NILC: Discovering New Intents with LLM-assisted Clustering

Technical Report

Hongtao Wang
Hong Kong Baptist University
Hong Kong SAR, China
cshtwang@comp.hkbu.edu.hk

Renchi Yang[*]
Hong Kong Baptist University
Hong Kong SAR, China
renchi@hkbu.edu.hk

Wenqing Lin
JD.com
China
linwenqing.8@jd.com

Abstract

New intent discovery (NID) seeks to recognize both new and known intents from unlabeled user utterances, which finds prevalent use in practical dialogue systems. Existing works towards NID mainly adopt a cascaded architecture, wherein the first stage focuses on encoding the utterances into informative text embeddings beforehand, while the latter is to group similar embeddings into clusters (i.e., intents), typically by K-Means. However, such a cascaded pipeline fails to leverage the feedback from both steps for mutual refinement, and, meanwhile, the embedding-only clustering overlooks nuanced textual semantics, leading to suboptimal performance.

To bridge this gap, this paper proposes NILC, a novel clustering framework specially catered for effective NID. Particularly, NILC follows an iterative workflow, in which clustering assignments are judiciously updated by carefully refining cluster centroids and text embeddings of uncertain utterances with the aid of *large language models* (LLMs). Specifically, NILC first taps into LLMs to create additional *semantic centroids* for clusters, thereby enriching the contextual semantics of the Euclidean centroids of embeddings. Moreover, LLMs are then harnessed to augment hard samples (ambiguous or terse utterances) identified from clusters via rewriting for subsequent cluster correction. Further, we inject supervision signals through non-trivial techniques *seeding* and *soft must links* for more accurate NID in the semi-supervised setting. Extensive experiments comparing NILC against multiple recent baselines under both unsupervised and semi-supervised settings showcase that NILC can achieve significant performance improvements over six benchmark datasets of diverse domains consistently.

CCS Concepts

• **Computing methodologies** → *Information extraction*; • **Information systems** → *Query intent*; *Clustering and classification.*

Keywords

intent discovery, clustering, large language models

[*]Corresponding Author

1 Introduction

New Intent Discovery (NID) [23] aims to identify and group user utterances from conversational systems into both known and previously unseen intents (goals) in an open-world context, which outputs new intent labels or heuristics for faster annotations thereafter and automates the expensive manual annotation of tremendous data. This task serves as building blocks to underpin a variety of applications, such as enhancing task-oriented dialogue systems (e.g., chatbots) [7] in e-commerce or other platforms [9, 10, 27], refining search results through better query understanding [37], and elevating personalized services [15, 16] with better user modeling. The ability to dynamically discover and adapt to new intents is essential to ensure the effectiveness of these systems in an open-world environment where user needs are constantly evolving.

Existing solutions towards NID tasks can be broadly divided into two categories: *embedding-based methods* and *LLM-driven approaches*. Embedding-based methods typically follow a cascaded pipeline. They first encode all the utterances into a semantic space (i.e., text embeddings), followed by applying standard clustering algorithms like K-Means to group utterances into intent clusters [1, 5, 43]. Most of them focus on pretraining or fine-tuning a text encoder, usually small language models, on available utterance corpora with limited or even without annotations, and thus, often struggle to capture the nuanced semantics hidden in utterances, particularly when dealing with technical or ambiguous utterances. Moreover, while this modular methodology enjoys merits in simplicity and flexibility, it fails to leverage the results from two independent steps to optimize each other.

With the advent of *large language models* (LLMs), a promising way is to capitalize on the massive knowledge and superb comprehension ability of LLMs for NID, motivating a series of LLM-based methods [6, 19, 28, 35]. However, as revealed in [8, 19], the use of text embeddings generated from LLMs in the cascaded pipeline for NID is not only computationally and financially expensive due to the sheer volume of utterance data in practice, but also suffers from poor performance caused by semantic drift as LLMs are trained on general-purpose corpora. A recent work [28] directly employs LLMs as classifiers to predict the intent categories of utterances, which often demands expensive fine-tuning for domain-specific understanding and appropriate granularity, and requires sophisticated prompting engineering with carefully curated examples to attain valid, stable, and decent results. Very recently, several attempts [9, 18, 45]

have been made towards combining the embedding-based and LLM-driven methodologies. Unfortunately, these efforts are limited by the static use of LLMs, primarily for embedding alignment, and the risk of inherited bias often outweighs the moderate performance gains. In particular, most of the extant LLM-aided solutions are even inferior to the state-of-the-art embedding-based methods [42], as evidenced by our experiments.

In a nutshell, existing NID works are still suboptimal due to the problematic cascaded workflows, semantic ambiguity pervaded in user utterances, improper and ineffective use of LLMs. This leads to a critical research question: can we devise a comprehensive framework that integrates embedding-based approaches with LLMs to overcome the limitations of both for enhanced NID cost-effectively?

Our Contributions. In response to these challenges, we present New Intent Discovery LLM-assisted Clustering (NILC), a novel and effective framework specialized for both unsupervised and semi-supervised NID. Distinct from prior methods that merely focus on embedding learning or alignment, NILC mainly works on the clustering phase and operates in an iterative fashion, dynamically refining cluster assignments and text embeddings with the assistance of LLMs in each iteration, which enables the mutual optimization of both steps, and hence, circumvent the deficiency of cascaded architectures. More specifically, in addition to the standard Euclidean centroids from embeddings, NILC introduces a *dual centroid scheme* that additionally generates a semantic centroid for each cluster by summarizing its theme with LLMs. This design allows for capturing nuanced semantics neglected in embedding-only clustering, leading to more accurate cluster assignments. On top of that, instead of using static text embeddings, NILC resorts to the *hard sample refinement* to identify hard samples (e.g., ambiguous utterances) based on current clustering results and leverage in-context learning (ICL) and generative capabilities of LLMs for subsequent augmentation and clustering. The cost-efficient use of LLMs for only clusters (exemplars therein) and hard samples not only bypasses the significant expense for all utterances, but also avoids introducing noise and contamination to high-quality samples that cause performance loss. Furthermore, under the semi-supervised settings, NILC also innovatively incorporates supervision into the clustering stage through *seeding* and *soft must-link* constraints from labeled data, fostering improved NID performance.

Our extensive experiments on six benchmark datasets demonstrate that NILC remarkably outperforms a wide range of recent baselines based on embeddings or LLMs in both unsupervised and semi-supervised settings. The consistent superiority of our NILC across various domains highlights the effectiveness of our proposed framework, techniques, and optimizations.

2 Related Work

2.1 Unsupervised Intent Discovery

Unsupervised intent discovery partitions unlabeled utterances into intent categories. Early methods used traditional algorithms like K-Means [21] and *Agglomerative Clustering* (AC) [11] on static embeddings (*e.g.*, TF-IDF [26], GloVe [24]), but struggled with complex semantics. To address this, deep clustering methods learn discriminative representations end-to-end, allowing the learned features

to be tailored specifically for the NID task. For instance, DEC [38] uses an autoencoder for dimensionality reduction, DCN [30] jointly performs feature learning and clustering, and DeepCluster [5] alternates between clustering to generate pseudo-labels and supervised training. However, these purely unsupervised methods cannot leverage prior knowledge from available labeled data, which is crucial for guiding the clustering process toward more semantically coherent intents and ensuring alignment with real-world application requirements.

2.2 Semi-supervised Intent Discovery

Incorporating labeled data, semi-supervised methods achieve superior performance through two main approaches: representation learning and clustering-based techniques.

Representation learning methods aim to learn a high-quality semantic space for text embeddings, often through contrastive learning. For example, MTP-CLNN [43] uses multi-task pre-training and a contrastive loss, while USNID [42] employs a centroid-guided mechanism for self-supervised targets. Clustering-based methods often use an iterative process. CDAC+ [20] uses pairwise constraints and a target distribution for refinement. DeepAligned [41] iteratively refines cluster assignments after pre-training on labeled data. A key challenge is catastrophic forgetting, which LatentEM [44] mitigates with a probabilistic EM framework treating intents as latent variables. SDC [1] leverages model bias from a pre-trained model to calibrate a trainable one. These methods, while powerful, primarily focus on representation learning and can struggle with ambiguous utterances.

2.3 New Intent Discovery with LLMs

The advent of LLMs has introduced a new paradigm for NID. These methods can be categorized based on how they utilize LLMs. A key application is using LLMs as zero- or few-shot discoverers. For instance, IntentGPT [28] employs a sophisticated prompting strategy with a few-shot sampler and feeds discovered intents back into the prompt for on-the-fly learning. However, this can be costly for large-scale applications.

A more common approach uses LLMs as knowledge distillers or supervisors to generate high-quality signals for training smaller models. For example, IDAS [6] uses an LLM to generate abstractive summaries of utterances, cleaning the input for more effective clustering. LANID [9] queries an LLM for pairwise relationship labels to construct a contrastive fine-tuning task. Similarly, GLEAN [45] learns from diverse LLM feedback to refine model representations. While these hybrid methods are effective, they often lack a deep, iterative integration of the LLM's reasoning capabilities into the clustering process itself. Our work addresses this gap by using LLMs to iteratively strengthen ambiguous data points and enhance cluster assignments, offering greater effectiveness and interpretability.

3 Preliminaries

3.1 Problem Statement

Let $\mathcal{Y}_k$ be a set of M known intent categories, and $\mathcal{Y}_u$ be a set of $K - M$ unknown intents. Accordingly, the set of all intents (also referred to as labels) is then denoted by $\mathcal{Y} = \mathcal{Y}_k \cup \mathcal{Y}_u$ and $|\mathcal{Y}| = K$. Given a set of labeled utterances from users $\mathcal{D}_l = \{(x_i, y_i)|y_i \in$
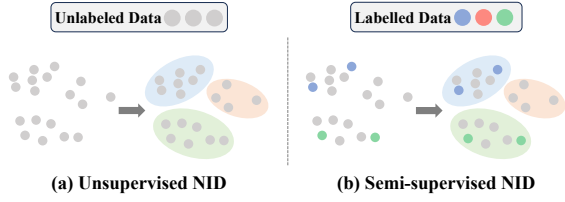
(a) Unsupervised NID   **(b) Semi-supervised NID**
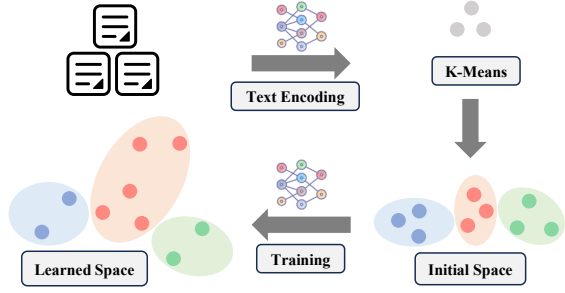
Figure 1: Illustration of NID settings.

Figure 2: Cascaded architecture for NID.

$\mathcal{Y}_k\}_{i=1}^{|\mathcal{D}_l|}$, where x_i is the i-th user utterance and y_i stands for the corresponding intent, and a set $\mathcal{D}_u = \{x_i\}_{i=1}^{|\mathcal{D}_u|}$ of unlabeled user utterances, *New Intent Discovery* (NID) aims to identify all the intent categories $\mathcal{Y}$ (containing both known and novel intents) in $\mathcal{D}_u$.

As depicted in Fig. 1, NID has two settings, i.e., unsupervised and semi-supervised ones [20, 31, 41]:

Unsupervised NID. Under the unsupervised setting, there are no labeled utterances available, i.e., $\mathcal{D}_l = \emptyset$. The goal is to cluster $\mathcal{D}_{\text{test}} = \mathcal{D}_u$ into K distinct intent categories.

Semi-supervised NID. In semi-supervised NID tasks, models are trained on training set $\mathcal{D}_{\text{train}}$ and evaluated on a *balanced* testing set $\mathcal{D}_{\text{test}} = \{(x_i, y_i)|y_i \in \mathcal{Y}\}_{i=1}^{|\mathcal{D}_{\text{test}}|}$, where the training set is composed of both labeled and unlabeled utterances, i.e., $\mathcal{D}_{\text{train}} = \mathcal{D}_l \cup \mathcal{D}_u$. In particular, under this setting, the labeled data is limited, typically with a labeled ratio $\frac{|\mathcal{D}_l|}{|\mathcal{D}_{\text{train}}|} \leq 10\%$ and a *known class ratio* (KCR) $\frac{|\mathcal{Y}_k|}{K} \leq 75\%$. The trained models are expected to discriminate known intents from $\mathcal{D}_{\text{test}}$, while mining for new intents in the rest of utterances.

3.2 Canonical Cascaded Architecture

The majority of methods for NID follow a cascaded pipeline as illustrated in Fig. 2, wherein the first stage focuses on mapping all utterances to a semantic embedding space, i.e., encoding them as text embeddings, while the second stage is a standard clustering algorithm (e.g., K-Means [21]) applied to group these embeddings into intent categories. In the literature, the first phase is often referred to as *intent learning*. A *pre-trained text encoder*, denoted as $\text{PTE}(\cdot)$, is trained or fine-tuned depending on the specific NID setting to produce the learned intent space. In the unsupervised setting ($\mathcal{D}_l = \emptyset$), the encoder $\text{PTE}(\cdot)$ is trained on the unlabeled data $\mathcal{D}_u$. In the semi-supervised setting, it is trained on $\mathcal{D}_{\text{train}} = \mathcal{D}_l \cup \mathcal{D}_u$, leveraging information from both known intents ($\mathcal{Y}_k$) and unlabeled data. After training, the final encoder maps each utterance x_i to a

---

Algorithm 1: NILC framework

Input: $\mathcal{D}_u \cup \mathcal{D}_l$, a pre-trained text encoder $\text{PTE}(\cdot)$, a large language model $\text{LLM}(\cdot)$, the number K of intents, the number T of iterations.

Output: K intent clusters $\{C_1, C_2, \ldots, C_K\}$.

1 $\{\mathbf{x}_i\}_{i=1}^{N} \leftarrow \text{PTE}(\mathcal{D}_u \cup \mathcal{D}_l)$;   ▷ Encoding utterances
2 $\{C_k\}_{k=1}^{K} \leftarrow K\text{-Means++}(\{\mathbf{x}_i\}_{i=1}^{N}, K)$; ▷ Initializing clusters
3 **for** $t \leftarrow 1$ *to* T **do**
    /* Dual Centroid Scheme (DCS)   */
4   **for** $k \leftarrow 1$ *to* K **do**
5   Compute Euclidean centroid $\boldsymbol{\mu}_k$;   ▷ Eq. (1)
6   Generate cluster summary s_k ;   ▷ Eq. (2)
7      $\boldsymbol{\theta}_k \leftarrow \text{PTE}(s_k)$;   ▷ Eq. (3)
8   Update cluster assignments ;   ▷ Eq. (4)
    /* Hard Sample Refinement (HSR)   */
9   Pick hard samples $\mathcal{H}$ with highest uncertainty in Eq. (8);
10   **for** $\mathbf{x}_i$ *in* $\mathcal{H}$ **do**
11   Generate refined utterance $\tilde{x}_i$;   ▷ Eq. (9)
12     $\tilde{\mathbf{x}}_i \leftarrow \text{PTE}(\tilde{x}_i)$;
13   Conditionally update $\mathbf{x}_i$ to $\tilde{\mathbf{x}}_i$;   ▷ Eq. (10)

---

d-dimensional embedding vector $\mathbf{x}_i = \text{PTE}(x_i)$, yielding the learned intent space $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$.

Representative NID works following the cascaded workflow include (i) DCN [30] that jointly performs intent learning and clustering, (ii) USNID [42] that designs a centroid-guided mechanism for self-supervised contrastive learning, and (iii) LatentEM [44] that resorts to a probabilistic EM framework to optimize intent assignments. However, this cascaded architecture incurs inherent deficiencies. The decoupled nature of the intent learning and clustering stages prevents any feedback or mutual refinement. Consequently, the final performance is highly sensitive to the initial embedding quality and risks overlooking nuanced semantics, particularly for ambiguous or technical utterances.

4 Methodology

In this section, we present our clustering framework NILC for both unsupervised and semi-supervised NID. Firstly, we provide an overview of NILC in § 4.1, followed by elucidating algorithmic details for updating clusters and embeddings in § 4.2 and 4.3, respectively. Subsequently, § 4.4 introduces our additional optimizations to inject semi-supervised signals into the clustering stage. Finally, we conduct analyses regarding our clustering objective and the computational complexity of NILC in § 4.5.

4.1 Synoptic Overview

NILC mainly focuses on the clustering of utterances for discovering new intents. At a high level, NILC proceeds in an iterative fashion, where each iteration involves two main stages assisted with LLMs: (i) updating clusters, including cluster centroids and assignments, and (ii) refining text embeddings based on the current clustering results. Aside from the Euclidean centroids averaged over the text embeddings in clusters, the basic idea of the first stage is to introduce *semantic centroids* generated by LLMs to better summarize the main semantic themes of detected clusters for more accurate
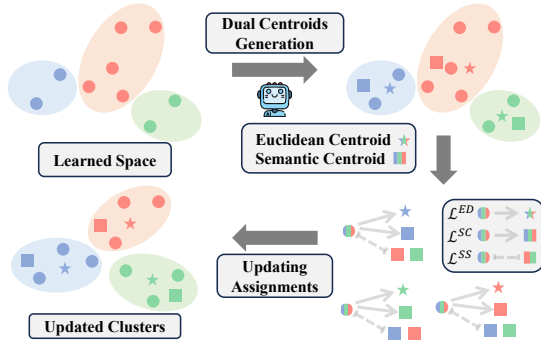
Figure 3: Dual Centroid Scheme in NILC.

Figure 4: Prompts for generating cluster summaries.

cluster assignments, which is referred to as *dual centroid scheme* (DCS). The second stage is targeted to cope with *hard samples*, whose corresponding utterances are ambiguous or terse, rendering it hard to get informative text embeddings via the text encoder and determine their intent clusters subsequently with high confidence. The idea of NILC is to harness the extensive knowledge in LLMs to augment these utterances, and hence, obtain refined text embeddings for certain cluster assignments, dubbed as *hard sample refinement* (HSR). Notice that this HSR paradigm enables us to capitalize on the feedback from clustering to optimize the text embeddings, preventing the defects of traditional cascaded pipelines remarked earlier.

Algorithm 1 summarizes the main steps in NILC. More concretely, given $\mathcal{D}_u \cup \mathcal{D}_l$, a PTE$(\cdot)$, an LLM$(\cdot)$, the numbers K of intents and T of iterations, NILC begins by encoding all the utterances $\{x_i\}_{i=1}^N$ in $\mathcal{D}_u \cup \mathcal{D}_l$ into text embeddings $\{\mathbf{x}_i\}_{i=1}^N$ (Line 1), and based thereon, initializes the K clusters using K-Means++ algorithm, a standard practice in clustering [2] (Line 2). After that, NILC starts an iterative procedure for updating clusters using DCS and refining hard samples using HSR (Lines 3-13). In each of the T iterations, the DCS phase will re-calculate the Euclidean centroids $\{\boldsymbol{\mu}_k\}_{k=1}^K$ of the current clusters $\{C_k\}_{k=1}^K$, and generate semantic centroids $\{\boldsymbol{\theta}_k\}_{k=1}^K$ through LLM$(\cdot)$ and PTE$(\cdot)$ (Lines 4-7). The cluster assignments of all utterance samples will later be updated accordingly at Line 8. Subsequently, NILC proceeds to the HSR stage, in which a small set of hard samples $\mathcal{H}$ is first identified, followed by a refinement of their textual contents leveraging LLM$(\cdot)$ and PTE$(\cdot)$, as well as a conditional update of their text embeddings (Lines 9-13).

In the succeeding subsections, we elaborate on the details of generating semantic centroids with LLMs, updating cluster assignments, as well as identifying and refining hard samples with LLMs.

4.2 Cluster Update with Dual Centroids

As displayed in Fig. 3, the DCS is to derive an Euclidean centroid $\boldsymbol{\mu}_k$ and a semantic centroid $\boldsymbol{\theta}_k$ for each of the K intent clusters $\{C_1, C_2, \ldots, C_K\}$. Akin to K-Means, the Euclidean centroid $\boldsymbol{\mu}_k$ of cluster C_k is computed as the mean of the embedding vectors of the samples therein, i.e.,

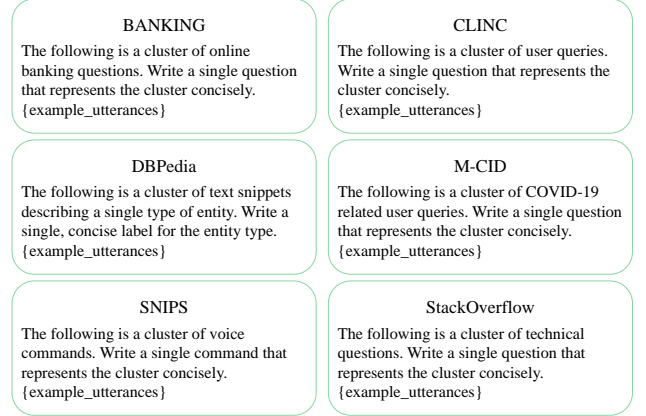$$\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} \mathbf{x}_i. \tag{1}$$

Although the text embeddings $\{\mathbf{x}_i\}_{i=1}^N$ are often obtained via the domain- or task-specific PTE, directly averaging them as centroids will obscure nuanced textual relationships. Moreover, such centroids have no explicit and precise semantic themes [8], i.e., the underlying intent categories are indefinite, rendering some utterances likely to be misassigned to the intent clusters. As a remedy, we directly summarize the utterances in each cluster C_k as a semantic centroid $\boldsymbol{\theta}_k$ to complement the Euclidean centroid $\boldsymbol{\mu}_k$.

Generating Semantic Centroids with LLMs. Instead of feeding all the utterances in each cluster to LLMs, which are both financially and computationally expensive, NILC cherry-picks a subset $\mathcal{S}_k \subset C_k$ of $|\mathcal{S}_k|$ (typically $|\mathcal{S}_k| = 10$) exemplars for cluster C_k based on a certain selection strategy[1], such as K-Means for maximal diversity, *maximal marginal Relevance* (MMR) for balancing relevance and diversity, *mean average distance* (MAD), and *nearest neighbors* (NN), etc. As exemplified in Fig. 4, together with a summary generation prompt p_{smry}, these samples are subsequently forwarded to an LLM to generate a textual summary s_k for cluster C_k:

$$s_k = \text{LLM}(p_{\text{smry}}, \mathcal{S}_k). \tag{2}$$

Intuitively, by unleashing the extensive knowledge and remarkable summarization capacity of LLMs, s_k can convey the intent semantics of cluster C_k more precisely. We then encode it as the semantic centroid in the form of embedding vectors by the PTE:

$$\boldsymbol{\theta}_k = \text{PTE}(s_k), \tag{3}$$

which has the potential to assist us in correcting the misassignments of utterance samples in cluster C_k.

Updating Cluster Assignments. Given dual centroids $\boldsymbol{\mu}_k$ and $\boldsymbol{\theta}_k$ for each cluster C_k, the next task is to update the cluster assignments for all utterance samples. The assignments are determined through minimizing a joint clustering cost function based on dual centroids, namely,

$$y_i = \underset{1 \leq k \leq K}{\arg\min} f(\mathbf{x}_i) = \mathcal{L}_i^{\text{ED}} + \alpha \cdot \mathcal{L}_i^{\text{SC}} + \beta \cdot \mathcal{L}_i^{\text{SS}} \tag{4}$$

where α and β stand for coefficients adjusting the importance of the terms $\mathcal{L}_i^{\text{SC}}$ and $\mathcal{L}_i^{\text{SS}}$, respectively. Specifically, the first term

---
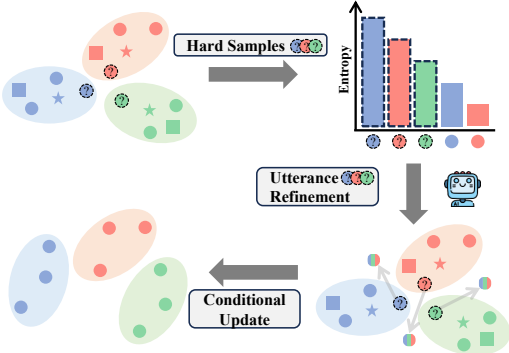[1]The details and evaluations are deferred to Appendix D, F.1, and G.

Figure 5: Hard Sample Refinement in NILC.

$\mathcal{L}_i^{\text{ED}}$ measures the Euclidean distance of sample $\mathbf{x}_i$ to any Euclidean centroid $\boldsymbol{\mu}_k$, i.e.,

$$\mathcal{L}_i^{\text{ED}} = \left\| \mathbf{x}_i - \boldsymbol{\mu}_k \right\|^2. \tag{5}$$

The other two terms, $\mathcal{L}_i^{\text{SC}}$ and $\mathcal{L}_i^{\text{SS}}$, quantify the semantic dissimilarities between sample $\mathbf{x}_i$ and the cluster from the intra- and inter-cluster perspectives, respectively. More specifically, $\mathcal{L}_i^{\text{SC}}$ is defined as the dissimilarity of utterance $\mathbf{x}_i$ and the semantic centroid $\boldsymbol{\theta}_k$ of cluster C_k:

$$\mathcal{L}_i^{\text{SC}} = 1 - \cos\left(\mathbf{x}_i, \boldsymbol{\theta}_k \right). \tag{6}$$

Intuitively, this term is minimized when we assign the cluster with the highest semantic similarity, i.e., most relevant theme, to $\mathbf{x}_i$, which in turn encourages increasing the intra-cluster semantic cohesion of resulting clusters.

Conversely, the semantic separation term $\mathcal{L}_i^{\text{SS}}$ serves as a repulsive force to enhance inter-cluster distinctiveness, which is formulated as

$$\mathcal{L}_i^{\text{SS}} = \cos(\mathbf{x}_i, \boldsymbol{\theta}_{\text{nbr}(k)}), \tag{7}$$

where $\boldsymbol{\theta}_{\text{nbr}(k)}$ represents the other semantic centroid that is closest to the semantic centroid $\boldsymbol{\theta}_k$ of the cluster assigned to $\mathbf{x}_i$. Through additionally minimizing the similarity of sample $\mathbf{x}_i$ to the nearest neighboring semantic centroid of $\boldsymbol{\theta}_k$, NILC ensures all utterances within cluster C_k radically differ from the rest of the intent clusters in terms of semantic themes.

4.3 Hard Sample Refinement with LLMs

In updated clusters, there usually exist a number of *hard samples*, which are located on the boundary of clusters, as their corresponding input utterances are usually noisy, ambiguous, overly sketchy, abbreviation-heavy, or cryptic due to the presence of jargon and slang, etc. For instance, the utterance "Confusion regarding laziness" from StackOverflow is ambiguous, as "laziness" could refer to general performance issues or a specific concept in functional programming. This ambiguity can lead to misclassification into a related but incorrect cluster (e.g., LINQ queries) [2]. As such, these samples are likely to cause the "centroid shift" since Euclidean centroids are simply averaged over within-cluster text embeddings, and thus, slightly distort the themes of the intent clusters and result in erroneous assignments. As depicted in Fig. 5, HSR mitigates this issue through a refinement scheme consisting of the following three steps. With HSR, the above utterance example can be rewritten as

---
[2]Case studies are provided in Appendix F.2.

Figure 6: Prompt template for HSR.

a more precise one "Understanding laziness in functional programming languages like Haskell," making it more likely to be correctly classified.

Identifying Hard Samples. To identify hard utterance samples, we propose to assess the uncertainty of each sample's clustering assignment and select a subset $\mathcal{H}$ comprising the top-δ (typically $\delta = 10$) samples with the highest uncertainty. Let $P(C_k|\mathbf{x}_i)$ be the posterior probability of assigning sample $\mathbf{x}_i$ to intent cluster C_k, i.e., soft assignment. Following the practice in *deep clustering* [38], we employ the classic Gaussian kernel to measure the similarity between sample $\mathbf{x}_i$ and centroid $\boldsymbol{\mu}_k$, and then transform it into the posterior probability via the softmax function:

$$P(C_k|\mathbf{x}_i) = \frac{\exp(-\|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2)}{\sum_{\ell=1}^{K} \exp(-\|\mathbf{x}_i - \boldsymbol{\mu}_\ell\|^2)}.$$

A simple and effective way to measure the assignment uncertainty is the prominent *Shannon entropy* [29]. In mathematical terms, for each sample $\mathbf{x}_i$, the uncertainty is formulated by

$$H(\mathbf{x}_i) = -\sum_{k=1}^{K} P(C_k|\mathbf{x}_i) \cdot \log P(C_k|\mathbf{x}_i). \tag{8}$$

A high entropy value indicates that the assignment probabilities are evenly distributed among all K clusters, implying high uncertainty. Intuitively, the larger $H(\mathbf{x}_i)$ is, the less confident we are about the cluster assignment for utterance x_i.

Context-aware Rewriting. After the obtainment of the hard sample set $\mathcal{H}$, NILC enters into rewriting their original utterances with LLMs. Inspired by a recent finding that editing input text can be more effective than generating it from scratch [39], we introduce a context-aware rewriting mechanism that structures the LLM's task to emulate a cognitive process of analysis before synthesis. This "judge-then-rewrite" paradigm first compels the model to analytically determine the most plausible cluster for a given hard sample before reformulating the utterance, thereby enhancing the relevance and quality of the generated text.

This process is guided by a meticulously designed prompt template, denoted as p_{ref} and detailed in Fig. 6. For each hard sample (i.e., ambiguous utterance) x_i, the prompt is furnished with rich in-context information pertinent to its currently assigned cluster

C_k, referred to as "home" cluster, and K_{nbr} nearest neighboring clusters $\{C_j\}_{j=1}^{K_{\mathrm{nbr}}}$ (typically $K_{\mathrm{nbr}} = 10$), dubbed as *neighboring clusters*. More precisely, we include the semantic summary s_k and a set of representative exemplars $\mathcal{S}_k$ of home cluster C_k into prompt p_{ref}, as well as its counterparts $\{s_j, \mathcal{S}_j\}_{j=1}^{K_{\mathrm{nbr}}}$ for the K_{nbr} neighboring clusters. Afterwards, the LLM is invoked to generate the refined utterance $\tilde{x}_i$ by

$$\tilde{x}_i = \mathrm{LLM}\left(p_{\mathrm{ref}}, x_i\right). \tag{9}$$

Conditional Update. Lastly, the revised version $\tilde{x}_i$ of utterance x_i is encoded by $\mathrm{PTE}(\cdot)$ previously used as the new text embedding $\tilde{\mathbf{x}}_i$. Instead of blindly updating $\mathbf{x}_i$ to $\tilde{\mathbf{x}}_i$ that might introduce noise from LLMs, we substitute $\tilde{\mathbf{x}}_i$ for original $\mathbf{x}_i$ for subsequent clustering steps only if $\tilde{\mathbf{x}}_i$ can leads to a reduction in the clustering cost $f(\mathbf{x}_i)$ of x_i defined in Eq. (4). Mathematically,

$$\mathbf{x}_i = \begin{cases} \tilde{\mathbf{x}}_i & \text{if } \min_{1 \le k \le K} f(\tilde{\mathbf{x}}_i) < \min_{1 \le k \le K} f(\mathbf{x}_i), \\ \mathbf{x}_i & \text{otherwise.} \end{cases} \tag{10}$$

This conditional update enforces that the clustering cost for each sample is greedily minimized even when we involve LLMs for data augmentation, i.e., rewriting utterances, thereby ensuring stable and reliable clustering results.

4.4 Optimizations for Semi-Supervised NID

Recall that under semi-supervised settings, we are provided with a limited set $\mathcal{D}_l$ of labeled utterance samples. In previous works, such supervision signals are solely considered in the text encoding phase, which are often underexploited. In semi-supervised NID, in addition to applying the PTE finetuned on $\mathcal{D}_{\mathrm{train}}$ containing both labeled and unlabeled datasets, NILC injects ancillary knowledge from $\mathcal{D}_l$ to the clustering stage through the *seeding* and *soft must-links* (SML) techniques in the sequel. The high-level ideas of these two optimizations are illustrated in Fig. 7.

Seeding. Taking inspiration from *seeded K-Means* [3], we perform a warm-start for the clustering process by aligning a subset of the initial Euclidean centroids with the known intents $\mathcal{Y}_k$ from the labeled dataset $\mathcal{D}_l$. The initial Euclidean centroids, denoted as $\{\boldsymbol{\mu}_k^0\}_{k=1}^K$, are obtained by initially running K-Means++ over all input text embeddings (Line 2 in Algorithm 1). The seed centroids, $\{\boldsymbol{\mu}_j^*\}_{j=1}^M$, are derived from the labeled data $\mathcal{D}_l$ by computing the mean embedding for all utterances belonging to each of the M known intents in $\mathcal{Y}_k$, i.e.,

$$\boldsymbol{\mu}_j^* = \mathrm{Mean}(\{\mathbf{x}_i | (x_i, y_i) \in \mathcal{D}_l, y_i = j\}).$$

The matching from $\{\boldsymbol{\mu}_k^0\}_{k=1}^K$ to $\{\boldsymbol{\mu}_j^*\}_{j=1}^M$ is framed as a linear assignment problem. The objective is to find an optimal mapping π that minimizes the total semantic distance, i.e., cosine dissimilarity, between the seed centroids and the initial Euclidean centroids:

$$\min_\pi \sum_{j=1}^M \left(1 - \cos(\boldsymbol{\mu}_j^*, \boldsymbol{\mu}_{\pi(j)}^0)\right), \tag{11}$$

which can be readily solved using the *Hungarian algorithm* [22].

Once the mapping π is found, the M initial centroids are replaced by their corresponding seed centroids: $\boldsymbol{\mu}_{\pi(j)}^0 \leftarrow \boldsymbol{\mu}_j^*$. This procedure anchors a portion of the clusters in the known semantic space, providing a strong inductive bias from the outset.
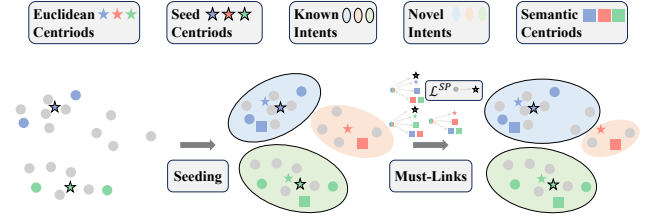

Figure 7: Semi-Supervised Optimizations of NILC.

Soft Must-Links. Inspired by the principles of *constrained clustering* [4, 36], we propose to impose constraints in the form of soft must-links in the course of iterative updating clusters. Such soft must-links aim at pulling utterance samples towards clusters that have been mapped to their known intents. Unlike previous constrained clustering methods [4, 36] that rely on pre-defined and static pairwise constraints, our approach is dynamic and semantically driven, providing a weighted and soft pull towards known intents.

This process involves two main steps. Firstly, in the t-th iteration, a dynamic one-to-one mapping π^t is established between the M known intents and the current K clusters. The mapping can be built based on either the similarities between their centroid embeddings or the answers from LLMs. Due to space limit, we defer the details and evaluations for these two strategies to Appendix E, F.3, and G. After that, if a cluster C_k is mapped to a known intent j, i.e., $\pi^t(k) = j$, we introduce an extra term to our cost function $f(\mathbf{x}_i)$ in Eq. (4) for assigning clusters as supervision:

$$\mathcal{L}_i^{\mathrm{SP}} = 1 - \cos(\mathbf{x}_i, \boldsymbol{\mu}_{\pi^t(k)}^*), \tag{12}$$

which seeks to minimize the distance from sample $\mathbf{x}_i$ to the known intent cluster $\boldsymbol{\mu}_{\pi^t(k)}^*$, acting as a soft must-link between them. Consequently, this leads to our new cost function:

$$f'(\mathbf{x}_i) = \mathcal{L}_i^{\mathrm{ED}} + \alpha \cdot \mathcal{L}_i^{\mathrm{SC}} + \beta \cdot \mathcal{L}_i^{\mathrm{SS}} + \gamma \cdot \mathcal{L}_i^{\mathrm{SP}}, \tag{13}$$

where the hyperparameter γ controls the strength of these soft must-link. Note that if no mapping exists for cluster C_k, the cost function remains the same in Eq. (4).

4.5 Analysis

The iterative process of our framework is analogous to an Expectation-Maximization (EM) algorithm, seeking to find an optimal partition $C = \{C_k\}_{k=1}^K$ and its associated representations (centroids $\{\boldsymbol{\mu}_k, \boldsymbol{\theta}_k\}$ and potentially refined embeddings $\{\tilde{\mathbf{x}}_i\}$) that minimize the global objective function $\mathcal{L} = \sum_{k=1}^K \sum_{\mathbf{x}_i \in C_k} f'(\mathbf{x}_i)$. Our algorithm iteratively seeks a local minimum for this objective by alternating between assignment and update/refinement phases.

The T iterations of NILC are performed periodically at specified intervals, while most other iterations are standard K-Means steps. Regarding computational cost, most standard steps consisting of only K-Means have a complexity of $O(NKd)$ dominated by the assignment phase, on which a complete NILC iteration builds with the same primary complexity of $O(NKd)$, adding overhead from a fixed number of LLM calls $(K + \delta)$ and the optional Hungarian algorithm for mapping $(O(\max(M,K)^3))$. Since $K, M, \delta \ll N$, the

Table 1: Dataset statistics.

Dataset	#Utterances (N)	#Intents (K)	Domain
CLINC	22,500	150	General
BANKING	13,083	77	Banking
StackOverflow	20,000	20	Technical
M-CID	1,745	16	Covid-19
SNIPS	14,484	7	Voice
DBPedia	14,000	14	Ontology

$O(NKd)$ term remains the dominant factor, and the computational cost of NILC is competitive with standard K-Means.

5 Experiments

This section experimentally evaluates the effectiveness of the proposed NILC framework. All experiments are conducted on a Linux machine with an NVIDIA A100 GPU (80GB RAM), AMD EPYC 7513 CPU (2.6 GHz), and 1TB RAM. The source code and datasets are publicly accessible at https://github.com/HKBU-LAGAS/NILC.

5.1 Datasets

We conduct extensive experiments on six challenging benchmark datasets to evaluate the performance of our proposed method. The datasets include: CLINC, a multi-domain dataset with 22,500 utterances and 150 intents; BANKING, a fine-grained dataset from the banking domain with 13,083 utterances and 77 intents; Stack-Overflow, a technical question dataset with 20,000 samples across 20 classes; SNIPS, a personal voice assistant dataset containing 14,484 utterances with 7 intents; M-CID, which consists of 1,745 utterances related to 16 COVID-19 service intents; and DBPedia, a dataset of 14,000 samples from 14 non-overlapping ontology classes. A summary of the dataset statistics is provided in Table 1.

5.2 Baselines and Settings

We compare our method against a wide range of baselines.
Unsupervised Methods. This category does not use any labeled data. We compare against: SAE [34], which is based on a stacked autoencoder; DEC [38] and DCN [30], which are deep clustering frameworks; CC [17] and SCCL [40], which are based on contrastive learning; and USNID [42], which leverages pre-training for intent discovery.
Semi-Supervised Methods. These methods leverage a small amount of labeled data. This category includes methods from various technical approaches: constrained clustering (KCL [13], MCL [14]); novel category discovery adapted from computer vision (DTC [12], GCD [33]); and methods for new intent discovery (CDAC+ [20], DeepAligned [41], SDC [1], MTP-CLNN [43], LatentEM [44]). We also include LANID [9], which uses an LLM to generate pairwise relationship labels for contrastive fine-tuning; and IntentGPT [28], which employs the LLM as a few-shot discoverer through a sophisticated prompting strategy. Additionally, we evaluate USNID in its semi-supervised setting.

Unless specified otherwise, we employ USNID as PTE($\cdot$) and GPT-4o-Mini as LLM($\cdot$) in NILC. The detailed settings for hyperparameters are provided in Appendix C. Following prior studies [9, 28, 42, 43], *normalized mutual information* (NMI), *adjusted*

Table 2: Ablation study on NILC.

Variant	DBPedia	M-CID	StackOverflow
	NMI/ARI/ACC	NMI/ARI/ACC	NMI/ARI/ACC
NILC (unsup.)	78.21/65.61/73.43	71.87/52.36/68.77	72.08/64.35/76.62
w/o DCS	77.14/64.25/72.00	69.49/49.25/67.34	71.67/63.95/76.45
w/o HSR	77.49/64.87/72.71	69.89/49.93/67.62	71.80/63.93/76.45
NILC (semi.)	89.99/84.88/92.00	83.42/73.20/85.10	80.73/76.67/87.28
w/o DCS	88.61/83.12/91.00	80.75/69.33/81.95	80.56/75.42/86.60
w/o HSR	88.73/83.43/91.14	81.42/70.37/82.81	80.57/76.09/86.98
w/o Seeding	89.36/84.19/91.57	82.20/71.73/83.95	80.53/76.36/87.08
w/o SML	89.09/83.87/91.43	82.06/71.33/83.67	80.28/76.16/86.95
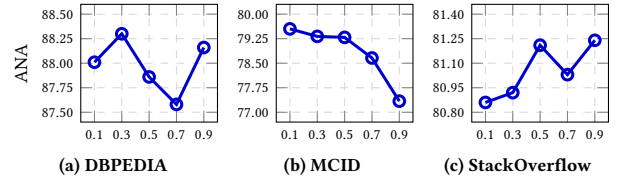
Figure 8: Varying α.

rand score (ARI), and *clustering accuracy* (ACC) are used as NID metrics.

5.3 NID Performance

Table 3 reports the NMI, ARI, and ACC scores for NILC and all baseline methods on the six benchmark datasets. We can draw several key observations.

First, under the unsupervised setting, NILC consistently and significantly outperforms all baselines across all datasets. For instance, on the M-CID dataset, NILC achieves an improvement of 7.22% in NMI, 7.72% in ARI, and 11.38% in ACC over the strongest baseline, USNID. This demonstrates the effectiveness of our LLM-assisted DCS and HSR in discovering coherent intent clusters without any labeled data.

Second, in the more practical semi-supervised setting, NILC continues to establish its superiority. It surpasses all recent and competitive baselines, including those that also leverage LLMs like LANID and IntentGPT. On the CLINC dataset, for example, NILC improves upon the runner-up USNID by 0.41% in NMI, 1.76% in ARI, and 1.67% in ACC. These gains are consistent across diverse domains, from general intents in CLINC to technical questions in StackOverflow, validating the robustness of injecting supervised signals through IS and ML within our iterative framework.

Another important observation is that while methods based on deep representation learning (e.g., USNID, MTP-CLNN) form a strong baseline, our framework's ability to iteratively refine both cluster assignments and embeddings provides an additional performance boost. This highlights the limitations of a static, cascaded approach and confirms the benefits of a more synergistic methodology where the clustering process and embedding space mutually enhance each other with the aid of an LLM's reasoning capabilities.

Table 3: NID Performance comparison. (best bolded and runner-up <u>underlined</u>)

	Method	CLINC NMI/ARI/ACC	BANKING NMI/ARI/ACC	StackOverflow NMI/ARI/ACC	M-CID NMI/ARI/ACC	SNIPS NMI/ARI/ACC	DBPedia NMI/ARI/ACC
Unsupervised	SAE	74.07/32.06/47.80	60.01/24.17/37.94	46.35/29.65/51.62	50.49/43.61/53.07	76.07/69.80/81.63	71.34/57.57/70.07
	DEC	75.14/32.22/49.24	62.85/25.94/38.84	60.26/36.92/59.93	51.09/<u>44.64</u>/53.73	84.49/80.58/87.80	74.54/59.87/70.74
	DCN	75.15/32.20/49.23	62.81/25.92/38.83	60.41/37.02/60.00	51.09/<u>44.64</u>/53.73	85.52/80.61/87.80	74.57/<u>59.89</u>/<u>70.76</u>
	CC	66.05/18.34/33.09	44.64/9.73/21.21	20.38/9.21/21.99	55.75/33.08/50.29	82.96/77.02/85.90	71.56/53.29/66.79
	SCCL	79.14/38.12/49.96	63.43/26.32/39.92	68.69/36.97/68.28	55.18/30.05/48.71	72.88/55.53/68.81	<u>77.02</u>/59.29/67.57
	USNID	<u>91.45</u>/<u>70.02</u>/<u>77.06</u>	<u>75.61</u>/<u>43.96</u>/<u>55.09</u>	<u>71.49</u>/<u>52.13</u>/<u>69.20</u>	<u>64.65</u>/41.51/<u>57.39</u>	<u>89.46</u>/<u>85.72</u>/<u>91.43</u>	75.15/58.90/68.06
	NILC	**91.58/71.53/78.36**	**77.43/47.74/59.45**	**72.08/64.35/76.62**	**71.87/52.36/68.77**	**91.56/90.34/95.57**	**78.21/65.61/73.43**
	Improv.	+0.13/+1.51/+1.30	+1.82/+3.78/+4.36	+0.59/+12.22/+7.42	+7.22/+7.72/+11.38	+2.10/+4.62/+4.14	+1.19/+5.72/+2.67
Semi-Supervised	KCL	86.10/58.86/69.22	73.07/45.49/59.27	64.84/55.86/71.06	44.10/22.29/40.92	77.94/64.11/74.00	78.78/61.63/69.13
	MCL	87.38/61.48/70.39	74.46/48.07/61.52	63.24/55.97/71.31	55.25/35.06/53.44	81.07/67.75/74.44	79.94/64.22/72.73
	DTC	89.43/67.26/77.61	73.98/43.95/56.11	62.64/53.32/70.36	33.78/11.43/29.80	73.50/62.83/74.01	77.72/58.98/68.14
	CDAC+	85.93/55.81/68.01	68.03/35.61/48.77	55.85/41.82/62.53	55.64/32.97/52.92	83.13/77.36/86.97	80.23/65.38/75.34
	GCD	88.99/65.58/76.42	71.99/42.85/56.43	60.80/42.25/65.28	60.71/40.80/58.71	81.52/78.10/89.72	79.36/64.81/76.63
	DeepAligned	93.89/79.75/86.49	79.12/52.46/63.73	73.83/60.26/77.87	48.34/23.28/41.26	88.09/85.21/92.71	84.34/69.99/78.86
	MTP-CLNN	95.44/84.23/89.30	84.63/64.32/75.33	73.88/64.04/79.68	76.76/63.29/78.08	89.95/87.90/93.39	80.17/67.13/78.14
	LatentEM	94.86/82.40/88.40	81.81/57.96/70.42	75.46/63.30/74.30	68.40/48.12/63.61	82.89/76.32/83.43	**91.46**/<u>83.40</u>/<u>87.00</u>
	USNID	<u>96.46</u>/<u>86.49</u>/<u>90.20</u>	<u>87.67</u>/<u>69.93</u>/<u>78.52</u>	<u>80.01</u>/<u>74.74</u>/<u>85.61</u>	79.04/66.53/78.83	<u>93.32</u>/<u>91.94</u>/<u>96.28</u>	86.29/76.79/85.40
	SDC	95.33/83.85/89.32	85.11/66.16/77.22	77.45/60.10/80.25	46.69/22.13/41.52	89.04/86.57/94.04	80.03/67.41/81.53
	IntentGPT [◇]	96.06/84.76/88.76	85.94/66.66/77.21	-	-	-	-
	LANID	96.08/85.25/89.81	87.18/68.56/76.75	75.30/64.71/77.42	<u>82.53</u>/<u>69.84</u>/<u>81.09</u>	91.70/90.23/94.75	85.38/74.72/83.87
	NILC	**96.87/88.25/91.87**	**87.74/71.30/81.07**	**80.73/76.67/87.28**	**83.42/73.20/85.10**	**95.61/95.14/97.86**	<u>89.99</u>/**84.88/92.00**
	Improv.	+0.41/+1.76/+1.67	+0.07/+1.37/+2.55	+0.72/+1.93/+1.67	+0.89/+3.36/+4.01	+2.29/+3.20/+1.58	-1.47/+1.48/+5.00

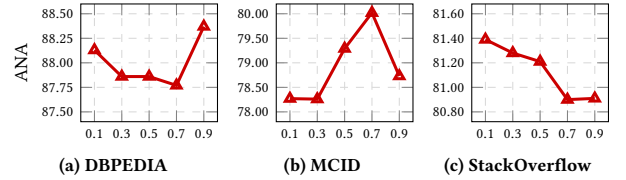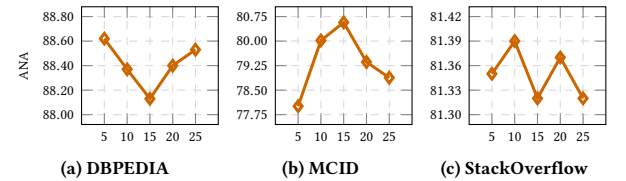[◇]Results are taken from the original paper. Missing values (-) indicate that the results are not available.

Table 4: Performance comparison of NILC on various encoders across different datasets.

Method	DBPedia (NMI/ARI/ACC)	M-CID (NMI/ARI/ACC)	StackOverflow (NMI/ARI/ACC)
SentenceBERT	70.34/54.58/67.71	60.30/37.19/59.03	68.08/55.67/67.32
NILC (SentenceBERT)	71.29/56.85/72.86	67.89/39.17/65.04	70.82/57.35/69.08
Instructor	70.79/57.65/71.86	56.46/31.57/49.28	74.21/62.81/77.28
NILC (Instructor)	77.70/64.66/73.43	62.55/40.62/59.03	75.78/63.00/77.40
MTP-CLNN	80.17/67.13/78.14	76.76/63.29/78.08	73.88/64.04/79.68
NILC (MTP-CLNN)	81.23/68.42/79.56	77.18/64.23/78.85	74.69/65.27/80.54
LatentEM	91.46/83.40/87.00	68.40/48.12/63.61	75.46/63.30/74.30
NILC (LatentEM)	91.52/85.61/89.75	69.14/51.24/64.97	76.34/64.29/75.72

Figure 10: Varying γ.

Figure 11: Varying the number δ of hard samples.
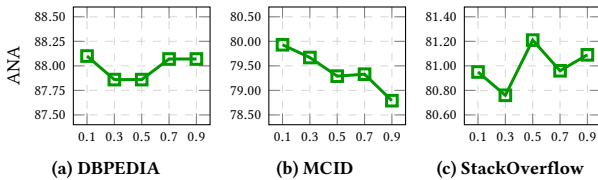
Figure 9: Varying β.

5.4 Ablation Study

To validate the effectiveness of the core components of NILC, we conduct a series of ablation studies. As shown in Table 2, we analyze the influence of removing key components from NILC in both unsupervised and semi-supervised settings. In the unsupervised setting, removing either DCS or HSR leads to a noticeable drop in performance across all tested datasets. For example, on DBPedia, removing DCS decreases the NMI by 1.07%, while removing HSR also degrades performance, confirming that both components are crucial for discovering high-quality clusters.
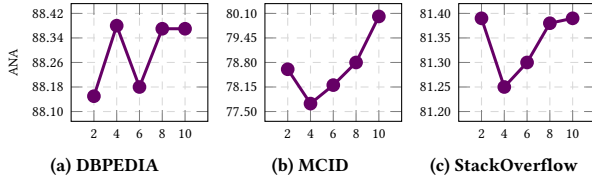
Figure 12: Varying the number K_{nbr} of neighboring clusters.
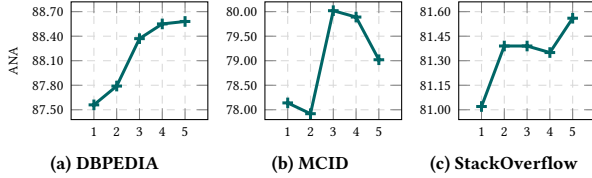
Figure 13: Varying the number T of NILC iterations.

Figure 14: Varying KCR.

Figure 15: Varying LLMs.

The same trend holds in the semi-supervised setting. Disabling DCS or HSR consistently lowers the NMI, ARI, and ACC scores. We also study the impact of removing the semi-supervised components: seeding and SML. The results show that both components contribute positively to the final performance. For instance, on M-CID, removing SML causes the ARI to drop from 73.20% to 71.33%. These findings underscore that each component in NILC plays an integral and synergistic role in its overall effectiveness.

5.5 Parameter Analysis

Let ANA be the average of ACC, NMI, and ARI. We conduct experiments to analyze the performance of NILC against strong competitors under different NID settings and the sensitivity of NILC to its key hyperparameters.

Known Class Ratio. Fig. 14 illustrates the performance of our method compared to strong baselines (LANID, USNID) as the Known Class Ratio (KCR) varies in {25%, 50%, 75%}. Across different datasets (DBPedia, MCID, and StackOverflow), our method consistently outperforms the others at every KCR level. Notably, the performance of all methods generally improves with a higher KCR, which is expected as more labeled data provides better supervision. However, the performance gap between our method and the competitors remains significant, highlighting the robustness of NILC even in low-resource settings.

Hyperparameter Sensitivity. We analyze the impact of the clustering cost weights α, β, and γ on DBPEDIA, MCID, and StackOverflow, with results shown in Figs. 8, 9, and 10. We can observe that NILC's performance remains stable across a wide range of values for each hyperparameter. For example, varying α in {0.1, 0.3, 0.5, 0.7, 0.9} on StackOverflow results in only minor fluctuations in ANA, which stays within a tight range of 80.86 to 81.24. This indicates that our method is not overly sensitive to the precise tuning of these weights, making it practical for real-world applications.

In-Context Learning Parameters. We examine the influence of key in-context learning (ICL) parameters. For selected hard samples, as shown in Fig. 11, it shows that a moderate number in {10, 15} often yields the best results on MCID, while performance on other
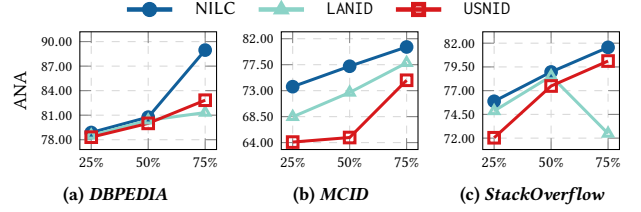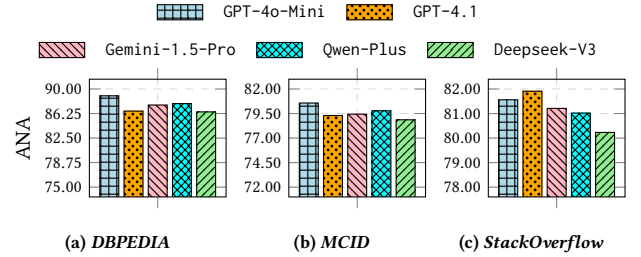
datasets is less sensitive. Fig. 12 shows the impact of the number of neighboring clusters used for context. Performance is generally robust, with the tendency that higher values deliver stronger results. Finally, Fig. 13 indicates that the model benefits from multiple LLM-driven NILC iterations, with performance generally improving or stabilizing after 3 iterations. This suggests that a few rounds of refinement are sufficient to achieve significant gains.

Pre-trained Text Encoders. We investigate the impact of different PTE(·) on NILC's performance. As shown in Table 4, we replace USNID with four models: SentenceBERT [25], Instructor [32], MTP-CLNN, and LatentEM. The results clearly demonstrate that NILC consistently enhances the performance of all encoders across DBPedia, M-CID, and StackOverflow. For instance, when applied to Instructor on DBPedia, NILC improves the NMI from 70.79% to 77.70%, ARI from 57.65% to 64.66%, and ACC from 71.86% to 73.43%. This underscores the robustness and versatility of NILC, as it is not dependent on a specific encoder but can effectively augment various text representation models to achieve better clustering outcomes.

Large Language Models. We evaluate the performance of NILC with different LLMs, including GPT-4o-Mini, GPT-4.1, Qwen-Plus, Gemini-1.5-Pro, and Deepseek-V3. As shown in Fig. 15, all LLMs achieve competitive results, indicating that our framework is robust and not overly sensitive to the choice of a specific LLM. This flexibility allows users to choose an LLM that best fits computational and financial constraints without a significant drop in performance, highlighting the practical applicability of NILC.

6 Conclusion

In this paper, we propose NILC, a framework for New Intent Discovery that synergizes embedding-based clustering and Large Language Models. Our method iteratively refines cluster assignments and text embeddings, featuring the dual centroid scheme (Euclidean and LLM-semantic) and an integrated hard sample refinement mechanism. We also demonstrate how to inject semi-supervised signals

through seeding and soft must-links. Experiments on six benchmark datasets show that NILC consistently achieves state-of-the-art performance in both unsupervised and semi-supervised settings, validating our integrated, LLM-assisted approach.

Acknowledgments

Ethical Considerations

The direct negative societal impacts of this research—specifically with respect to fairness, privacy, and security—are minimal. Nonetheless, as with other NID solutions, erroneous results by the method may affect system functionality. Although the algorithm's effectiveness has been extensively validated through experiments, occasional inaccuracies, particularly when processing noisy data, are still possible. To mitigate these risks, it is recommended to enhance data quality through rigorous data cleaning and preprocessing prior to method deployment.

References

[1] Wenbin An, Haonan Lin, Jiahao Nie, Feng Tian, Wenkai Shi, Yaqiang Wu, Qianying Wang, and Ping Chen. 2025. Unleashing the Potential of Model Bias for Generalized Category Discovery. In *AAAI*, Vol. 39. 15365–15373.

[2] David Arthur and Sergei Vassilvitskii. 2007. k-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 1027–1035.

[3] Sugato Basu, Arindam Banerjee, and Raymond J Mooney. 2002. Semi-supervised clustering by seeding. In *Proceedings of the nineteenth international conference on machine learning*. 27–34.

[4] Sugato Basu, Arindam Banerjee, and Raymond J Mooney. 2004. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining*. SIAM, 333–344.

[5] 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)*. 132–149.

[6] Maarten De Raedt, Fréderic Godin, Thomas Demeester, and Chris Develder. 2023. IDAS: Intent discovery with abstractive summarization. *arXiv preprint arXiv:2305.19783* (2023).

[7] Liesbeth Degand and Philippe Muller. 2020. Introduction to the special issue on dialogue and dialogue systems. *Traitement Automatique des Langues* 61, 3 (2020), 7–15.

[8] Jairo Diaz-Rodriguez. 2025. k-LLMmeans: Summaries as Centroids for Interpretable and Scalable LLM-Based Text Clustering. *arXiv e-prints* (2025), arXiv–2502.

[9] Lu Fan, Jiashu Pu, Rongsheng Zhang, and Xiao-Ming Wu. 2025. Lanid: Llm-assisted new intent discovery. *arXiv preprint arXiv:2503.23740* (2025).

[10] Xibin Gao, Radhika Arava, Qian Hu, Thahir Mohamed, Wei Xiao, Zheng Gao, and Mohammad AbdelHady. 2021. Graphire: Novel intent discovery with pretraining on prior knowledge using contrastive learning. *Technical Report* (2021).

[11] K Chidananda Gowda and GJPR Krishna. 1978. Agglomerative clustering using the concept of mutual nearest neighbourhood. *Pattern recognition* 10, 2 (1978), 105–112.

[12] Kai Han, Andrea Vedaldi, and Andrew Zisserman. 2019. Learning to discover novel visual categories via deep transfer clustering. In *Proceedings of the IEEE/CVF international conference on computer vision*. 8401–8409.

[13] Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. 2017. Learning to cluster in order to transfer across domains and tasks. *arXiv preprint arXiv:1711.10125* (2017).

[14] Yen-Chang Hsu, Zhaoyang Lv, Joel Schlosser, Phillip Odom, and Zsolt Kira. 2019. Multi-class classification without multi-class labels. *arXiv preprint arXiv:1901.00544* (2019).

[15] Haoyang Li, Xin Wang, Ziwei Zhang, Jianxin Ma, Peng Cui, and Wenwu Zhu. 2021. Intention-aware sequential recommendation with structured intent transition. *IEEE Transactions on Knowledge and Data Engineering* 34, 11 (2021), 5403–5414.

[16] Yinfeng Li, Chen Gao, Xiaoyi Du, Huazhou Wei, Hengliang Luo, Depeng Jin, and Yong Li. 2022. Automatically discovering user consumption intents in meituan.

[17] In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3259–3269.

[17] Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. 2021. Contrastive clustering. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 8547–8555.

[18] Jinggui Liang, Lizi Liao, Hao Fei, and Jing Jiang. 2024. Synergizing large language models and pre-trained smaller models for conversational intent discovery. In *Findings of the Association for Computational Linguistics ACL 2024*. 14133–14147.

[19] I-Fan Lin, Faegheh Hasibi, and Suzan Verberne. 2025. SPILL: Domain-Adaptive Intent Clustering based on Selection and Pooling with Large Language Models. *arXiv preprint arXiv:2503.15351* (2025).

[20] Ting-En Lin, Hua Xu, and Hanlei Zhang. 2020. Discovering new intents via constrained deep adaptive clustering with cluster refinement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 8360–8367.

[21] James B McQueen. 1967. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.* 281–297.

[22] G Ayorkor Mills-Tettey, Anthony Stentz, and M Bernardine Dias. 2007. The dynamic hungarian algorithm for the assignment problem with changing costs. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-27* 7 (2007).

[23] Yutao Mou, Keqing He, Yanan Wu, Pei Wang, Jingang Wang, Wei Wu, Yi Huang, Junlan Feng, and Weiran Xu. 2022. Generalized intent discovery: Learning from open world dialogue system. *arXiv preprint arXiv:2209.06030* (2022).

[24] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[25] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[26] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*. Springer, 232–241.

[27] Aaron Rodrigues, Mahmood Hegazy, and Azzam Naeem. 2025. From Intent Discovery to Recognition with Topic Modeling and Synthetic Data. *arXiv preprint arXiv:2505.11176* (2025).

[28] Juan A Rodriguez, Nicholas Botzer, David Vazquez, Christopher Pal, Marco Pedersoli, and Issam Laradji. 2024. Intentgpt: Few-shot intent discovery with large language models. *arXiv preprint arXiv:2411.10670* (2024).

[29] Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal* 27, 3 (1948), 379–423.

[30] Xiang Shen, Yinge Sun, Yao Zhang, and Mani Najmabadi. 2021. Semi-supervised intent discovery with contrastive learning. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*. 120–129.

[31] Xiaoshuai Song, Keqing He, Pei Wang, Guanting Dong, Yutao Mou, Jingang Wang, Yunsen Xian, Xunliang Cai, and Weiran Xu. 2023. Large language models meet open-world intent discovery and recognition: An evaluation of chatgpt. *arXiv preprint arXiv:2310.10176* (2023).

[32] 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. *arXiv preprint arXiv:2212.09741* (2022).

[33] Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. 2022. Generalized category discovery. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 7492–7501.

[34] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research* 11, 12 (2010).

[35] Hongtao Wang, Taiyan Zhang, Renchi Yang, and Jianliang Xu. 2025. Cequel: Cost-Effective Querying of Large Language Models for Text Clustering. In *CIKM*.

[36] Xiang Wang, Buyue Qian, and Ian Davidson. 2014. On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery* 28 (2014), 1–30.

[37] Yu Wang, Zhengyang Wang, Hengrui Zhang, Qingyu Yin, Xianfeng Tang, Yinghan Wang, Danqing Zhang, Limeng Cui, Monica Cheng, Bing Yin, et al. 2023. Exploiting intent evolution in e-commercial query recommendation. In *KDD*. 5162–5173.

[38] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*. PMLR, 478–487.

[39] Fanghua Ye, Meng Fang, Shenghui Li, and Emine Yilmaz. 2023. Enhancing conversational search: Large language model-aided informative query rewriting. *arXiv preprint arXiv:2310.09716* (2023).

[40] Dejiao Zhang, Feng Nan, Xiaokai Wei, Shangwen Li, Henghui Zhu, Kathleen McKeown, Ramesh Nallapati, Andrew Arnold, and Bing Xiang. 2021. Supporting clustering with contrastive learning. *arXiv preprint arXiv:2103.12953* (2021).

[41] 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*, Vol. 35. 14365–14373.

[42] Hanlei Zhang, Hua Xu, Xin Wang, Fei Long, and Kai Gao. 2023. A clustering framework for unsupervised and semi-supervised new intent discovery. *IEEE Transactions on Knowledge and Data Engineering* 36, 11 (2023), 5468–5481.
[43] Yuwei Zhang, Haode Zhang, Li-Ming Zhan, Xiao-Ming Wu, and Albert Lam. 2022. New intent discovery with pre-training and contrastive learning. *arXiv preprint arXiv:2205.12914* (2022).
[44] Yunhua Zhou, Guofeng Quan, and Xipeng Qiu. 2023. A probabilistic framework for discovering new intents. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 3771–3784.
[45] Henry Peng Zou, Siffi Singh, Yi Nian, Jianfeng He, Jason Cai, Saab Mansour, and Hang Su. 2025. Glean: Generalized category discovery with diverse and quality-enhanced llm feedback. *arXiv preprint arXiv:2502.18414* (2025).

A Notation

Table 5 provides a summary of the key notations used throughout this paper.

Table 5: Summary of notations.

Symbol	Description
$\mathcal{Y}, \mathcal{Y}_k, \mathcal{Y}_u$	Sets of all, known, and unknown intents.
K, M	Total and known intent counts.
$\mathcal{D}_l, \mathcal{D}_u$	Sets of labeled and unlabeled utterances.
$\mathcal{D}_{train}, \mathcal{D}_{test}$	Training and testing sets.
X, x_i	Set of utterances and the i-th utterance.
$\mathcal{X}, \mathbf{x}_i$	Set of utterance embeddings and the embedding for x_i.
y_i	The i-th utterance's intent.
N, d	Number of utterances and embedding dimension.
T, t	Total iterations and current iteration index.
$PTE(\cdot)$	Pre-trained Text Encoder.
$LLM(\cdot)$	Large Language Model for generation and refinement.
p_{smry}	The prompt template for summary generation.
$C_k, \boldsymbol{\mu}_k$	The k-th cluster and its Euclidean centroid.
$\boldsymbol{\theta}_k, s_k$	The semantic centroid and textual summary for cluster C_k.
$\boldsymbol{\theta}_{nbr(k)}$	The nearest neighboring semantic centroid to $\boldsymbol{\theta}_k$.
$\mathcal{S}_k$	Set of representative exemplars from cluster C_k.
$f(\cdot)$	The clustering cost function.
$\cos(\cdot)$	Cosine similarity between two utterance embeddings.
$\mathcal{L}^{ED}$	The Euclidean distance cost.
$\mathcal{L}^{SC}, \alpha$	The semantic cohesion cost and its weight.
$\mathcal{L}^{SS}, \beta$	The semantic separation cost and its weight.
p_{ref}	The prompt template for utterance refinement.
$\mathcal{H}$	Set of identified hard samples for refinement.
$H(\cdot), \delta$	Shannon entropy function and the number of hard samples.
K_{nbr}	The number of neighboring clusters for HSR context.
$\tilde{x}_i, \tilde{\mathbf{x}}_i$	Refined utterance and its new embedding.
$s_h, \mathcal{S}_h$	Summary and exemplars of a sample's home cluster.
$\{\boldsymbol{\mu}_k^0\}_{k=1}^K$	The initial Euclidean centroids.
$\{\boldsymbol{\mu}_j^*\}_{j=1}^M$	Seed centroids from labeled data $\mathcal{D}_l$.
$\mathcal{L}^{SP}, \gamma$	The supervised cost and its weight.
$Mean(\cdot)$	The mean of a set of embeddings.

B Baseline Repositories

Table 6 lists the public code repositories used for the baseline methods in our experiments.

Table 6: Code repositories for baselines.

Baseline	Code Repository
SAE/DEC	https://github.com/piiswrong/dec
DCN	https://github.com/boyangumn/DCN
CC	https://github.com/XLearning-SCU/2021-AAAI-CC
SCCL	https://github.com/amazon-science/sccl
KCL/MCL	https://github.com/GT-RIPL/L2C
DTC	https://github.com/k-han/DTC
CDAC+	https://github.com/thuiar/CDAC-plus
GCD	https://github.com/sgvaze/generalized-category-discovery
DeepAligned	https://github.com/HanleiZhang/DeepAligned-Clustering
MTP-CLNN	https://github.com/fanolabs/NID_ACLARR2022
LatentEM	https://github.com/zyh190507/Probabilistic-discovery-new-intents
USNID	https://github.com/thuiar/TEXTOIR
SDC	https://github.com/Lackel/SDC
LANID	https://github.com/floatSDSDS/LANID

C Hyperparameter Settings

Table 7 details the hyperparameter configurations. Across all settings, we employ USNID as $PTE(\cdot)$ and GPT-4o-Mini as $LLM(\cdot)$. We use fixed parameters for DCS and HSR: $|\mathcal{S}_k| = 10$, $\delta = 10$, and $K_{nbr} = 10$. For brevity, these constant values are omitted from the main hyperparameter table.

We can observe that MMR proves to be the most effective choice in the majority of cases. Notably, for the semi-supervised experiments on the BANKING and CLINC datasets, we opt for a similarity-based mapping strategy instead of one reliant on the LLM. This is because these datasets feature a large number of known intents (over a hundred). The LLM struggles to fully comprehend and accurately map such a wide range of intents while strictly adhering to the required output format. Consequently, a direct similarity-based mapping provides a more robust and effective solution in high-cardinality scenarios.

Table 7: Hyperparameter settings for NILC.

Setting	Dataset	Selection Strategy	T	α	β	γ	Mapping Strategy
Unsupervised	BANKING	MMR	3	0.5	0.5	–	–
	CLINC	MMR	3	0.5	0.3	–	–
	DBPedia	MMR	3	0.9	0.5	–	–
	M-CID	NN	2	0.3	0.3	–	–
	SNIPS	MMR	3	0.3	0.5	–	–
	StackOverflow	MMR	3	0.9	0.1	–	–
Semi-Supervised	BANKING	MMR	3	0.9	0.3	0.5	Similarity-based
	CLINC	MMR	3	0.5	0.5	0.5	Similarity-based
	DBPedia	MAD	3	0.5	0.5	0.5	LLM-based
	M-CID	NN	3	0.3	0.1	0.5	LLM-based
	SNIPS	MMR	3	0.5	0.5	0.5	LLM-based
	StackOverflow	MMR	3	0.9	0.7	0.1	LLM-based

D Selection Strategies for $\mathcal{S}_k$

K-Means++. This strategy adapts the seeding procedure of K-Means++ to select a geometrically diverse set of exemplars. The selection is iterative. Let $\mathcal{S}_k^{(i)}$ be the set of i selected exemplars. The first exemplar, $\mathbf{x}_1$, is chosen uniformly at random from C_k to form $\mathcal{S}_k^{(1)}$. For $i = 2, \ldots, |\mathcal{S}_k|$, each subsequent exemplar $\mathbf{x}_i$ is chosen from the remaining embeddings $C_k \setminus \mathcal{S}_k^{(i-1)}$ with a probability proportional $G(\mathbf{x}_i)$ to its minimum squared Euclidean distance to the set of already-selected exemplars $\mathcal{S}_k^{(i-1)}$:

$$G(\mathbf{x}_i) = \frac{\min_{\mathbf{x}_s \in \mathcal{S}_k^{(i-1)}} ||\mathbf{x}_i - \mathbf{x}_s||^2}{\sum_{\mathbf{x}_j \in C_k \setminus \mathcal{S}_k^{(i-1)}} \min_{\mathbf{x}_s \in \mathcal{S}_k^{(i-1)}} ||\mathbf{x}_j - \mathbf{x}_s||^2} \quad (14)$$

This method is designed to maximize the diversity of $\mathcal{S}_k$, ensuring broad coverage of the cluster's semantic space.

Mean Average Distance (MAD). The MAD strategy identifies exemplars from the cluster's periphery by selecting those that are,

Table 8: Evolution of s_{10} for C_{10} on M-CID.

Iteration	Summary s_{10} for Cluster C_{10}
1	What cleaning and disinfecting practices are effective in preventing the spread of COVID-19 on surfaces?
2	How long does the coronavirus survive on various surfaces and materials, and what cleaning practices are recommended?
3	What are the best cleaning practices and precautions to prevent COVID-19 transmission from surfaces and packages?

Task:
Create a strict one-to-one mapping from each 'Predefined Intent' to the single most appropriate 'Cluster Summary'.
Rules:
- Every Predefined Intent must be mapped to exactly one Cluster Summary.
- A Cluster Summary can only be used for one mapping.
- You must find the best possible pair for every intent, even if the match is not perfect.
Inputs:
Predefined Intent List: {known_labels_list}
Cluster Summaries to Map: {summaries_list}
Output Format:
Provide the mapping using the format 'Predefined Intent -> Cluster X'.
Output ONLY the mapping lines.
Mapping:

Figure 16: Prompt template for LLM-based mappings.

on average, most dissimilar from other members of the cluster. We select the set $\mathcal{S}_k$ by maximizing the mean distance:

$$\mathcal{S}_k = \underset{\mathcal{S} \subset C_k, |\mathcal{S}|=|\mathcal{S}_k|}{\arg\max} \sum_{\mathbf{x}_i \in \mathcal{S}} \frac{1}{|C_k| - 1} \sum_{\mathbf{x}_j \in C_k, j \neq i} ||\mathbf{x}_i - \mathbf{x}_j|| \quad (15)$$

The theoretical justification is that these boundary points are crucial for defining the cluster's extent and improving its separation from neighboring clusters.

Maximal Marginal Relevance (MMR). MMR provides a formal framework for balancing relevance to the cluster's central theme with the diversity of the selected exemplars. After an initial exemplar is chosen based on maximum similarity to the geometric centroid $\boldsymbol{\mu}_k$, subsequent exemplars are selected iteratively to maximize the following objective function:

$$\underset{\mathbf{x}_j \in C_k \setminus \mathcal{S}_k}{\arg\max} \left[\cos(\mathbf{x}_j, \boldsymbol{\mu}_k) - \max_{\mathbf{x}_s \in \mathcal{S}_k} \cos(\mathbf{x}_j, \mathbf{x}_s) \right] \quad (16)$$

This ensures that $\mathcal{S}_k$ is composed of exemplars that are both highly representative and non-redundant.

Nearest Neighbors (NN). This strategy selects the most central and prototypical instances of the cluster. The centrality $C(\mathbf{x}_i)$ of an embedding is defined as its cumulative similarity to all other embeddings within the cluster:

$$C(\mathbf{x}_i) = \sum_{\mathbf{x}_j \in C_k, j \neq i} \cos(\mathbf{x}_i, \mathbf{x}_j) \quad (17)$$

The set $\mathcal{S}_k$ is formed by the utterances corresponding to the embeddings with the highest centrality scores. The premise is that

the most central points are the most faithful representatives of the underlying intent.

E Mapping Strategies for π^t

Embedding-based Mapping. This approach matches the known seed centroids $\{\boldsymbol{\mu}_j^*\}_{j=1}^M$ to the current semantic centroids $\{\boldsymbol{\theta}_k^t\}_{k=1}^K$ by solving the assignment problem that minimizes cosine dissimilarity, using the Hungarian algorithm:

$$\min_{\pi^t} \sum_{m=1}^M \left(1 - \cos(\boldsymbol{\mu}_j^*, \boldsymbol{\theta}_{\pi^t(j)}^t) \right) \quad (18)$$

LLM-based Mapping. This strategy employs the LLM to perform a direct semantic mapping between the known intent labels $\mathcal{Y}_k$ and the generated cluster summaries $\{s_k^t\}_{k=1}^K$. As detailed in Fig. 16, the prompt p_{map} is designed to constrain the LLM to behave like an optimal assignment algorithm to create a strict one-to-one mapping by enforcing rules that require every intent to be matched with a unique cluster summary. This process compels the LLM to find the best possible pairing for each intent:

$$\pi^t = \text{LLM}\left(p_{\text{map}}, \mathcal{Y}_k, \{s_k^t\}_{k=1}^K \right) \quad (19)$$

F Case Studies
F.1 Evolution of Semantic Centroids

To understand how NILC iteratively refines the semantic understanding of each cluster, we can dive into the evolution of the LLM-generated cluster summaries, which act as semantic centroids. Table 8 presents a case study from M-CID, tracking the summary s_{10} for C_{10} over 3 iterations.

Initially, the cluster is summarized with a broad question about general cleaning practices. As the cluster assignments and embeddings are refined, the summary evolves to become more specific and detailed. By Iteration 2, it narrows its focus to the survivability of the virus on surfaces. Finally, in Iteration 3, the summary crystallizes into a precise and actionable utterance about the "best practices and precautions" for preventing surface transmission. This progressive refinement demonstrates how NILC's iterative process enhances the coherence and specificity of the discovered intents.

F.2 Successful Refinement of Hard Samples

To illustrate the utility of HSR, we present a qualitative example on StackOverflow. HSR clarifies ambiguous utterances by leveraging the LLM's contextual understanding.

For instance, the utterance x_i = "Confusion regarding laziness" is ambiguous. It was initially misclassified into a cluster about LINQ queries because "laziness" can relate to deferred execution, which is not the utterance's core intent. NILC identifies this high-uncertainty sample and provides the LLM with the context of its assigned and neighboring clusters, as detailed in Table 9. Note that the "True Cluster" is shown for illustrative purposes; the LLM only receives the "home" cluster and neighboring clusters as the context, not its ground-truth identity.

The LLM analyzes the competing intents and, recognizing that "laziness" is a core concept in Haskell, rewrites the utterance into

Table 11: Analysis of Semi-Supervised Mapping Strategies.

Dataset	Mapping Strategy	NMI	ARI	ACC
DBPedia	Similarity-based	89.41	83.96	91.43
	LLM-based	89.36	84.19	91.57
M-CID	Similarity-based	81.49	70.56	83.09
	LLM-based	83.06	72.48	84.53
StackOverflow	Similarity-based	80.13	75.89	86.85
	LLM-based	80.53	76.47	87.18

Table 12: Analysis of Representative Sampling Methods.

Dataset	Selection Strategy	NMI	ARI	ACC
DBPedia	MMR	89.36	84.19	91.57
	MAD	89.99	84.88	92.00
	NN	88.90	83.81	91.43
	K-Means++	89.36	84.03	91.43
M-CID	MMR	83.06	72.48	84.53
	MAD	81.83	70.64	83.09
	NN	83.36	73.12	85.10
	K-Means++	82.62	71.66	83.67
StackOverflow	MMR	80.53	76.47	87.18
	MAD	80.19	75.86	86.77
	NN	80.33	76.15	86.93
	K-Means++	80.28	76.16	86.95

Table 9: An example of the HSR process for an ambiguous utterance on StackOverflow.

Component	Content
Hard Sample (x_i)	Confusion regarding laziness
Assigned Cluster (C_4)	**Summary (s_4):** What are the various techniques and best practices for effectively using LINQ to query and manipulate data, including handling distinct values, dynamic queries, joins, and return types?
True Cluster (C_6)	**Summary (s_6):** What are some common challenges and best practices when working with Haskell, including syntax, error handling, and functional constructs?
LLM Task	Given the context, analyze the best fit for the utterance and rewrite it to be an unambiguous exemplar of that theme.
Refined Utterance ($\tilde{x}_i$)	Understanding laziness in functional programming languages like Haskell

Table 10: Comparison of mapping strategies on DBPedia.

Mapping Strategy	Summary s_k for Cluster C_k	Mapped Known Intent
Similarity-based	C_0: Books and Publications	WrittenWork
	C_1: Notable Individuals	OfficeHolder
	C_2: Plant Species	Plant
	C_3: Organisms	Animal
	C_4: Historic and Cultural Institutions	Building
	C_6: Historical Vehicles and Vessels	MeanOfTransportation
	C_8: Diverse Companies and Organizations	Company
	C_9: Films	Film
	C_{10}: Geographical Features	NaturalPlace
	C_{13}: **Professional Athlete**	**Artist**
LLM-based	C_0: Literary Works	WrittenWork
	C_1: Notable Individuals	OfficeHolder
	C_2: Plant Species	Plant
	C_3: Organisms	Animal
	C_4: Historic and Cultural Institutions	Building
	C_5: **Music Albums and Compilations**	**Artist**
	C_6: Historical Vehicles and Vessels	MeanOfTransportation
	C_8: Corporations and Organizations	Company
	C_9: Films	Film
	C_{10}: Geographical Features	NaturalPlace

a clear and specific question. The new embedding $\tilde{x}_i$ for the refined utterance has a much lower clustering cost and is confidently reassigned to the correct Haskell-related cluster. This case study demonstrates how HSR actively corrects the data manifold, improving cluster cohesion and separation by resolving ambiguity.

F.3 Comparison of Two Mapping Strategies

To showcase the superiority of our LLM-based mapping approach over the traditional similarity-based method, we present a detailed comparison of the mappings generated for DBPedia, as shown in Table 10.

While both methods successfully map many clusters, the similarity-based approach, which relies on cosine distance between centroids, makes a critical error. It incorrectly maps C_{13}, summarized as "Professional Athlete", to the known intent "Artist". Although athletes can be metaphorically considered "artists" of sports, this is not the correct semantic relationship on DBPedia. This error underscores a fundamental limitation of relying purely on embedding similarity in a Euclidean space; such an approach is confined to geometric proximity and lacks the awareness of semantic context, possibly causing it to be misled by abstract or metaphorical connections that an LLM, with its richer world knowledge, can correctly disambiguate.

In contrast, our LLM-based method leverages its world knowledge and reasoning capabilities. It correctly discerns that "Professional Athlete" does not fit the "Artist" category and instead makes the more semantically sound decision to map C_5 ("Music Albums and Compilations") to "Artist". This leads to a more effective injection of semi-supervised signals, a more accurate must-links constraint relationship, and, ultimately, a more accurate final clustering.

G Empirical Studies of Mapping and Sampling Strategies

We further analyze the specific strategies used to represent clusters. Table 11 compares the LLM-based mapping strategy against a traditional similarity-based approach for semi-supervised NID. The LLM-based strategy consistently outperforms the similarity-based one, particularly on the M-CID dataset, where it yields a 1.92% improvement in ARI. This shows that LLMs can capture the semantic alignment between known intent labels and cluster summaries more effectively than simple embedding similarity. In Table 12, we analyze different representative sampling methods for generating cluster summaries. While all methods perform well, the MAD, NN, and MMR strategies show slight advantages on DBPedia, M-CID, and StackOverflow, respectively, suggesting that the optimal sampling strategy can be dataset-dependent.