, distant samples are pushed further apart in Euclidean space, while closer samples remain near, facilitating the formation of more distinct boundaries. We also analyze the STAR method from the perspectives of **gradient**, **clustering**, and **generalized EM**. Detailed analyses are provided in Appendix A.2.

4.4 Inference

Previous methods (An et al., 2023a, 2024) use clustering inference on sample embeddings from F_θ extracted from D_{test}, which is unsuitable for real-time tasks, such as intent detection, which require immediate response and cannot wait to collect enough test samples for clustering. We introduce

[1]For interesting form similarities to physical laws within the STAR method, see Appendix A.1.3.

Dataset	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{F}\rvert$	# Train	# Test
CLINC	10	150	18000	1000
WOS	7	33	8362	2420
HWU64	18	64	8954	1031

Table 1: Statistics of datasets (An et al., 2023a). #: number of samples. $|\mathcal{C}|$: number of coarse-grained categories. $|\mathcal{F}|$: number of fine-grained categories.

an alternative, centroid inference, suitable for both real-time and other contexts. Using F_θ, we derive sample embeddings from D_{train} and assign fine-grained pseudo-labels through clustering. For each fine-grained cluster, only the embeddings of samples from the predominant coarse-grained category (the category with the most samples in this fine-grained cluster) are averaged to form centroid representations. These approximated centroids are used to determine the fine-grained category of each test sample based on cosine similarity. A visual explanation is in Appendix A.1.5.

5 Experiments

5.1 Experimental Settings

5.1.1 Datasets

We conduct experiments on three benchmark datasets: CLINC (Larson et al., 2019), WOS (Kowsari et al., 2017), and HWU64 (Liu et al., 2021). CLINC is an intent detection dataset spanning multiple domains. WOS is used for paper abstract classification, and HWU64 is designed for assistant query classification. Dataset statistics are provided in Table 1.

5.1.2 Baselines for Comparison

We compare our methods against the following baselines. **Language models**: BERT (Devlin et al., 2019b), BERT with coarse-grained fine-tuning, Llama2 (Touvron et al., 2023), Llama2 with coarse-grained fine-tuning and GPT4 (Achiam et al., 2023). **Self-training** baselines: DeepCluster (DC) (Caron et al., 2018), DeepAlignedCluster (DAC) (Zhang et al., 2021), and PseudoPrototypicalNet (PPNet) (Boney and Ilin, 2017; Ji et al., 2020). **Contrastive learning** baselines: SimCSE (Gao et al., 2021), Ancor (Bukchin et al., 2021), Delete (Wu et al., 2020), Nearest-Neighbor Contrastive Learning (NNCL) (Dwibedi et al., 2021), Contrastive Learning with Nearest Neighbors (CLNN) (Zhang et al., 2022), Soft Neighbor Contrastive Learning (SNCL) (Chongjian et al., 2022), Weighted Self-Contrastive Learning (WSCL) (An et al., 2022), Denoised Neighborhood Aggregation (DNA), and Dynamic Order Weighted Network (DOWN) (An et al., 2023a, 2024). We also explore variants incorporating the cross-entropy loss (+CE).

5.1.3 Evaluation Metrics

To evaluate the quality of the discovered fine-grained clusters, we use the Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) and Normalized Mutual Information (NMI) (Lancichinetti et al., 2009). For assessing classification performance, we use clustering Accuracy (ACC) (Kuhn, 2010; An et al., 2023a). Detailed descriptions of these metrics are provided in Appendix A.5.

5.1.4 Implementation Details

To ensure fair comparisons with baselines, we use the BERT-base-uncased model as the backbone for all STAR method variants. We adhere to the hyperparameters used by the integrated baselines to demonstrate the effectiveness of our STAR method. The learning rate for both pre-training and training is $5e^{-5}$, using the AdamW optimizer with a 0.01 weight decay and 1.0 gradient clipping. The batch size for pre-training, training, and testing is 64. The temperature τ is set to 0.07. The exponential base B in loss is set to 10. The number of neighbors k is set to {120, 120, 250} for the CLINC, HWU64, and WOS datasets, respectively. Epochs for pretraining and training are set to 100 and 20, respectively. The γ values are {0.03, 0.05, 0.1} for the CLINC, HWU64, and WOS datasets. The momentum coefficient m is set to 0.99. Further details are provided in Appendix A.4.

5.1.5 Research Questions

The following research questions (**RQ**s) are investigated: 1. What is the impact of STAR method on FCDC tasks? 2. What are the effects of the proposed real-time centroid inference compared to traditional clustering inference? 3. How does each component of the STAR method affect performance? 4. How can we effectively and efficiently set the base for the exponential function in the STAR method?

5.2 Result Analysis (RQ1)

As shown in Table 2, STAR method variants outperform SOTA methods across all datasets and metrics, validating the effectiveness of the STAR method in FCDC tasks. Language models like BERT, Llama2 and GPT4 (Devlin et al., 2019b; Touvron et al.,

Methods	**HWU64**			**CLINC**			**WOS**		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
BERT (Devlin et al., 2019b)	33.52	17.04	56.90	34.37	17.61	64.75	31.97	18.36	45.15
BERT + CE	37.89	33.68	74.63	43.85	32.37	78.58	38.29	36.94	64.72
Llama2 (Touvron et al., 2023)	19.27±1.21	5.21±0.46	44.34±0.85	20.77±2.61	5.83±1.52	49.7±3.68	9.85±1.14	1.26±0.75	18.27±2.28
Llama2 + CE	32.40±5.46	17.32±5.95	57.53±5.78	45.69±6.85	29.38±6.55	72.66±7.13	18.51±1.50	7.8±1.18	29.66±3.23
GPT4 (Achiam et al., 2023)	10.77±1.86	0.14±0.05	35.17±3.68	9.56±2.12	0.11±0.06	46.69±3.24	7.56±1.51	0.15±0.04	27.78±2.98
DC (Caron et al., 2018)	18.05	43.34	29.74	26.40	12.51	61.26	29.17	13.98	53.27
DAC (Zhang et al., 2021)	29.14	12.89	52.99	29.16	14.15	62.78	28.47	15.94	43.52
DC + CE	41.73	27.81	66.81	30.28	13.56	62.38	38.76	35.21	60.30
DAC + CE	42.19	28.15	66.50	42.09	28.09	72.78	39.42	33.67	61.60
PPNet (Ji et al., 2020)	58.36±2.51	47.63±1.96	79.75±1.02	70.15±1.86	59.31±0.96	85.08±0.81	62.59±1.41	50.81±1.21	72.19±0.68
STAR-PPNet (ours)	63.19±2.38	52.21±1.33	81.66±1.21	73.21±1.97	61.87±0.79	86.16±0.47	66.15±1.33	53.61±1.24	73.82±0.74
Delete (Wu et al., 2020)	21.30	6.52	44.13	47.11	31.28	73.39	24.50	11.68	35.47
SimCSE (Gao et al., 2021)	24.48	8.42	46.94	40.22	23.57	69.02	25.87	13.03	38.53
Ancor + CE	32.90	30.71	74.73	44.44	31.50	74.67	39.34	26.14	54.35
NNCL (Dwibedi et al., 2021)	32.98	30.02	73.24	17.42	13.93	67.56	29.64	28.51	61.37
SimCSE + CE	34.04	31.81	74.86	52.53	37.03	77.39	41.28	34.47	61.62
Delete + CE	35.13	31.84	74.88	47.87	33.79	76.25	41.53	33.78	61.01
CLNN (Zhang et al., 2022)	37.21	34.66	75.27	19.96	14.76	68.30	29.48	28.42	60.99
Ancor (Bukchin et al., 2021)	37.34	34.75	74.99	45.60	33.11	75.23	41.20	37.00	65.42
SNCL (Chongjian et al., 2022)	42.32	38.17	76.39	55.01	45.64	82.93	36.27	33.62	62.35
WSCL (An et al., 2022)	59.52	49.34	79.31	74.02	62.98	88.37	65.27	51.78	72.46
DNA (An et al., 2023a)	70.81	59.66	83.31	87.66	81.82	94.69	74.57	63.30	76.86
STAR-DNA (ours)	75.79±0.93	65.27±1.12	85.34±0.36	89.25±0.17	83.47±0.27	95.11±0.05	77.19±0.81	64.97±0.75	77.91±0.76
DOWN (An et al., 2024)	78.92	68.17	86.22	91.79	86.70	96.05	80.00	67.09	78.87
STAR-DOWN (ours)	**80.31±0.26**	**70.22±0.59**	**87.28±0.31**	**92.45±0.38**	**87.05±0.17**	**96.20±0.07**	**81.98±0.67**	**69.27±0.60**	**79.99±0.40**

Table 2: The average performance (%) in terms of Accuracy (ACC), Adjusted Rand Index (ARI), and Normalized Mutual Information (NMI) on three datasets for the FCDC language task. To ensure fair comparisons with previous works (An et al., 2022, 2023a, 2024) and demonstrate the effectiveness of STAR, we use the same clustering inference mechanism and also average the results over three runs with identical common hyperparameters. Some baselines results are cited from aforementioned previous works, where standard deviations are not originally provided.

Methods	**HWU64**			**CLINC**			**WOS**		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
STAR-DOWN (clustering)	80.31±0.26	70.22±0.59	87.28±0.31	92.45±0.38	87.05±0.17	96.20±0.07	81.98±0.67	69.27±0.60	79.99±0.40
STAR-DOWN (centroid)	79.44±0.51	69.13±0.75	86.97±0.40	92.60±0.45	87.16±0.53	96.21±0.09	81.89±0.53	69.05±0.39	79.78±0.32

Table 3: Comparison of clustering and centroid inference mechanisms. "Clustering" clusters test set sample embeddings to determine each sample's fine-grained category, while "Centroid" infers the category by comparing each test sample's cosine similarity to fine-grained centroids.

2023; Achiam et al., 2023) (GPT4 prompt in Appendix A.6) perform poorly on the FCDC task due to the lack of fine-grained supervisory information. Self-training methods like DC, DAC, and PPNet (Caron et al., 2018; Zhang et al., 2021; Ji et al., 2020) also struggle because they rely on noisy fine-grained pseudo-labels and overlook comprehensive semantic similarities (CSS). Contrastive learning methods such as SNCL (Chongjian et al., 2022) and WSCL (An et al., 2022) perform better by leveraging positive pairs. DNA (An et al., 2023a) and DOWN (An et al., 2024) further enhance feature quality by filtering false positives and weighting them by rank. However, these methods still do not use CSS for sample distributions. Integrating the STAR method with existing baselines enhances performance across all datasets, consistently improving sample distributions in the Euclidean space.

The superior performance of STAR is attributed to three factors: First, bidirectional KL divergence measures CSS, pushing negative samples further away and relatively bringing positive samples closer based on CSS magnitude, making fine-grained clusters easier to distinguish. Second, the base B of the exponential in Eq. 4 is a trainable scalar, balancing CSS magnitude and semantic structure. Third, STAR variants iteratively bootstrap model performance in neighborhood retrieval and representation learning through a generalized EM process (detailed in Appendix A.2.3).

5.3 Inference Mechanism Comparison (RQ2)

Previous methods (Chongjian et al., 2022; An et al., 2023a, 2024) perform a nearest neighbor search over the examples of the found fine-grained clusters for fine-grained category prediction (we refer to this technique as cluster inference). We speed up this process making it better suitable for real-time tasks by developing a centroid inference mechanism (see Section 4.4). Results in Table 3 demon-

Methods	ACC	ARI	NMI
ours	80.31±0.26	70.22±0.59	87.28±0.31
w/o CE	78.61±0.44	67.32±0.86	85.62±0.36
w/o KL loss	78.97±0.32	68.03±0.36	85.81±0.16
w/o KL weight	79.26±0.42	68.86±0.37	86.21± 0.07
w/o KL weight and loss	78.96±0.15	68.21±0.22	86.32±0.10

Table 4: Results (%) of the ablation study for STAR-DOWN on the HWU64 Dataset.

Base value	ACC	ARI	NMI
trainable B (ours)	80.31±0.26	70.22±0.59	87.28±0.31
e	79.96±0.12	68.89±0.55	86.66±0.10
10	80.22±0.27	69.61±0.65	87.08±0.30
16	80.73±0.32	70.14±0.58	87.25±0.36
66	80.57±0.38	70.20±0.52	87.07±0.15

Table 5: Averaged results (%) and their standard deviations over three runs of multiple STAR-DOWN methods with **five different base values** on the HWU64 dataset. To set base value conveniently, we set B as a trainable scalar.

strates that results of centroid inference are competitive with cluster inference. When results are of the former are lower, this is due to two factors: clustering inference leverages inter-relations among test set samples for richer features, while centroid inference depends on centroids derived from noisy samples with fine-grained pseudo-labels. Despite these issues, centroid inference remains a viable option for real-time applications, balancing immediate analytical needs with slight performance trade-offs.

5.4 Ablation Study (RQ1 & RQ3)

We examine the impact of various components of the STAR method in STAR-DOWN, as detailed in Table 4. Our results yield the following insights. (**1**) Excluding coarse-grained supervision information during training (w/o CE) reduces model performance, as this information is crucial for effective representation learning. (**2**) Omitting the first loss term (w/o KL loss) from Eq. 4 diminishes performance. The KL loss term aligns the KL divergence between data samples and the query with their semantic similarities. Without it, $B^{d_{KL}(q_i,h_k)}$ fails to guide the query sample distribution based on semantic similarities in Eq. 4. (**3**) Removing the KL weight $B^{d_{KL}(q_i,h_k)}$ from Eq. 4 (w/o KL weight) reduces effectiveness. The loss no longer utilizes fine-grained semantic similarities measured by $B^{d_{KL}(q_i,h_k)}$ in the logarithmic space to direct the query sample distribution in comparison to all samples. (**4**) Eliminating both the KL loss term and the KL weight in Eq. 4 leads to a performance decline. This omission prevents the optimization of the query sample towards positive samples in the logarithmic space and fails to leverage fine-grained semantic similarities in the logarithmic space to influence the distribution of query samples relative to all samples in the Euclidean space.

5.5 Exponential Base Impact (RQ4)

In the STAR method's loss equation (Eq. 4), $B^{d_{KL}(q_i,h_k)}$ modulates the distribution of q_i and h_k in the Euclidean space based on their semantic similarity in the logarithmic space, as quantified by the bidirectional KL divergence. The base B is used to enhance semantic differences, improving the discriminability of fine-grained categories. We experimented with multiple constant values and a trainable configuration for B, with multiple STAR-DOWN results presented in Table 5. The multiple STAR-DOWN methods with various base values consistently outperform the DOWN method (Table 2), demonstrating the effectiveness and robustness of the STAR method regardless of the base value B. Notably, base values that are either too low (e.g., e) or too high (e.g., 66) disrupt the semantic representation by inadequately or excessively emphasizing semantic similarities in the logarithmic space. To set base value conveniently, we set B as a trainable scalar, achieving favorable outcomes as indicated in Table 5.

5.6 Inference of Category Semantics

Prior works (An et al., 2023a, 2024) only discovered fine-grained categories and assigned them numeric indices without elucidating the categories semantics, thus constraining their broader application. We propose utilizing the commonsense reasoning capabilities of large language models (**LLMs**) to infer the semantics of these categories. Specifically, we employ a trained encoder, F_θ, to extract embeddings from all train set samples and cluster these embeddings to assign fine-grained pseudo-labels to each train set sample. For each fine-grained category indicated by a specific pseudo-label, we aggregate all predicted samples from the training set and use an LLM to deduce the category semantics. Details on the LLM prompt are provided in Appendix A.7.

5.7 Error Analysis

As shown in the results of Table 6, considering this is an unsupervised experiment, the majority of fine-

Category	ACC	Category	ACC	Category	ACC	Category	ACC
alarm_query	63.16	datetime_convert	100.0	general_explain	53.85	iot_hue_lightoff	89.47
alarm_remove	72.73	datetime_query	57.89	general_joke	91.67	iot_hue_lighton	66.67
alarm_set	84.21	email_addcontact	100.0	general_negate	100.0	iot_hue_lightup	35.72
audio_volume_down	87.5	email_query	73.68	general_praise	100.0	iot_wemo_off	88.89
audio_volume_mute	80.0	email_querycontact	79.95	general_quirky	42.11	iot_wemo_on	85.72
audio_volume_up	76.92	email_sendemail	63.16	general_repeat	73.68	lists_createoradd	94.74
calendar_query	63.16	general_affirm	100.0	iot_cleaning	100.0	lists_query	84.21
calendar_remove	84.21	general_commandstop	100.0	iot_coffee	100.0	lists_remove	94.74
calendar_set	84.21	general_confirm	89.47	iot_hue_lightchange	73.68	music_likeness	88.89
cooking_recipe	89.47	general_dontcare	100.0	iot_hue_lightdim	58.33	music_query	63.16
music_settings	100.0	qa_maths	92.86	transport_taxi	100.0	news_query	78.95
qa_stock	100.0	transport_ticket	89.47	recommendation_events	78.95	transport_traffic	94.74
play_audiobook	89.47	recommendation_locations	100.0	play_game	89.47	play_music	89.47
qa_currency	100.0	takeaway_order	78.95	qa_definition	89.47	qa_factoid	52.63
recommendation_movies	100.0	weather_query	89.47	transport_query	78.95	social_post	73.68

Table 6: The error analysis analyzing the discovered fine-grained categories from STAR-DOWN method on the HWU64 dataset. The numerical values represent the classification accuracy (ACC) for each fine-grained category.

grained category samples, such as play_audiobook and qa_currency, are classified with reasonable accuracy, demonstrating the qualitative effectiveness of our unsupervised method, STAR-DOWN. However, certain fine-grained categories, such as datetime_query, exhibit lower classification performance compared to others. A possible reason is that queries often contain descriptive text, which can distract from correctly classifying the text into the intended query category. For example, the query "tell me what time it is in Dallas, Texas" falls under the datetime_query category, but its descriptive nature may lead to misclassification into location-related categories.

Additionally, some fine-grained categories have very nuanced semantic differences, making them particularly challenging for fine-grained discovery tasks. Examples include general_quirky, iot_hue_lightdim, iot_hue_lightup, qa_factoid and so on. For instance, the iot_hue_lightup category refers to increasing light brightness, which must be carefully distinguished from simply turning the light on.

5.8 Visualization

We visualize the sample embeddings of STAR-DOWN in Figure 3. The results demonstrate that our method forms distinguishable clusters for fine-grained categories, proving STAR's effectiveness in separating dissimilar samples and clustering similar ones. Additionally, we visualize the generalized EM perspective of STAR-DOWN in Appendix A.1.6.

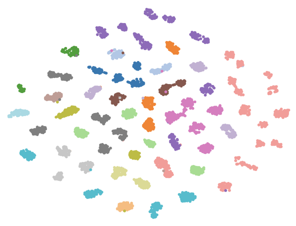


Figure 3: The t-SNE visualization of sample embeddings from STAR-DOWN method on the HWU64 dataset, with different colors representing different coarse-grained categories. The distinct clusters represent the discovered fine-grained categories.

6 Conclusion

We propose the STAR method for fine-grained category discovery in natural language texts, which utilizes comprehensive semantic similarities in the logarithmic space to guide the distribution of textual samples, including conversational intents, scientific paper abstracts, and assistant queries, in the Euclidean space. STAR pushes query samples further away from negative samples and brings them closer to positive samples based on the comprehensive semantic similarities magnitude. This process forms compact clusters, each representing a discovered category. We theoretically analyze the effectiveness of STAR method. Additionally, we introduce a centroid inference mechanism that addresses previous gaps in real-time evaluations. Experiments on three natural language benchmarks demonstrate that STAR achieves new state-of-the-art performance in fine-grained category discovery tasks for text classification.

Limitations

Although the proposed STAR method, integrated with existing contrastive learning methods, achieves superior performance in fine-grained category discovery tasks, its variants require additional memory to store a data queue for neighbors retrieval and feature learning.

Ethical Consideration

Our study introduces a novel method that leverages comprehensive semantic similarities to improve the distinction of fine-grained clusters in fine-grained category discovery tasks. This contribution has no direct negative social impacts.

Acknowledgements

We would like to thank all our families and friends for their support throughout this work. This research was partially supported by the China Scholarship Council. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors. Marie-Francine Moens is supported by the ERC Advanced Grant CALCULUS (788506).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Wenbin An, Feng Tian, Ping Chen, Siliang Tang, Qinghua Zheng, and Qianying Wang. 2022. Fine-grained category discovery under coarse-grained supervision with hierarchical weighted self-contrastive learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1323.

Wenbin An, Feng Tian, Wenkai Shi, Yan Chen, Qinghua Zheng, Qianying Wang, and Ping Chen. 2023a. Dna: Denoised neighborhood aggregation for fine-grained category discovery. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12292–12302.

Wenbin An, Feng Tian, Wenkai Shi, Haonan Lin, Yaqiang Wu, Mingxiang Cai, Luyan Wang, Hua Wen, Lei Yao, and Ping Chen. 2024. Down: Dynamic order weighted network for fine-grained category discovery. *Knowledge-Based Systems*, 293:111666.

Wenbin An, Feng Tian, Qinghua Zheng, Wei Ding, QianYing Wang, and Ping Chen. 2023b. Generalized category discovery with decoupled prototypical network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12527–12535.

Rinu Boney and Alexander Ilin. 2017. Semi-supervised and active few-shot learning with prototypical networks. *arXiv preprint arXiv:1711.10856*.

Guy Bukchin, Eli Schwartz, Kate Saenko, Ori Shahar, Rogerio Feris, Raja Giryes, and Leonid Karlinsky. 2021. Fine-grained angular contrastive learning with coarse labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8730–8740.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149.

Tianshui Chen, Liang Lin, Riquan Chen, Yang Wu, and Xiaonan Luo. 2018. Knowledge-embedded representation learning for fine-grained image recognition. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 627–634.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Ze Chen, Wanting Ji, Linlin Ding, and Baoyan Song. 2023. Fine-grained document-level financial event argument extraction approach. *Engineering Applications of Artificial Intelligence*, 121:105943.

GE Chongjian, Jiangliu Wang, Zhan Tong, Shoufa Chen, Yibing Song, and Ping Luo. 2022. Soft neighbors are positive supporters in contrastive visual representation learning. In *The Eleventh International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. 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.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. 2021. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Rayid Ghani and Andrew Fano. 2002. Building recommender systems using a knowledge base of product semantics. In *Proceedings of the Workshop on Recommendation and Personalization in ECommerce at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web based Systems*, pages 27–29. Citeseer.

Xiaoting Guo, Wei Yu, and Xiaodong Wang. 2021. An overview on fine-grained text sentiment analysis: Survey and challenges. In *Journal of Physics: Conference Series*, volume 1757, page 012038. IOP Publishing.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2:193–218.

Zilong Ji, Xiaolong Zou, Tiejun Huang, and Si Wu. 2020. Unsupervised few-shot feature learning via self-supervised training. *Frontiers in computational neuroscience*, 14:83.

Kamran Kowsari, Donald E Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S Gerber, and Laura E Barnes. 2017. Hdltex: Hierarchical deep learning for text classification. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 364–371. IEEE.

Harold W Kuhn. 2010. The hungarian method for the assignment problem. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 29–47.

Andrea Lancichinetti, Santo Fortunato, and János Kertész. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New journal of physics*, 11(3):033015.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 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.

Dan Li, Shuai Wang, Jie Zou, Chang Tian, Elisha Nieuwburg, Fengyuan Sun, and Evangelos Kanoulas. 2021. Paint4poem: A dataset for artistic visualization of classical chinese poems. *arXiv preprint arXiv:2109.11682*.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. 2020. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*.

Ruixue Lian, William A Sethares, and Junjie Hu. 2024. Learning label hierarchy with supervised contrastive learning. *arXiv preprint arXiv:2402.00232*.

Mohamed Lichouri, Khaled Lounnas, and Mohamed Zakaria Amziane. 2024. dzfinnlp at arafinnlp: Improving intent detection in financial conversational agents. *arXiv preprint arXiv:2407.13565*.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2021. Benchmarking natural language understanding services for building conversational agents. In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction: 10th International Workshop on Spoken Dialogue Systems*, pages 165–183. Springer.

Ruotian Ma, Zhang Lin, Xuanting Chen, Xin Zhou, Junzhe Wang, Tao Gui, Qi Zhang, Xiang Gao, and Yun Wen Chen. 2023. Coarse-to-fine few-shot learning for named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4115–4129.

Dheeraj Mekala, Varun Gangal, and Jingbo Shang. 2021. Coarse2fine: Fine-grained text classification on coarsely-grained annotated data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 583–594.

Zhengxin Pan, Fangyu Wu, and Bailing Zhang. 2023. Fine-grained image-text matching by cross-modal hard aligning network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19275–19284.

Viral Parekh, Karimulla Shaik, Soma Biswas, and Muthusamy Chelliah. 2021. Fine-grained visual attribute extraction from fashion wear. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3973–3977.

Wongi Park and Jongbin Ryu. 2024. Fine-grained self-supervised learning with jigsaw puzzles for medical image classification. *Computers in Biology and Medicine*, page 108460.

Chang Tian, Wenpeng Yin, Dan Li, and Marie-Francine Moens. 2024. Fighting against the repetitive training and sample dependency problem in few-shot named entity recognition. *IEEE Access*.

Chang Tian, Wenpeng Yin, and Marie-Francine Moens. 2022. Anti-overestimation dialogue policy learning for task-completion dialogue system. *arXiv preprint arXiv:2207.11762*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Sagar Vaze, Andrea Vedaldi, and Andrew Zisserman. 2024. No representation rules them all in category discovery. *Advances in Neural Information Processing Systems*, 36.

Jan Vellmer, Peter Mandl, Tobias Bellmann, Maximilian Balluff, Manuel Weber, Alexander Döschl, and Max-Emanuel Keller. 2023. A machine learning approach to enterprise matchmaking using multilabel text classification based on semi-structured website content. In *International Conference on Information Integration and Web Intelligence*, pages 493–509. Springer.

Shijie Wang, Jianlong Chang, Zhihui Wang, Haojie Li, Wanli Ouyang, and Qi Tian. 2024a. Content-aware rectified activation for zero-shot fine-grained image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Shijie Wang, Zhihui Wang, Haojie Li, Jianlong Chang, Wanli Ouyang, and Qi Tian. 2024b. Accurate fine-grained object recognition with structure-driven relation graph networks. *International Journal of Computer Vision*, 132(1):137–160.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466*.

Hanlei Zhang, Hua Xu, Ting-En Lin, and Rui Lyu. 2021. Discovering new intents with deep aligned clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14365–14373.

Yuwei Zhang, Haode Zhang, Li-Ming Zhan, Xiao-Ming Wu, and Albert Lam. 2022. New intent discovery with pre-training and contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 256–269.

Zhun Zhong, Enrico Fini, Subhankar Roy, Zhiming Luo, Elisa Ricci, and Nicu Sebe. 2021. Neighborhood contrastive learning for novel class discovery. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10867–10875.

A Appendix

A.1 STAR-DOWN

A.1.1 DOWN Pre-train and Neighbors Retrieval

DOWN (An et al., 2024) pre-trains the Encoder to incorporate coarse-grained and general knowledge. The pre-training loss L_{pre} is defined as the sum of the cross-entropy loss L_{ce} and the masked language modeling loss L_{mlm}:

$$L_{\text{pre}} = L_{\text{ce}} + L_{\text{mlm}}, \tag{8}$$

given that D_{train} is the train set and Y_{coarse} represents the coarse-grained labels for the training set, let $N_{train} \in D_{train}$ be a training batch with coarse labels Y_c.

The cross-entropy loss for a single training batch is calculated as the average loss over all samples in the batch. Here, $\hat{y}_i$ denotes the predicted probability distribution for the i-th sample in the batch, and y_i represents the ground truth probability distribution for the i-th sample in the batch. The cross-entropy loss L_{ce} for the training batch N_{train} is given by:

$$L_{\text{ce}} = -\frac{1}{|N_{train}|}\sum_{i=1}^{|N_{train}|}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}), \tag{9}$$

in this context: C is the number of categories, $y_{i,c}$ is a binary indicator (0 or 1) indicating whether category label c is the correct classification for sample i, $\hat{y}_{i,c}$ is the predicted probability for category c for sample i.

The masked language modeling (mlm) loss L_{mlm} for the training batch N_{train} is expressed as the average negative log-likelihood of the true token given the masked context for each token in each sample in the batch:

$$L_{\text{mlm}} = -\frac{1}{|N_{train}|}\sum_{i=1}^{|N_{train}|}\frac{1}{T}\sum_{j=1}^{T}\log p(\hat{x}_{i,j} \mid x_{i,j}^{\text{masked}}). \tag{10}$$

In this equation: N_{train} is a training batch. T is the length of each training sequence. $\hat{x}_{i,j}$ is the predicted token. $x_{i,j}^{\text{masked}}$ is the masked token. $p(\hat{x}_{i,j} \mid x_{i,j}^{\text{masked}})$ is the predicted probability of the true token $\hat{x}_{i,j}$ given the masked input $x_{i,j}^{\text{masked}}$.

A.1.2 KL Divergence

In this section, we calculate the bidirectional KL divergence between the query sample embedding q_i and positive or negative sample embedding h_k with the function $d_{KL}(q_i, h_k)$.

We assume that sample embeddings follow the Gaussian distributions. Specifically, we use projection networks f_μ and f_Σ to produce Gaussian distribution parameters:

$$\mu_i = f_\mu(q_i), \quad \Sigma_i = \text{ELU}(f_\Sigma(q_i)) + (1+\epsilon),$$

$$\mu_k = f_\mu(h_k), \quad \Sigma_k = \text{ELU}(f_\Sigma(h_k)) + (1+\epsilon),$$

where $\mu_i, \mu_k \in \mathbb{R}^l$ and $\Sigma_i, \Sigma_k \in \mathbb{R}^{l\times l}$ represent the mean and diagonal covariance of the Gaussian embeddings, respectively. The covariances have nonzero elements only along the diagonal. The functions f_μ and f_Σ are implemented as ReLU followed by single-layer networks. ELU (exponential linear unit) ensures numerical stability, with $\epsilon \approx e^{-14}$. Here, l, the Gaussian embedding dimension, is 128.

Given the Gaussian distribution parameters $\mu_i, \mu_k \in \mathbb{R}^l$ and $\Sigma_i, \Sigma_k \in \mathbb{R}^{l\times l}$, we define the corresponding Gaussian distributions $\mathcal{N}_i = \mathcal{N}(\mu_i, \Sigma_i)$ and $\mathcal{N}_k = \mathcal{N}(\mu_k, \Sigma_k)$ for the query sample embedding q_i and the positive or negative sample embedding h_k.

The bidirectional KL divergence between the query sample embedding q_i and the positive or negative sample embedding h_k is calculated using the function $d_{KL}(q_i, h_k)$, which measures fine-grained semantic similarities:

$$d_{KL}(q_i, h_k) = \frac{1}{2}\left(D_{\text{KL}}[\mathcal{N}_i \| \mathcal{N}_k] + D_{\text{KL}}[\mathcal{N}_k \| \mathcal{N}_i]\right), \tag{11}$$

where

$$\begin{aligned} & D_{\text{KL}}[\mathcal{N}_i \| \mathcal{N}_k] = D_{\text{KL}}[\mathcal{N}(\mu_i, \Sigma_i) \| \mathcal{N}(\mu_k, \Sigma_k)] \\ & = \frac{1}{2}(\text{Tr}(\Sigma_k^{-1}\Sigma_i) + (\mu_k - \mu_i)^T \Sigma_k^{-1}(\mu_k - \mu_i) \\ & - l + \log \frac{|\Sigma_k|}{|\Sigma_i|}). \end{aligned} \tag{12}$$

A.1.3 Form Similarity

$$F = G\frac{m_1 m_2}{r^2}. \tag{13}$$

In astronomy, gravitational force is crucial for determining the orbits of celestial bodies. According to the formula, the mass of the bodies significantly influences their orbital paths.

Inspired by astronomy, we use fine-grained comprehensive semantic similarities between the query sample and each available positive or negative sample to guide sample distributions in the embedding space, where the similarities are measured by bidirectional KL divergence, as shown in Eq. 14:

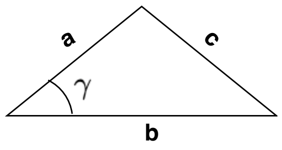


Figure 4: Cosine rule. a, b, and c denote the lengths of the triangle's sides, and γ represents the angle. The cosine rule is used in the Appendix A.1.4.

$$\begin{aligned} B^{d_{KL}(q_i,h_k)} &= \left(e^{\log(B)}\right)^{d_{KL}(q_i,h_k)} \\ &= e^{\log(B)\cdot d_{KL}(q_i,h_k)} \\ &= e^{\log(B)\cdot(\log M + \log m)} \\ &= e^{\log(B)\cdot \log(Mm)} \\ &= e^{B'\cdot \log(Mm)}. \end{aligned} \tag{14}$$

Since bidirectional KL divergence is asymmetrical, it consists of two components: $\log M$ and $\log m$.

In the STAR method, the bidirectional KL divergence $B^{d_{KL}(q_i,h_k)}$ is a key component of the loss function (Eq. 4) that guides sample distributions. It can be decomposed into two divergence components, $\log(Mm)$, which is analogous to the gravitational force term $m_1 m_2$.

A.1.4 Cosine Similarity Conversion

In trigonometry, cosine rule relates the lengths of the sides of a triangle to the cosine of one of its angles. In Figure 4, for a triangle with sides a, b, and c, with γ being the angle opposite side c, the law of cosines is expressed as:

$$c^2 = a^2 + b^2 - 2ab\cos\gamma \tag{15}$$

In this context, we use $\cos\gamma$ to represent the cosine similarity between q_i and h_k, and c to denote the Euclidean distance between the query sample q_i and the positive or negative sample h_k. Given that the sample embeddings are normalized to have a length of 1, we have the following relationship:

$$c^2 = 1 + 1 - 2\cos\gamma,$$

thus,

$$c^2 \overset{c}{=} -\cos\gamma.$$

Here, $q_i^{\text{T}} h_k$ quantifies the cosine similarity between q_i and h_k, so

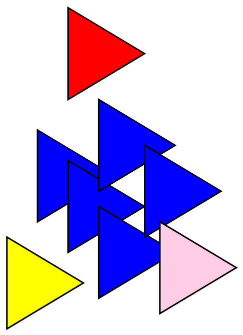


Figure 5: The centroid inference mechanism. Triangles represent samples with the same fine-grained pseudo label; different colors denote various coarse-grained labels. Only samples with predominant coarse-grained labels, represented by blue triangles, are used to approximate fine-grained centroids; all others are excluded.

$$c^2 \overset{c}{=} -q_i^{\mathrm{T}} h_k.$$

The cosine similarity and the negative Euclidean distance represents the similar mathematical meaning. Therefore, a smaller Euclidean distance between two samples corresponds to a larger cosine similarity. The STAR-DOWN procedure is outlined in Algorithm 1. In Step 3, STAR-DOWN introduces a novel contrastive loss, as specified in Eq. 5. To ensure fair validation of its effectiveness, STAR-DOWN adheres to Steps 1 and 2 of the DOWN procedure.

A.1.5 Centroid Inference

As shown in Figure 5, we introduce centroid inference, an alternative inference suitable for real-time and other contexts. Using F_θ, we derive sample embeddings from D_{train} and assign fine-grained pseudo-labels through clustering. For each fine-grained cluster, only the embeddings of samples from the predominant coarse-grained category are averaged to approximate centroid representations. These approximated centroids are then used to determine the fine-grained category of each test sample based on cosine similarity.

A.1.6 Visualization

To verify the generalized EM perspective of STAR-DOWN, we visualize the true neighbor rate and model performance curves using three metrics during the training process, as shown in Figures 6a and 6b. The results indicate that STAR-DOWN progressively retrieves more accurate neighbors and improves model performance across the three metrics throughout the training. This improvement is due to the positive feedback loop where more accurate neighbor retrieval enhances feature learning, and enhanced feature learning, in turn, leads to more accurate neighbor retrieval. Thus, STAR-DOWN effectively estimates true neighbors in the E-step and obtains better representations in the M-step, with two steps alternately performed to gradually enhance each other.

A.1.7 Algorithm Procedure

The STAR-DOWN procedure is outlined in Algorithm 1. In Step 3, STAR-DOWN introduces a novel contrastive loss, as specified in Eq. 5. To ensure fair validation of its effectiveness, STAR-DOWN adheres to Steps 1 and 2 of the DOWN procedure.

A.2 STAR-DOWN Analyses

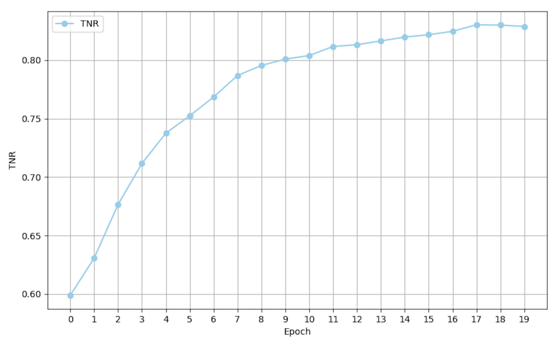
The STAR method upgrades the original contrastive loss L_1^i in L_{train} to the new loss L_2^i as shown in Eq. 16. The first term in L_2^i optimizes the method in the logarithmic space, increasing the KL divergence magnitude in accordance with semantic differences. The second term, L_{2-2}^i, optimizes sample distributions in the Euclidean space. Since fine-grained category discovery occurs in the Euclidean space, our analyzes focus on L_{2-2}^i, which optimizes the distributions of query samples within the Euclidean space:

$$\begin{aligned} L_2^i = &-\gamma \sum_{h_j \in N_i} \omega_j \cdot \log \frac{\exp(-d_{KL}(q_i, h_j)/\tau)}{\sum_{h_k \in Q} \exp(-d_{KL}(q_i, h_k)/\tau)} \\ &- \sum_{h_j \in N_i} \omega_j \cdot \log \frac{\exp(q_i^{\mathrm{T}} h_j/\tau)}{\sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau)}, \end{aligned} \tag{16}$$

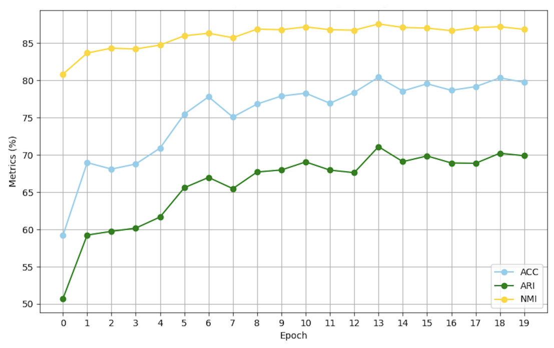
$$\begin{aligned} L_{2-2}^i = &- \sum_{h_j \in N_i} \omega_j \cdot \log \frac{\exp(q_i^{\mathrm{T}} h_j/\tau)}{\sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau)} \\ = &\sum_{h_j \in N_i} \omega_j \cdot (\log \sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau) \\ &- (q_i^{\mathrm{T}} h_j/\tau)). \end{aligned} \tag{17}$$

A.2.1 Gradient Analysis

From the **gradient** perspective, the gradient optimizes the method to project samples into the Euclidean space, where samples form into fine-grained clusters and each cluster represents a discovered fine-grained category. Each query sample q_i has multiple positive samples $h_j \in N_i$. To simplify the understanding of the STAR method's



(a) True Neighbor Rate (TNR) during training.



(b) Model performance during training.

Figure 6: The validation of generalized EM perspective on the HWU64 dataset.

gradient, we focus on the loss component related to a specific h_j from the loss L^i_{2-2}. This gives us:

$$\begin{aligned} L^{ij}_{2-2} =& -\omega_j \cdot \log \frac{\exp(q_i^{\mathrm{T}} h_j/\tau)}{\sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau)} \\ =& \omega_j \cdot (\log \sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau) \\ & - (q_i^{\mathrm{T}} h_j/\tau)). \end{aligned} \tag{18}$$

For the query sample q_i, we investigate the gradient related to the negative sample $h_n \in Q$. With the loss L^{ij}_{2-2}, the gradient becomes:

$$\frac{\partial L^{ij}_{2-2}}{\partial [q_i^{\mathrm{T}} h_n]} = \frac{\omega_j}{\tau} \cdot \frac{B^{d_{KL}(q_i,h_n)} \cdot \exp(q_i^{\mathrm{T}} h_n/\tau)}{\sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau)}. \tag{19}$$

The larger the semantic difference in $B^{d_{KL}(q_i,h_n)}$, the larger the gradient with regards to $q_i^{\mathrm{T}} h_n$, the more $q_i^{\mathrm{T}} h_n$ will decrease to push q_i away from h_n.

For the gradient relevant to the positive sample $h_j \in N_i$, with the upgraded loss L^{ij}_{2-2}:

$$\frac{\partial L^{ij}_{2-2}}{\partial [q_i^{\mathrm{T}} h_j]} = -\frac{\omega_j}{\tau} \cdot (1 - \frac{B^{d_{KL}(q_i,h_j)} \cdot \exp(q_i^{\mathrm{T}} h_j/\tau)}{\sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau)}). \tag{20}$$

A smaller semantic difference in $B^{d_{KL}(q_i,h_j)}$ results in a larger gradient magnitude with respect to $q_i^{\mathrm{T}} h_j$, thereby increasing $q_i^{\mathrm{T}} h_j$ and bringing q_i closer to h_j.

Overall, the gradient optimizes sample distributions in the Euclidean space by leveraging comprehensive semantic similarities in the logarithmic space. Large semantic differences (low semantic similarities) between the query sample and an available sample push the query sample further away in the Euclidean space, while small semantic differences (high semantic similarities) bring the query sample closer. Consequently, samples form distinguishable fine-grained clusters in the Euclidean space, with each cluster representing a discovered category.

A.2.2 Clustering Perspective Interpretation

From the **clustering** perspective, the loss L^i_{2-2} can be written as:

$$\begin{aligned} L^i_{2-2} =& -\sum_{h_j \in N_i} \omega_j \cdot \log \frac{\exp(q_i^{\mathrm{T}} h_j/\tau)}{\sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau)} \\ =& \sum_{h_j \in N_i} \omega_j \cdot \log \sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau) \\ & - \sum_{h_j \in N_i} \omega_j \cdot (q_i^{\mathrm{T}} h_j/\tau) \\ =& \sum_{h_j \in N_i} \omega_j \cdot \log \sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau) \\ & - \frac{1}{\tau} q_i^{\mathrm{T}} (\sum_{h_j \in N_i} \omega_j \cdot h_j) \\ =& \sum_{h_j \in N_i} \omega_j \cdot \log \sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau) \\ & - q_i^{\mathrm{T}} c_i/\tau \\ =& \sum_{h_j \in N_i} \omega_j \cdot \log \sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau) \\ & + \frac{1}{2\tau} \{(q_i - c_i)^2 - \|q_i\|^2 - \|c_i\|^2\} \\ \overset{c}{=}& \sum_{h_j \in N_i} \omega_j \cdot \log \sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau) \\ & + \frac{1}{2\tau} (q_i - c_i)^2, \end{aligned} \tag{21}$$

where $c_i = \sum_{h_j \in N_i} \omega_j \cdot h_j$ is the weighted average of query q_i neighbors' embeddings, $\overset{c}{=}$ indicates equal up to a multiplicative and/or an additive constant. $\|q_i\|^2 = 1$ because of normalization, and $\|c_i\|^2$ is a constant since the neighbor embedding h_j is from the dynamic queue without gradient.

In Eq. 21, the loss term L^i_{2-2} indicates that, from a clustering perspective, query samples will cluster around the neighbor centroids. These query samples will be distributed in the Euclidean space based on the comprehensive semantic similarities in the logarithmic space between the query sample and each available positive or negative sample, as measured by $B^{d_{KL}(q_i,h_k)}$, effectively distancing dissimilar samples, so that samples could form distinguishable clusters and each cluster represents a discovered fine-grained category.

A.2.3 Generalized EM Perspective Interpretation

From the **generalized EM** perspective, If we treat the centers of the weighted neighbors $C = \{c_i\}_{i=1}^{N}$ as hidden variables, we can interpret our model from the Expectation Maximization (EM) perspective following (An et al., 2023a).

At the E-step, we estimate the hidden variables by retrieving neighbors and weighting the neighbors' embeddings.

$$\{c_i|\theta, q_i, Q\}_{i=1}^{N} = \sum_{h_j \in N_i} \omega_j \cdot h_j, \tag{22}$$

where θ represents the parameters of Encoder, Q is the data queue, N_i is the positive samples set of the query sample q_i.

At the M-step, we optimize the Encoder parameters θ:

$$\begin{aligned} &\arg\min_{\theta} \sum_{q_i \in N_{train}} \left(\frac{1}{2\tau}(q_i - c_i)^2\right. \\ &+ \sum_{h_j \in N_i} \omega_j \cdot \log \sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(q_i^{\mathrm{T}} h_k/\tau)), \end{aligned} \tag{23}$$

where N_{train} is the training batch, and h_k is the positive or negative sample from data queue Q.

Accurate neighbors enhance representation learning, which in turn facilitates the retrieval of more accurate neighbors. This iterative process enables STAR-DOWN to progressively improve performances in both representation learning and neighborhood retrieval. Detailed empirical results are presented in Appendix A.1.6.

A.3 STAR Method Variants

A.3.1 STAR-DNA

DNA (An et al., 2023a) is a promising contrastive learning based method to solve the fine-grained category discovery task. DNA has three steps.

Step 1: pre-training:

$$L_{\text{pre}} = L_{\text{ce}}. \tag{24}$$

DNA pre-trains the Encoder F_θ to learn the coarse-grained information with Eq. 24.

Step 2: neighbors retrieval and refinement:

DNA retrieves positive samples from the data queue Q for each query sample q_i and applies three principles to eliminate potential false-positive neighbors: Label Constraint, Reciprocal Constraint, and Rank Statistic Constraint. The resulting positive set is S_i.

Step 3: training:

DNA trains the model parameters with the following loss:

$$L = -\frac{1}{|D|}\sum_{q_i \in D}\frac{1}{|S_i|}\sum_{h_j \in S_i}\log\frac{\exp(q_i^{\mathrm{T}} h_j/\tau)}{\sum_{h_k \in Q}\exp(q_i^{\mathrm{T}} h_k/\tau)}. \tag{25}$$

D denotes the training batch, S_i represents the positive set for q_i, and τ is the temperature parameter.

DNA iteratively performs steps 2 and 3 to enhance model performance. However, DNA does not utilize fine-grained semantic similarities to guide the distributions of query samples. To ensure a fair comparison, STAR-DNA upgrades the training loss in step 3 while following the same initial two steps as DNA.

STAR-DNA introduces a new loss function in step 3 to discover fine-grained semantic similarities:

$$\begin{aligned} L = &-\gamma\frac{1}{|D|}\sum_{q_i \in D}\frac{1}{|S_i|}\sum_{h_j \in S_i}\log\frac{\exp(\frac{-d_{KL}(q_i,h_j)}{\tau})}{\sum_{h_k \in Q}\exp(\frac{-d_{KL}(q_i,h_k)}{\tau})} \\ &-\frac{1}{|D|}\sum_{q_i \in D}\frac{1}{|S_i|}\sum_{h_j \in S_i}\log\frac{\exp(\frac{q_i^{\mathrm{T}} h_j}{\tau})}{\sum_{h_k \in Q} B^{d_{KL}(q_i,h_k)} \cdot \exp(\frac{q_i^{\mathrm{T}} h_k}{\tau})}. \end{aligned} \tag{26}$$

A.3.2 STAR-PPNet

The Prototypical Network with pseudo-labels (Boney and Ilin, 2017; Ji et al., 2020) (PPNet) is a widely used approach for fine-grained unsupervised classification tasks. PPNet typically involves two steps: in step 1, it employs a clustering algorithm to assign pseudo fine-grained labels to each query sample q_i in the train set D_{train}. In step 2, it trains using these pseudo-labels with loss Eq. 27 to cluster query samples with the same fine-grained pseudo label, thereby discovering fine-grained categories:

$$L = -\frac{1}{|N_{train}|} \sum_{i=1}^{|N_{train}|} L_i, \tag{27}$$

$$L_i = \log \frac{\exp(-d(q_i, \mathbf{p}_c))}{\sum_{c'=1}^{C} \exp(-d(q_i, \mathbf{p}_{c'}))}. \tag{28}$$

N_{train} is the training batch, C is the number of fine-grained pseudo-categories, q_i is the query sample embedding from the training set D_{train}, $d(\cdot, \cdot)$ is typically the Euclidean distance, and $\mathbf{p}_c$ is the prototype embedding of category c, computed as the mean of the embeddings of the support set examples of category c, $\mathbf{p}_{c'}$ is the prototype embedding of category c', c' is a category among C fine-grained categories:

$$\mathbf{p}_c = \frac{1}{N_c} \sum_{i=1}^{N_c} q_i^c, \tag{29}$$

where N_c is the number of samples in category c and q_i^c is the embedding of query sample with the pseudo fine-grained label of category c.

STAR-PPNET incorporates the fine-grained semantic similarities between the query sample and cluster centroids into the loss:

$$L = -\frac{1}{|N_{train}|} \sum_{i=1}^{|N_{train}|} L^i, \tag{30}$$

$$\begin{aligned} L^i = {} & \log \frac{\exp(-d(q_i, \mathbf{p}_c))}{\sum_{c'=1}^{C} B^{d_{KL}(q_i, \mathbf{p}'_c)} \cdot \exp(-d(q_i, \mathbf{p}_{c'}))} \\ & + \gamma \log \frac{\exp(-d_{KL}(q_i, \mathbf{p}_c))}{\sum_{c'=1}^{C} \exp(-d_{KL}(q_i, \mathbf{p}_{c'}))}, \end{aligned} \tag{31}$$

A.4 Implementation Details

For comparison, we use the same BERT model, bert-base-uncased, for feature extraction as in the original baseline papers. We employ GPT4 (version gpt-4-0125-preview) and Llama2 with 7B parameters. We fine-tune Llama2 with the LoRA technique, where the LoRA rank is 8, and the LoRA α is 32. The sample feature dimension in the embedding space is 768, and for calculating KL divergence, it is 128. The number of neighbors k is set to {120, 120, 250} for the CLINC, HWU64, and WOS datasets, respectively. We use random seeds {0, 1, 2}. The dimension for Rank Statistic Constraint in the DNA baseline is set to 5. The PyTorch version is 1.11.0.

A.5 Evaluation Metrics

The formula of ARI (Hubert and Arabie, 1985) is:

$$ARI = \frac{RI - E(RI)}{max(RI) - E(RI)}, \tag{32}$$

where RI is the rand index and the $E(RI)$ is the expectation of RI. Given a test set with n samples, a sample pair is simply any two distinct samples chosen from the test set. a: Number of pairs in the same cluster in both predicted labels and ground truth labels. b: Number of pairs in different clusters in both predicted labels and ground truth labels. c: Number of pairs in the same cluster in predicted labels but different in ground truth labels. d: Number of pairs in different clusters in predicted labels but same in ground truth labels.

$$RI = \frac{a+b}{a+b+c+d}. \tag{33}$$

The formula of Normalized Mutual Information (NMI) (Lancichinetti et al., 2009) is:

$$NMI = \frac{2 * I(\hat{y}; y)}{H(\hat{y}) + H(y)}, \tag{34}$$

where $\hat{y}$ is the prediction from clustering and y is the ground truth. $I(\hat{y}; y)$ is the mutual information between $\hat{y}$ and y, $H(\hat{y})$ and $H(y)$ represent the entropy of $\hat{y}$ and y, respectively.

ACC (Kuhn, 2010; An et al., 2023a) is the metric to evaluate the clustering accuracy:

$$ACC = \frac{\sum_{i=1}^{N} \mathbb{I}\{\mathcal{P}(\hat{y}_i) = y_i\}}{N}, \tag{35}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function, it returns 1 if the condition inside the braces is true, and 0 otherwise. In this formula, it checks whether the predicted label $\mathcal{P}(\hat{y}_i)$ matches the true label y_i. $\hat{y}_i$ is the prediction from clustering and y_i is the ground-truth label, N is the number of samples, and $\mathcal{P}(\cdot)$ is the permutation map function from the Hungarian algorithm (Kuhn, 2010).

A.6 Baseline GPT4 Prompt

The following prompt is designed for GPT4 using the HWU64 dataset. To adapt it for other datasets, simply adjust the number of coarse-grained classes and the index of fine-grained classes accordingly.

Our dataset comprises two tab-separated value (TSV) files:

- `coarse_label_GPT4.tsv`: This file contains text content and corresponding coarse-grained class labels for each line. The training set encompasses a total of 18 distinct coarse-grained classes.

- `test_GPT4.tsv`: This file consists of multiple lines, each containing a single text entry.

The objective is to assign fine-grained class labels to the text content in the `test_GPT4.tsv` file. Each fine-grained class will be represented by a unique integer identifier ranging from 0 to 63. Each fine-grained class is a sub-class of the existing coarse-grained classes defined in `coarse_label_GPT4.tsv`.

Please complete the following tasks according to your understanding:

1. **Prediction**: Assign a fine-grained class label (integer from 0 to 63) to each text content in the `test_GPT4.tsv` file.
2. **Output**: Print the predicted labels in order, corresponding to the order of text entries in the test file. Collect all the predicted labels in a list.
3. **Explanation**: Provide a semantic explanation (meaning description) for each predicted fine-grained class label.

A.7 Category Semantics Inference Prompt

In order to infer the discovered fine-grained category semantics, we leverage the commonsense reasoning capabilities of GPT4.

The following prompt is employed:

Can you please check what i last said. Could you please check and confirm my last command. I like you check question please. Did you get what just said correctly? I like you check it please. Did you get my last instruction correctly? Please can you check and confirm what just said. Would you check the details please. ...... Lower the lights in the house. Please can you confirm the command. Could you please check and confirm the detail. Would you check and confirm question please. Please confirm details. Will you please check and confirm it. Please would you check instruction. Would you please check question. Could you please check the detail. Could you please check and confirm details. Check the last thing said please. What you just said was not so clear to me. I would like you check and confirm it please. Did you get what said correctly? I like you to check and confirm what i said please.

Hello. The provided sentence samples originate from the domain of assistant query classification. Predominantly, these samples fit into a specific category, though a minority are noisy and do not align with the primary category. Could you please identify and name this category, and clarify its definition?

The ground truth semantics of this discovered fine-grained category is: *general-confirm* in the assistant query domain.

The outputs from GPT4 are as follows:

The primary category for the majority of the sentence samples you've provided seems to be "confirmation requests." This category comprises sentences where the speaker asks the assistant to confirm if a command or a query was understood or executed correctly. The requests often seek verification or validation of an action that was presumably communicated earlier, either explicitly or implicitly.

Definition of Confirmation Requests: Confirmation requests are communications aimed at verifying whether an earlier instruction, command, or statement was received, understood, and/or processed correctly. This category is common in user-assistant interactions where clarity and confirmation of understanding are critical to ensure accurate task execution.

Clearly, this method offers a practical and heuristic approach to deducing category semantics.

Algorithm 1: STAR-DOWN

```
Initialize pre-training epoch $P$, training epoch $E$,
Encoder $F_\theta$, Momentum Encoder $F_{\theta_k}$, and train set $D_{train}$
Step 1: pre-training
for epoch =0,...,P-1 do
    Initialize pre-training batch size
    for $N_{pre} \in D_{train}$ do
        The Encoder $F_\theta$ is trained with pre-training batch $N_{pre}$ using loss $L_{\text{pre}}$ (Eq. 8)
        Update Encoder parameters $\theta$
    end
end
for epoch =0,...,E-1 do
    Initialize nearest neighbors number $K$
    Step 2: neighbors retrieval & weighting
    Construct the gradient-free data queue: $Q = F_{\theta_k}(D_{train})$.
    Each query sample $x_i \in D_{train}$, the normalized feature: $q_i = F_\theta(x_i)$.
    Each $x_i \in D_{train}$, retrieve $K$ nearest neighbors to form the positive samples set $N_i$:
      $N_i = \{h_j \mid h_j \in \underset{h_l \in Q}{\text{argtop}}_k(\text{CosineSimilarity}(q_i, h_l))\}$
    Each neighbor $h_j \in N_i$ is weighted: $\omega_j = \phi \cdot \alpha^{-\frac{l_{ij}}{k}}$ (described in Section 4.2).
    Step 3: training
    Initialize training batch size
    for $N_{train} \in D_{train}$ do
        The Encoder $F_\theta$ is trained with training batch $N_{train}$ and associated neighbors set $N_i$ and
          $Q$ using loss $L_{\text{train}}$ (Eq. 5)
        Update Encoder and Momentum Encoder parameters $\theta$ and $\theta_k$
    end
end
```