

RESEARCH

Open Access



Label space reduction for transductive zero-shot classification with large language models

Nathan Vandemoortele^{1*}, Bram Steenwinckel¹, Femke Ongena¹ and Sofie Van Hoecke¹

*Correspondence:
Nathan Vandemoortele
Nathan.Vandemoortele@UGent.be
¹IDLab, Ghent University-imec,
Technologiepark-Zwijnaarde 126,
9050 Gent, Belgium

Abstract

Zero-shot classification enables models to assign labels to unseen classes without task-specific training and has become increasingly effective with the advent of Large Language Models (LLMs). However, existing zero-shot classification approaches typically present all candidate labels as a flat, unstructured list within the prompt. As the number of classes grows, this formulation exacerbates well-known limitations of LLMs, including attention dilution and positional bias, ultimately degrading classification performance. We present Label Space Reduction (LSR), a test-time training method for transductive zero-shot LLM classification. LSR uses LLM-generated pseudo-labels to iteratively refine the classification label space by systematically ranking and reducing candidate classes, enabling the model to concentrate on the most relevant options. By leveraging unlabeled data with the statistical learning capabilities of data-driven models, LSR dynamically optimizes the label space representation at test time. Our experiments across seven benchmarks demonstrate that LSR improves macro-F1 scores by an average of 7.0% (up to 14.2%) with LLAMA-3.1-70B and 3.3% (up to 11.1%) with CLAUDE-3.5-SONNET compared to standard zero-shot classification baselines. To reduce the computational overhead of LSR, which requires an additional LLM call at each iteration, we propose distilling the model into a probabilistic classifier, enabling lightweight deployment without repeated LLM calls.

Keywords Large language models, Zero-shot classification, Transductive learning, Test-time training, Label space reduction

1 Introduction

Zero-shot learning is a powerful paradigm that enables machine learning models to handle new tasks without requiring task-specific training. Instead, these models transfer knowledge from previously learned tasks to tackle novel challenges [1]. A key application of this paradigm is zero-shot classification. In this approach, models can identify examples from classes they've never seen during training by making connections to known classes through semantic embeddings or attribute spaces [1, 2]. This capability addresses a major limitation of traditional supervised learning: the need for labeled training data for every target class. As a result, zero-shot classification is particularly



© The Author(s) 2026. **Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

valuable in real-world applications where collecting labeled examples for all possible classes is expensive or impractical [2].

The field of Large Language Models (LLMs) has further advanced zero-shot classification by leveraging their extensive pretraining on diverse datasets. Consequently, LLMs demonstrate an unprecedented ability to understand and generate human-like text, enabling them to infer semantic relationships between input text and class descriptions without task-specific fine-tuning [3–5]. For example, when presented with a maintenance log entry like “Bearing temperature increased to 85 °C with unusual vibration patterns and metallic noise during operation,” an LLM can identify critical indicators such as “increased temperature,” “vibration,” and “metallic noise” to accurately classify the text under the “bearing failure” category, even if equipment failure classification was not explicitly included in its pretraining. This versatility highlights the practical utility of LLMs in real-world applications.

However, current approaches to LLM zero-shot classification often rely on simplistic methods that present class options as unstructured, flat lists within the prompt. This becomes increasingly problematic as the number of potential classes expands. Specifically, LLMs face critical limitations when dealing with long contexts due to key limitations, often attributed to factors such as positional bias [6, 7] and attention dilution [6], which reduces their ability to accurately distinguish between a large set of potential classes. As the context length grows, the fixed attention budget of transformers must be distributed across more tokens, as the attention weights are normalized to sum to 1. This shared attention makes it harder to focus strongly on relevant information, as demonstrated in “Needle in the haystack” tests [8, 9] where LLMs struggle to retrieve key facts from long contexts [9, 10]. The challenge becomes even more complex when multiple “needles” must not only be retrieved, but also reasoned over [9, 11]. As a result, by reducing the label space, LLMs can allocate their attention more effectively, enhancing reasoning processes such as step-by-step thinking [12, 13], where each step builds upon previous information [14].

Reducing the label space is not a new concept and is well-established in multi-label classification, generally falling into two categories: those that leverage correlations between features and labels through machine learning models [15, 16], and those that operate independently of the feature space [17]. Feature-independent approaches include methods such as label embeddings [18] or are based on hierarchical or descriptive label relationships [19]. However, all these approaches have limitations as they typically require labeled training data and are not model-agnostic [17], so they cannot be readily applied. Moreover, they generate unordered static label spaces that cannot dynamically adapt to the generative outputs of LLMs. While the concept of label space reduction has been implicitly demonstrated in RAG systems [20], addressing similar challenges, label space reduction strategies for LLMs in the context of zero-shot classification remain, to our knowledge, an underexplored area.

In this paper, we introduce Label Space Reduction (LSR), a test-time training method for transductive zero-shot classification that tackles the challenges of *large* label spaces, defined here as those containing 10–100 classes, in zero-shot multi-classification tasks. The presented method makes two key contributions: First, we develop an innovative iterative system that formulates label space reduction as a transductive test-time training procedure, using LLM-generated pseudo-labels on unlabeled target data to rank and

reduce candidate classes. This enables the LLM to focus on the most relevant options and significantly improves zero-shot multi-classification accuracy. Second, we propose a cost-efficient solution that distils the LSR process into a probabilistic classifier, addressing the computational overhead of LSR while preserving classification accuracy and enabling efficient inference.

2 Related work

Our work builds upon three key research areas, namely zero-shot classification with LLMs, extreme multi-label classification, and sampling strategies used in language model decoding.

2.1 Zero-shot classification with LLMs

While initially developed for text classification, LLMs have demonstrated remarkable zero-shot classification capabilities across diverse domains, including image classification through vision- or multimodal-language models [21–23], tabular data classification [24], time series classification [25], and even audio classification [26, 27]. Early work demonstrated that providing task descriptions and class definitions in natural language could guide models in making reasonable predictions [28, 29]. This capability was further enhanced through prompt engineering techniques [30] and the development of instruction-tuned models [31]. While these approaches have shown promising results, their performance can be sensitive to prompt design and may struggle with nuanced category distinctions. More recent work has focused on improving model reasoning through popular techniques such as chain-of-thought reasoning [12, 13] and combining robustness methods such as self-consistency [32, 33]. However, these approaches do not fundamentally address the challenges posed by large-scale classification tasks with long context requirements.

2.2 Extreme multi-label classification

There are notable connections between our work and extreme multi-label classification (XMC), a field that addresses classification problems involving thousands to millions of classes [34–36]. While XMC tasks have traditionally been approached in a supervised manner, researchers have also explored zero-shot XMC methods [37]. Early approaches relied on retrieval-based techniques such as TF-IDF [38] and BM25 [39] to match input texts with label embeddings. A major advancement occurred when Reimers and Gurevych [40] demonstrated the effectiveness of dense embeddings for retrieval, which were subsequently applied to various XMC tasks [37, 41, 42]. More recent work has focused on LLM-based approaches, emphasizing true zero-shot settings where test labels are entirely unseen, unlike generalized zero-shot settings that may include seen labels. For example, ICXML [36] adopts a hybrid generation-and-retrieval approach, first generating demonstrations with an LLM and then aligning the outputs with available labels to create a shortlist of candidates. LMTX [43] achieved state-of-the-art performance by employing an LLM as a teacher to iteratively train a bi-encoder. However, these approaches still incorporate retrieval-based components and impose an arbitrary label-space cut-off, effectively constraining the LLM's performance to that of the underlying embedding model. Moreover, we propose a solution that prioritizes the role of the LLM as a reasoner for classification, which is applicable to solving classification tasks

beyond just text classification. In this work, we do not compare with XMC benchmarks, as they are intended for extreme label spaces, which is beyond the scope of the method.

2.3 Sampling strategies

Language model decoding is the problem of generating text by selecting tokens from the predicted probability distribution of the model. Early methods like greedy selection and random sampling often led to repetitive or incoherent output. Beam search and temperature-based sampling [44] improved diversity and quality but failed to address the long tail of irrelevant and low-probability tokens, which collectively can have significant probability mass [45]. This realization led to the development of more distribution-aware sampling strategies. Top-k sampling [46] limits choices to the k most probable tokens, while nucleus (Top-p) sampling [45] selects tokens whose cumulative probability exceeds p . More recently, Min-p sampling [47] enforces a minimum probability threshold to maintain coherence. Adaptive Semantic-Aware Typicality Sampling (ASTS) [48] further advances this line by dynamically adjusting thresholds and integrating reward-penalty scoring to promote semantic coherence. Our implementation leverages these distribution-aware thresholds to filter out low-probability classes.

3 Methodology

Addressing the shortcomings of current methods, we present our LSR methodology, which consists of three distinct phases that operate in an iterative test-time training process. In this section, we detail each phase of the process, as illustrated in Fig. 1.

3.1 Problem definition

Let $\mathcal{X}$ denote the input space and $\mathcal{Y} = \{1, \dots, K\}$ be a finite label space comprising K distinct classes. We frame LSR as zero-shot classification in a transductive test-time training setting. Given an unlabeled dataset $\mathcal{D} = \{x_1, \dots, x_n\}$ where $x_i \in \mathcal{X}$ consists of n samples, we aim to improve the zero-shot classification performance of an

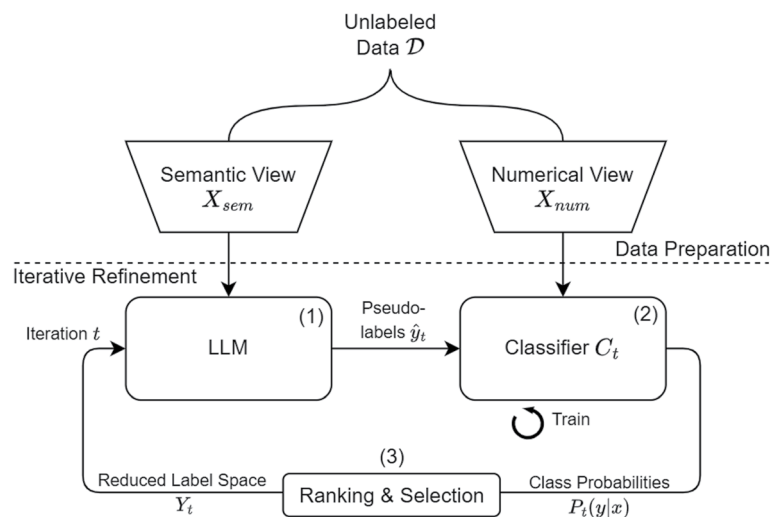


Fig. 1 Illustration of the proposed methodology. **1** The LLM categorizes the data by selecting labels from the full label set. **2** With these pseudo-labels a classifier is trained on the data's numerical representation, which generates probabilities for each class. **3** The labels are then ranked and filtered based on an adaptive threshold to form a reduced label set, which is fed back to the LLM, initiating the next iteration

LLM for these n samples without using ground-truth labels, considering the following constraints:

- *Single-label Multi-Classification* Each sample $x_i \in \mathcal{D}$ corresponds to exactly one class-label $y_i \in \mathcal{Y}$
- *Closed-world Assumption* The complete set of possible unseen classes $\mathcal{Y}$ is known and fixed a priori
- *Unlabeled Data* Access to a collection of unlabeled target-domain samples $\mathcal{D}$ which is used for test-time training

These constraints represent a common and practical set of conditions found in many real-world zero-shot classification scenarios, where unlabeled target-domain data is available but labeled examples are not. The key challenge lies in optimizing classification performance without labeled examples or external knowledge (such as label descriptions or hierarchical relationships), while minimizing token usage and enabling deployment on consumer hardware.

3.2 Data representation and preprocessing

The framework operates on dual representations of the data:

1. Numerical view: $X_{num} \in \mathbb{R}^{n \times d}$, where d is the feature dimensionality
2. Semantic view: $X_{sem} = (s_1, \dots, s_n)$, where s_i is the natural language description of x_i

The aim of this dual representation approach is to find a statistical correlation between an LLM's predictions and the underlying feature distributions of a novel task. This framework can be applied to classification tasks with arbitrary data types, provided that the data can be represented both numerically and described semantically in natural language.

3.3 Initial classification

The LLM starts generating the initial set of pseudo-labels $\hat{y}_0$ for the unlabeled target dataset:

$$\hat{y}_0 = \text{LLM}(X_{sem}, \mathcal{Y}) \quad (1)$$

where the LLM's prompt includes Chain-of-Thought (CoT) reasoning [12, 13] which improves performance and makes better use of the reduced label space. Initially, all candidate labels are presented as a list of options in the prompt, and the model is tasked with categorizing the sample. The complete prompt template can be found in Appendix B.

3.4 Probabilistic classifier

At each iteration t , we train a new probabilistic classifier C_t using current pseudo-labels $\hat{y}_t$, which outputs a probability distribution over K classes:

$$P_t(y|x) = C_t(X_{num}, \hat{y}_t),$$

where $P_t(y|x) \in [0, 1]^K$ and $\sum_i P_t(y_i|x) = 1$ (2)

At each iteration, the classifier C_t is first trained using a hold-out validation strategy to prevent overfitting, then retrained on the complete dataset using the same hyperparameters.

By training C_t on the LLM's pseudo-labels, we measure the agreement between predictions derived from the underlying numerical feature representations. For example, in text classification, changes in punctuation or synonyms should not meaningfully affect the predicted class. Although we do not explicitly transform the input, we leverage the natural variation within each class across the dataset. As a result, the classifier's output distribution $P_t(y|x)$ serves as a proxy for the LLM's uncertainty at iteration t .

3.5 Ranking and selection

For any given sample $x \in D$, the labels are ranked according to the class probabilities predicted by the current classifier C_t :

$$R_t(x) = \text{argsort}_{\text{descending}}(P_t(y|x)) \quad (3)$$

Next, the labels are selected based on the adaptive threshold following Min-p sampling, which includes all labels whose probability exceeds a fraction p of the maximum probability $P_t(y|x)$:

$$\text{Min-p}(x, p, t) = \left\{ y \in Y \mid P_t(y|x) \geq p \cdot \max_{y' \in Y} P_t(y'|x) \right\} \quad (4)$$

We propose an adapted version of Min-p, denoted as Min-p⁺, which includes the LLM's current prediction $\hat{y}_t$ at iteration t during the refinement process:

$$\text{Min-p}^+(x, p, t) = \text{Min-p}(x, p, t) \cup \hat{y}_t \quad (5)$$

The threshold p can be optimized to achieve a target average dimensionality k in the reduced label space. Since p is an indirect parameter that affects the final dimensionality, it is more intuitive to control k directly. This relationship can be expressed as follows:

$$\frac{1}{n} \sum_{i=1}^n w_{i,t} |\text{Min-p}^+(x_i, p, t)| \approx k \quad (6)$$

where $w_{i,t}$ is a class-balancing weight inversely proportional to the frequency of $\hat{y}_{i,t}$ computed as follows:

$$w_{i,t} = \frac{n}{K \cdot f_{\hat{y}_{i,t}}}, \quad (7)$$

where n is the number of samples, K is the number of distinct classes, and $f_{\hat{y}_{i,t}}$ is the number of times the predicted class $\hat{y}_{i,t}$ appears in D .

By optimizing p , the label space is reduced per instance such that the average dimensionality of the label space across all samples in the dataset D is k . Consequently, the number of candidate labels adapts dynamically to each instance: samples with higher uncertainty retain more candidate labels, while more confident predictions retain fewer. Finally, by weighting samples according to class frequency, we better balance the influence of easy and difficult classes; without this weighting, the label space would be skewed toward more challenging classes.

3.6 Iterative refinement

Our iterative test-time training refinement approach, outlined in Algorithm 1, consists of multiple steps that progressively improve classification accuracy. The process begins by obtaining initial predictions from the LLM (line 4), which are used as pseudo-labels to train a baseline classifier (line 8). In each subsequent iteration, we create a reduced label space by integrating the classifier's confidence scores with the LLM's previous predictions (lines 9-11). The LLM then generates new predictions (line 6), but instead of considering all possible labels, it focuses on a filtered set of ranked candidates for each sample. This filtered approach enables more focused CoT reasoning by limiting the LLM's attention to fewer plausible options. The classifier is then retrained using these refined predictions (line 8). The process continues until a predetermined number of iterations i (lines 2-13), where a higher number of iterations typically leads to convergence, at which point no significant changes can be observed in the label space. This iterative process creates a feedback loop where the LLM and classifier help to improve each other's performance and potentially correct initial misclassifications with more focused reasoning.

The final predictions $\hat{y}_{\text{final}}$ are determined through majority voting (line 14):

$$\hat{y}_{\text{final}} = \text{mode}(\{\hat{y}_0, \dots, \hat{y}_i\}) \quad (8)$$

By aggregating predictions from previous iterations, we provide robustness against individual iteration errors. Furthermore, aggregation helps mitigate issues when the process fails to reach proper convergence and circumvents the need for complex stopping criteria.

Require: Dataset D , Label space $\mathcal{Y}$, Dimensionality k , Iterations i

Ensure: Final predictions $\hat{y}_{\text{final}}$

```

1: predictions  $\leftarrow []$ 
2: for  $t = 0$  to  $i$  do
3:   if  $t = 0$  then
4:      $\hat{y}_t \leftarrow \text{LLM}(D, \mathcal{Y})$ 
5:   else
6:      $\hat{y}_t \leftarrow \text{LLM}(D, Y_{t-1})$ 
7:   end if
8:    $C_t \leftarrow \text{TrainClassifier}(D, \hat{y}_t)$ 
9:    $P_t \leftarrow C_t(D)$ 
10:   $p \leftarrow \text{FindOptimalThreshold}(P_t, k, \hat{y}_t)$ 
11:   $Y_t \leftarrow \text{FilterLabelSpace}(P_t, p, \hat{y}_t)$ 
12:  predictions.append( $\hat{y}_t$ )
13: end for
14:  $\hat{y}_{\text{final}} \leftarrow \text{mode}(\text{predictions})$ 

```

Algorithm 1 Iterative Label Space Refinement

3.7 Practical inference

While the complete iterative test-time training methodology described above is valuable for offline evaluation and benchmarking, repeated training cycles during live inference may be impractical for real-world applications. For example, in a system that monitors customer support tickets that need to be classified and routed every few minutes, frequent model retraining would introduce unacceptable costs and latency. Therefore, we propose two alternative strategies:

3.7.1 Full inference

This approach reuses the classifiers saved from every iteration to generate rankings, following the established methodology. While this maintains the benefits of LSR and reduces computational overhead, it has two main limitations: the classifier probabilities are not calibrated with new test samples, and each iteration requires an LLM call. Due to these constraints, we do not explore this approach further in this paper.

3.7.2 Direct inference

Alternatively, we propose to train a final classifier C_{final} on the combined LLM predictions from all iterations, effectively distilling an LLM ensemble into a single classifier that remains frozen during deployment.

$$C_{final} = \text{train}(X_{num}, [\hat{y}_0, \dots, \hat{y}_i]) \quad (9)$$

For a new sample x , we can then directly use the classifier's probabilities to make predictions:

$$\begin{aligned} P(y|x) &= C_{final}(x) \\ \hat{y}_{final} &= \underset{y}{\operatorname{argmax}}(P(y|x)) \end{aligned} \quad (10)$$

Thus, direct inference only requires a trained classifier for inference, which allows for the use of larger LLMs or additional test-time scaling techniques during the iterative training phase, enhancing performance at a one-time fixed cost. Once distilled, new samples are processed by the frozen probabilistic classifier rather than through repeated autoregressive LLM inference. This provides a lightweight deployment setting, although a detailed inference-time analysis is left for future work.

4 Experimental setup

In this section we describe the datasets, evaluation metrics and implementation details used in our experiments.

4.1 Datasets

We evaluate our method on seven diverse datasets spanning different classification tasks and domains (see Table 1). The selection includes standard text classification benchmarks and a tabular classification task, with each dataset containing a substantial

Table 1 Characteristics of 7 classification datasets across multiple domains

	Amazon massive scenario	Amazon massive intent	Banking77	Mtop domain	Mtop intent	Crime	DBpedia
Task	Text class	Text class	Text class	Text class	Text class	Tabular class	Text class
Domain	Customer Service	Customer Service	Finance	Virtual Assistant	Virtual Assistant	Legal	General Knowledge
Language	English	English	English	English	English	English	English
# Samples	2974	2974	3080	4386	4386	3514	3448
# Classes	18	59	77	11	102	37	70
Balanced Classes ?	No	No	Yes	No	No	Yes	Yes

Key attributes include task type, domain, language, sample size, number of classes, and class balance distribution

number of classes to properly evaluate LSR. Due to budgetary constraints, we create subsets of some of the larger datasets.

4.1.1 MTEB datasets

Five datasets (AMAZON MASSIVE SCENARIO, AMAZON MASSIVE INTENT, BANKING77, MTOP DOMAIN, MTOP INTENT) were selected from the MTEB leaderboard,¹ covering text classification in customer service, finance and virtual assistant domains. The MTEB benchmark [49] contains numerous datasets, from which we specifically selected those with a substantial number of classes (>10). We report results on the test sets of these datasets. The second test set mentioned in Sect. 6.3, is a subset of the MTEB training data.

4.1.2 Crime dataset

The CRIME tabular dataset is derived from the San Francisco Police Department's incident reports.² It consists of a subset of up to 100 randomly selected samples per class (where available) across 37 distinct crime categories and includes seven key features: Category, Description, Day of Week, Police District, Resolution, Address, and Time. This tabular dataset presents a unique challenge in combining textual and categorical information for classification.

4.1.3 DBpedia dataset

DBPEDIA [50], derived from structured content extracted from Wikipedia, provides a classification challenge for encyclopedia articles. While a simpler 14-class variant is commonly used in NLP benchmarks [51], we specifically use a more challenging hierarchical version available on Kaggle³ that contains three levels with 9, 70, and 219 classes respectively. Our evaluation uses a balanced subset of the data, containing 50 random samples per class where available, focusing on the second level (L2) of the hierarchical structure which consists of 70 classes. Although the data has an inherent hierarchical structure, our method treats all classes independently and does not utilize these hierarchical relationships.

4.2 Model selection and configuration

4.2.1 Large language models

Our framework mainly uses LLAMA-3.1-70B-INSTRUCT [52], accessed through the DeepInfra API,⁴ due to its reproducibility and open-weight nature, which enables verification and replication of our results. We selected this model and other recent open-weight models including GEMMA-2-27B-IT [53] and QWEN-2.5-72B-INSTRUCT [54], as well as the closed-source CLAUDE-3.5-SONNET(OLD) [55], to ensure a diverse representation of model architectures and training approaches. Classification is performed in batches of 10 randomly selected samples, and we instruct the LLM to apply CoT reasoning. To ensure deterministic behavior, we set the temperature parameter to 0. The template for the prompt is shown in Appendix B.

¹<https://huggingface.co/spaces/mteb/leaderboard>.

²Police Department Incident Reports: Historical 2003 to May 2018, <https://datasf.org/opendata/>.

³<https://www.kaggle.com/datasets/danofer/dbpedia-classes>.

⁴<https://deepinfra.com>.

4.2.2 Text embeddings

To obtain numerical values for classification tasks, dense text embeddings are generated using the open-source BGE-LARGE-EN-V1.5 [56] Sentence-BERT (SBERT) model (355 M parameters, 1.34GB serialized). Through binary quantization in Sentence Transformers [57], we compress the vectors from 1024 to 128 dimensions. Although some information loss is expected, we reduce the embedding dimensionality as a trade-off to improve generalization by mitigating the curse of dimensionality and to accelerate classifier training. Moreover, binary quantization has been shown to retain very strong performance on MTEB retrieval tasks [58].

4.2.3 Classifier

For classification, we selected CATBOOST [59] as the probabilistic classifier. CATBOOST is a gradient boosting framework well suited to structured, tabular data, for which it has demonstrated state-of-the-art performance [60]. It is also effective in limited-data settings, making it appropriate for the relatively small training sets produced during distillation, while remaining lightweight, with serialized models typically under 1MB. We use the same fixed CatBoost configuration across all datasets to avoid dataset-specific tuning and keep the evaluation consistent and reproducible. Specifically, we use default parameters with $lr = 0.1$ and $depth = 3$. To mitigate overfitting, we use a stratified hold-out validation set, reserving 20% of the data for early stopping while maintaining the original class distribution. Class imbalance is addressed through class weights during training. After determining the optimal number of iterations through early stopping criteria, we retrain the classifier on the complete dataset.

Both BGE-LARGE-EN-V1.5 and CATBOOST models were evaluated on standard consumer-grade computing hardware (Intel Core i7-1365U processor @ 5.2 GHz, 16GB DDR4 RAM, running Windows 11).

4.3 Data preparation

Our methodology requires both numerical and semantic representations of the data. Here, we detail the preparation process for each view.

4.3.1 Numerical view

For text classification tasks, numerical text embeddings are generated using BAAI/BGE-LARGE-EN-V1.5, which produces 128-dimensional feature vectors as input to the classifier. For tabular classification tasks using the CRIME dataset, which contains both semantic and numerical features, we treat all semantic features as numerical through categorical encoding. Additionally, we extract the hour of the day from the Time variable as a new feature. In total, we obtain 6 features (Description, Police District, Resolution, Address, Day of Week, Hour of Day).

4.3.2 Semantic view

To create the semantic representation, we convert all contextual features, whether originally semantic or numerical, into a list of key-value pairs that can be directly incorporated into the prompt. Examples of these representations are provided with the prompt template in Appendix B.

5 Metrics and evaluation

Our goal is to measure classification performance, where we assume that all classes are of equal importance. Therefore, we evaluate our method using the Macro F1-score, as it treats all classes equally, regardless of their frequency in the dataset:

$$F1_{macro} = \frac{1}{K} \sum_{i=1}^K \frac{2(\text{precision}_i \cdot \text{recall}_i)}{\text{precision}_i + \text{recall}_i} \quad (11)$$

where K is the number of classes, and precision_i and recall_i are calculated for each class i .

It is important to note that LSR operates in a transductive test-time training setting. The method thus uses the unlabeled target dataset to train classifiers on pseudo-labels generated by the LLM, but it does not use ground-truth labels during this process. Therefore, LSR differs from standard zero-shot prompting, while preserving the absence of labeled supervision. The reported scores are computed against the ground truth labels of the test set.

6 Experiments

In this section, we present the experimental results and key findings of our proposed method. Each experiment was conducted only once to limit our expenses.

6.1 Ranking methods

We evaluate different label selection strategies derived from the proposed Min- p^+ formulation in Sect. 3.5.

For this experiment, we used the LLM (LLAMA-3.1-70B) predictions to train the CAT-BOOST classifier and obtain probability-based rankings. Performance was measured using the Hit@ k metric using Eq. 12, which is also called the “hit rate”. The hit rate represents the percentage of instances where the ground truth label appears in the reduced label space, which may vary depending on the strategy.

$$\text{Hit@}k = \max_{i=1..k} \begin{cases} 1, & r_i \in T \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

We compared five strategies:

- *Top-k*: Selects a fixed number k of highest probable labels
- *Top-p* (nucleus sampling): Selects labels until the cumulative probability exceeds threshold p
- *Min-p*: Selects labels whose probabilities exceed a fraction of the highest probability
- *Min- p^+ (uw)*: Our proposed adaptation of Min- p without class-weighted label space.
- *Min- p^+ (w)*: Min- p^+ with a class-weighted label space. For simplicity, and unless otherwise stated, we refer to Min- p^+ (w) as Min- p^+ in the remainder of this paper.

As illustrated in Fig. 2, common sampling strategies exhibit notable limitations. Top- k sampling, while straightforward, is suboptimal due to its fixed selection size and fails to adapt to varying probability distributions, often including irrelevant low-probability classes. Top- p sampling, though more distribution-aware, is limited by its hard probability cutoff, which can arbitrarily exclude classes with similar probabilities that fall just below the threshold.

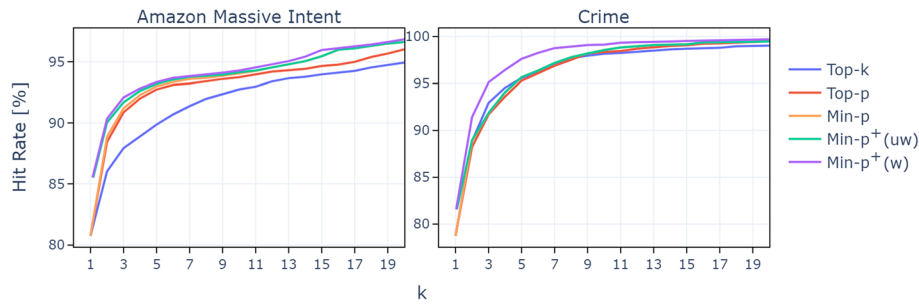


Fig. 2 Hit rates of sampling strategies across reduced label space sizes (k). Results show the percentage of true labels inside the reduced space using LLAMA-3.1-70B during the first iteration of LSR

Min-p sampling offers a more effective approach to label selection by implementing an adaptive threshold relative to the highest probability. In our methodology, we adapt Min-p sampling with two key modifications. We include the current LLM prediction to increase stability throughout the iteration (denoted as Min-p⁺(uw) in Fig. 2). Additionally, we introduce class weighting of the label space (Min-p⁺(w)) to address cases where probabilities are more uniformly distributed, as unweighted approaches tend to select many labels, skewing the resulting label space towards challenging classes. By weighting the label space according to class frequencies, we achieve better balance between easy and difficult classes, leading to improved hit rates.

6.2 Label space reduction

We analyze how varying the label space size (k) affects the performance of LLAMA-3.1-70B across multiple datasets. We measure performance using Macro F1-scores, calculated through majority voting over 15 iterations. For consistency, each experiment with different values of k begins with the same zero-shot LLM predictions, starting from randomly shuffled labels in the prompt. Our results follow the complete experimental methodology outlined in Sect. 3.

6.2.1 Iterations

Our analysis reveals two key observations from Fig. 3.

First, macro-F1 scores show an initial boost after the first iteration, with the most significant gains occurring within the first five iterations, followed by convergence around the tenth iteration. This early convergence pattern suggests that, while our framework supports extended iterations, substantial performance benefits can be achieved with minimal computational overhead. This behavior is consistent across all values of k , with the AMAZON MASSIVE INTENT dataset ($k = 1.5$) being a notable exception.

Second, we observe distinct performance patterns between different types of datasets. Datasets with rare classes, such as AMAZON MASSIVE SCENARIO/INTENT (see Table 1), exhibit more volatile performance across iterations. In contrast, datasets with more balanced class distributions (e.g., BANKING77, CRIME, and DBPEDIA) demonstrate smoother performance curves and generally achieve better results. This disparity can be attributed to the data-driven nature of the process. Since the classifier is trained on pseudo-labels generated by the LLM, errors in these pseudo-labels may affect the subsequent ranking of candidate labels. One possible source of error propagation is the early ranking stage, where the classifier has not yet benefited from iterative refinement and

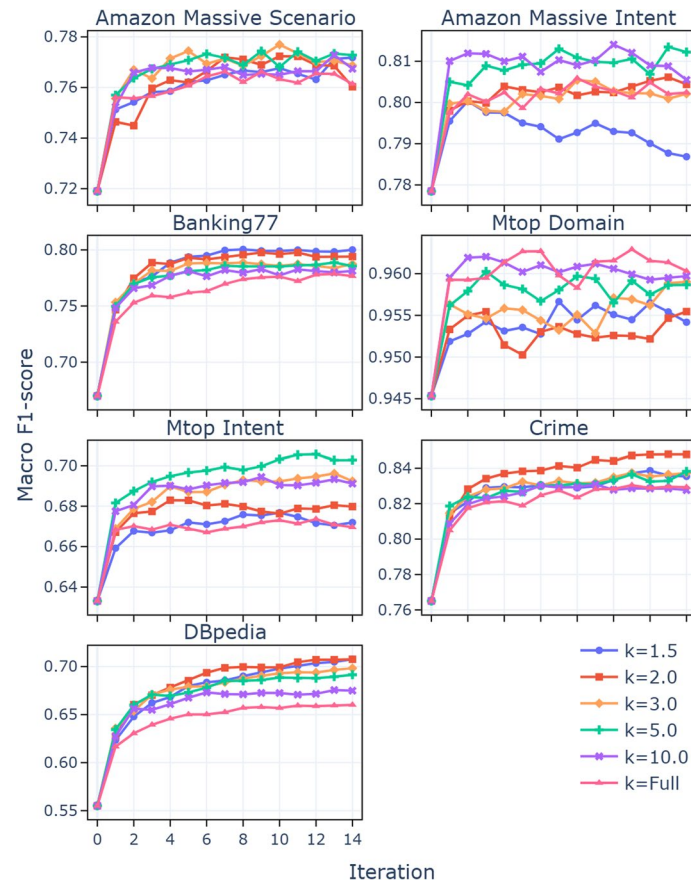


Fig. 3 Performance comparison of LSR with varying label space sizes (k). When $k = Full$, the labels are ranked but not reduced. Lines show macro-F1 scores over 15 iterations, starting from zero-shot baseline predictions (LLAMA-3.1-70B)

its rankings may still be noisy. Under-represented classes are also at risk, as the classifier has fewer pseudo-labeled examples from which to learn reliable decision boundaries. In extreme cases, the ground-truth class may be ranked too low and excluded from the selected label space altogether. When this occurs, the LLM cannot predict that class in later iterations because it is no longer presented as a candidate. The case of AMAZON MASSIVE INTENT ($k = 1.5$) exemplifies this phenomenon, where performance gains are initially rapid but prediction errors gradually accumulate, rather than resolving them. In general, this suggests that LSR may be less reliable when some classes are poorly represented in the pseudo-labeled data, for example due to class imbalance or semantic ambiguity between labels. Conversely, the smoother curves observed for balanced datasets suggest that better pseudo-label coverage across classes can lead to more stable refinement.

6.2.2 Ranking

As the reduction of the label space may not be the only source of LSR's performance improvements, we conduct an experiment in which the labels are randomly shuffled in each iteration without reducing the label space. As shown in Table 2 (denoted by $Y = RAND$), small average gains of 2% are made with a notable exception of 4.9% on AMAZON MASSIVE SCENARIO over standard zero-shot prompting (LLAMA-3.1-70B).

Table 2 Results of LSR with varying label space dimensionality (k)

	Amazon massive scenario	Amazon massive intent	Banking77	Mtop Domain	Mtop Intent	Crime	DBpedia	Average Score
Llama-3.1-70B	.719	.778	.670	.945	.633	.765	.555	.724
w/ CoT-SC	.734	.798	.696	.956	.650	.767	.579	.740
w/ LSR								
Y=Rand	.768	.801	.698	.954	.620	.784	.580	.744
k=Full	.763	.806	.771	.962	.670	.826	.654	.779
k=10.0	.768	.813	.785	.961	.692	.832	.672	.789
k=5.0	.777	.812	.788	.961	.700	.833	.686	.794
k=3.0	.779	.802	.791	.960	.692	.834	.686	.792
k=2.0	.769	.805	.799	.959	.683	.842	.697	.794
k=1.5	.771	.796	.799	.959	.676	.832	.687	.789

All macro-F1 scores are absolute values, the largest gain for each task is indicated in bold. Chain-of-Thought with Self-Consistency (CoT-SC) scores are a majority vote of 15 resamples

These results show that the order in which the labels are given influences model performance due to positional bias and demonstrates the importance of a ranked label space.

In the second part of this analysis, we quantify the performance gain of the ranking process. As shown in Table 2, our method demonstrates improvements without reduction ($k = Full$) and a substantial portion of the performance gain comes from the label ranking process, despite the fact that the LLM is not explicitly instructed to consider the label order. This effect is most evident with MTOP DOMAIN, where we observe improvements from ranking but none from reduction, given its already small label space ($K = 11$). On average, label ranking accounts for 78.6% of the maximum performance gain, while the reduction of the label space contributes to the remaining 21.4% (calculated from Table 2, $\frac{\text{Average Score}_{k=Full} - \text{Average Score}_{Llama-3.1-70B}}{\text{Average Score}_{k=2.0} - \text{Average Score}_{Llama-3.1-70B}}$). While ranking proves effective, it differs fundamentally from typical embedding-based ranking approaches [61], which produce static, non-adaptive rankings. In contrast, our classifier used in LSR is a data-driven ranker which adapts to the task at test-time. These rankings are calibrated across samples and iteratively refined based on the current top prediction from that ranking.

To illustrate this, we compared the zero-shot rankings of the classifier (CATBOOST) during LSR with $k = Full$ (which ranks without reducing the label space) to that of the embedding model used in the numerical view. For the latter, we obtain the rankings by calculating the cosine similarity between the input and label embeddings. We evaluate using the hit rate as defined in Eq. 12 and investigate the Top-k candidates.

From Fig. 4, we observe that the classifier-based rankings and classification accuracy (Hit@1) significantly outperform the embedding model across all evaluated datasets. This supports our architectural choice to leverage the best model for initial label ranking, rather than relying on embeddings. Since the quality of the results would be constrained by the initial candidate selection, any errors at this stage would be impossible to recover in subsequent refinement steps. While computationally more intensive, and at risk of retrieval issues associated with long contexts, the ability to process labels within the context window and directly leverage the LLM's knowledge base and reasoning capabilities leads to superior performance.

Figure 4 reveals that the hit rate does not reach 100% for all datasets when k is equal to the full label space, particularly evident in the DBpedia dataset. This occurs when certain

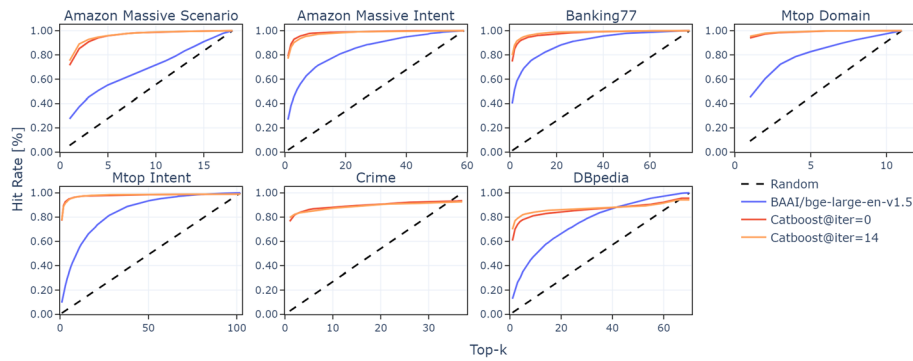


Fig. 4 Comparison of zero-shot ranking performance between classifier-based (CATBOOST) with LSR and embedding-based (BAAI/BGE-LARGE-EN-V1.5) approaches across different datasets. The hit rate (Hit@k) is shown for varying numbers of Top-k candidates, demonstrating consistently better performance of the classifier-based approach trained on LLM (LLAMA-3.1-70B) pseudo-labels. Note, the embedding approach is only applicable to text classification tasks

classes are not predicted by the LLM and thus the classifier cannot incorporate them during training. Furthermore, we observe that while the rankings are properly optimized among the top candidates, this optimization does not extend across all class-labels in the label space, unlike what we observe for the embedding-based rankings. While our current implementation with LSR prioritizes the top candidates, future work could focus on improving the consistency of rankings across the entire label space.

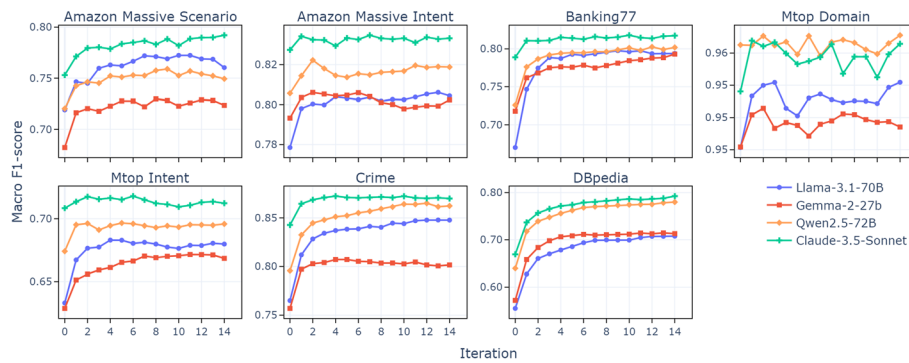
6.2.3 Selection

Our experiments demonstrate that the optimal value of k varies across tasks. We hypothesize that this variation in optimal label space size is related to the semantic similarity between labels. When labels are more semantically similar, a larger label space may be necessary to effectively distinguish between closely related options. This hypothesis is supported by the difference observed in Table 2 between AMAZON MASSIVE SCENARIO and AMAZON MASSIVE INTENT classifications, where INTENT, being a more fine-grained classification of SCENARIO, requires a larger optimal k . Through empirical analysis across datasets, we found that k reaches an optimum at $k = 2$ and $k = 5$, consistently yielding the highest average performance improvements. The range $k \in [2, 5]$ appears to provide a good balance between exploration and exploitation, suggesting these values as practical defaults for zero-shot applications.

Majority Voting The use of majority voting across iterations in our method resembles aspects of Self-Consistency (SC) [32], which has similarly demonstrated improvements in zero-shot performance at test time. Therefore, we conducted a comparison with CoT-SC using a temperature of 0.7 and 15 resamples to ensure equal computational budget before majority voting, as shown in Table 2. While CoT-SC showed a modest average absolute improvement of 1.6% compared to zero-shot prompting (LLAMA-3.1-70B), our method achieved substantially higher gains of up to 7.0% on average. Note that these averages are indicative and highly dependent on the specific task, as shown in the table. We hypothesize that CoT-SC could be complementary to our technique, as its performance gains originate from a different source, i.e., from sampling different reasoning paths of the model with a high temperature [32]. With additional resources, further experiments could evaluate the extent of this complementarity.

Table 3 Results of LSR with various LLMs. We report the gain in macro-F1 scores between zero-shot predictions and after 15 iterations with $k = 2$

Model	Amazon massive scenario	Amazon massive intent	Banking77	Mtop domain	Mtop intent	Crime	DBpedia
Llama-3.1-70B	.719	.778	.670	.945	.633	.765	.555
w/ LSR	+0.050	+0.027	+0.129	+0.014	+0.050	+0.077	+0.142
Gemma-2-27b	.682	.793	.718	.945	.629	.757	.572
w/ LSR	+0.047	+0.011	+0.063	+0.006	+0.041	+0.051	+0.140
Qwen2.5-72B	.720	.806	.726	.961	.674	.796	.639
w/ LSR	+0.038	+0.012	+0.072	+0.002	+0.021	+0.065	+0.130
Claude-3.5-Son	.753	.827	.789	.954	.708	.843	.669
w/ LSR	+0.035	+0.009	+0.027	+0.009	+0.009	+0.030	+0.111

**Fig. 5** Performance of LSR ($k = 2$) with various LLMs. Lines show macro-F1 scores over 15 iterations, starting from zero-shot baseline predictions

6.2.4 Comparison across multiple large language models

We conducted a comprehensive evaluation of LSR with different sized LLMs and capabilities to demonstrate its general applicability. In this setup, we set $k = 2$, which we determined to be an optimal value in Sect. 6.2.3. The results are reported in Table 3 and an overview is also provided in Fig. 5.

Our experiments empirically validate that LSR consistently improves performance across all LLMs and datasets tested. On average, LLAMA-3.1-70B gains 7.0%, GEMMA-2-27B gains 5.1%, QWEN2.5-72B gains 4.9%, and CLAUDE-3.5-SONNET gains 3.3% across all datasets over zero-shot prompting (LLAMA-3.1-70B). The magnitude of improvement varies, as the potential for improvement is inversely related to the baseline performance on a given task. This can be attributed to task difficulty: for example, all models score 95% or higher on MTOP DOMAIN and already saturate the benchmark. Additionally, model capability plays a role, as we generally observe smaller performance improvements from more capable models such as QWEN2.5-72B and CLAUDE-3.5-SONNET. A notable exception is DBPEDIA, where all models gain at least 11% using LSR. This leads us to hypothesize that more capable models can inherently handle larger label spaces more effectively. Conversely, less capable models show more substantial performance gains from LSR. This finding has important implications, as our method could enable smaller, more accessible models to achieve performance levels comparable to larger, more resource-intensive models.

6.3 Direct inference

This experiment evaluates whether the knowledge accumulated during transductive test-time training can be distilled into a classifier that is later applied directly to new samples. We evaluate the direct inference approach outlined in Sect. 3.7 using pseudo-labels accumulated from 15 iterations of LLAMA-3.1-70B-INSTRUCT with $k = 2.0$, we train a CATBOOST classifier with a 20% stratified hold-out set to prevent overfitting. The results are provided in Table 4.

This classifier demonstrates performance comparable to the offline methodology ($k = 2.0$ in Table 2), namely within 0.5% absolute points difference, except for MTOP INTENT where the classifier shows a 2.2% lower performance. This could be attributed to epistemic uncertainty or noise in the training data, possibly due to inconsistent labeling by the LLM or inadequate feature representations.

Interestingly, when making additional LLM predictions using the classifier's rankings, now using using Min-p sampling from Eq. 4 as there are no pseudo-labels, we observe decreased performance across most metrics (Table 4), except for MTOP INTENT which now improves, to within 0.6% of the offline performance. These results show that the classifier actually exceeds the LLM's zero-shot capabilities, as we effectively distilled knowledge from multiple LLM outputs.

While these results are promising for offline evaluation and benchmarking purposes, our goal is to deploy a frozen classifier for inference. Therefore, we evaluated the classifier's performance on a separate test set of equal size and class distribution from the original datasets to simulate a real-world deployment scenario.

Our experiments (Unseen Data in Table 4) reveal that the classifier's performance degrades substantially across most datasets in this scenario, though mostly maintaining positive improvements over zero-shot prompting (LLAMA-3.1-70B). For instance, the improvement margin decreases from +8.1% to +2.3% for CRIME, and in the most severe case, it decreases from +14.7% to -3.4% for DBPEDIA. This performance degradation can be attributed to two main factors. First, there is a distribution shift. Changes in zero-shot LLM performance indicate that the test data distribution differs from the training data, affecting the classifier's effectiveness. This sensitivity is not unique to LSR, but is a general property of data-driven models trained on a finite sample distribution. In practical settings, the unlabeled data used for LSR should therefore be representative of the target distribution, and the classifier may need to be updated when substantial distribution shifts occur. Second, there are generalization challenges. This is particularly evident in

Table 4 Results of LSR with direct inference

Model	Amazon massive scenario	Amazon massive intent	Banking77	Mtop domain	Mtop intent	Crime	DBpedia
Llama-3.1-70B	.719	.778	.670	.945	.633	.765	.555
Classifier	+.045	+.022	+.136	+.013	+.028	+.081	+.147
w/ LLM	+.037	+.022	+.111	+.004	+.044	+.072	+.142
Unseen Test							
Llama-3.1-70B	.721	.644	.666	.945	.337	.794	.570
Classifier (frozen)	+.021	+.005	+.109	-.011	+.039	+.023	-.034
w/ LLM	+.047	+.024	+.112	+.003	+.135	+.058	+.065

LLAMA-3.1-70B w/ LSR is distilled into a probabilistic classifier (CATBOOST). All macro-F1 scores show gains over zero-shot baseline predictions on two different test sets. Bold indicates the larger gain between the distilled classifier with and without an additional LLM prediction, for each test set

datasets with limited samples per class. For example, in MTOP INTENT, 41 classes have fewer than 5 samples, including 15 classes with just one sample, making it difficult for the classifier to generalize effectively.

These observations indicate a trade-off between performance and robustness. While LSR enables effective label space reduction, the distilled classifier may be more sensitive to distribution shifts when training data is scarce or not fully representative. In contrast, the original LLM-based predictions remain more robust under such shifts, as they do not rely on a fixed reduced label space. We therefore recommend ensuring a minimum number of instances per class before applying LSR (see Appendix A), with emphasis on adequately representing the underlying sample distribution rather than achieving a specific total sample size.

Interestingly, we find that adding an extra LLM inference step is now beneficial and helps mitigate performance degradation as it is less susceptible to distribution shift due to extensive pre-training and zero-shot capabilities. This is evidenced by improvements across all datasets, with gains improving from 2.3% to 5.8% for CRIME and -3.4% to +6.5% for DBPEDIA. While this approach can be effective during a cold-start, we recommend collecting sufficient samples per class (using pseudo-labels) before training the classifier, to improve generalization to unseen data. Nevertheless, challenges related to generalization and distribution shifts are inherent in data-driven approaches and should be carefully considered during real-world deployment.

7 Discussion and limitations

Our results demonstrate that optimizing the label space significantly improves LLM accuracy on classification tasks, with especially strong gains for smaller models. This suggests that, in transductive zero-shot classification, part of the difficulty lies not only in the model's ability to assign the correct label, but also in the size and order of the candidate label space. By iteratively ranking and reducing this label space, LSR enables the model to focus on increasingly relevant candidate labels. Although the process is initially computationally intensive, we address this limitation through distillation, resulting in a more lightweight classifier that outperforms standard LLM zero-shot classification.

Nevertheless, several limitations remain. First, LSR is constrained by the size of the LLM's context window, and the optimal label space size may vary depending on the semantic similarity between labels. Second, although LSR improves classification accuracy, the full efficiency-accuracy trade-off has not yet been systematically evaluated in terms of inference time, training overhead, and computational cost across the full LSR pipeline and relevant inference baselines. Third, while our experiments include a ranking-based baseline, broader comparisons with shortlist-based candidate generation are still needed to more fully contextualize the observed gains. In addition, the evaluation is limited by the absence of repeated runs across different random seeds, which would also help to further validate the observed performance gains. Fourth, the distilled classifier remains sensitive to imbalanced class distributions, which can affect performance when classes are underrepresented. Finally, the iterative process has an unrecoverable failure mode in which, if the ground-truth label is discarded at any stage, it cannot be recovered in subsequent iterations.

8 Conclusion

In this paper, we present LSR, a test-time training method for transductive zero-shot LLM classification that improves performance through iterative ranking and reduction of the label space. Our results demonstrate that optimizing the label space significantly improves LLM accuracy on classification tasks, particularly for smaller models, and that distillation can make the resulting classifier more practical for deployment.

Future work should evaluate the deployment efficiency of LSR under realistic inference settings, compare it against a broader range of candidate-shortlisting methods, and assess robustness across repeated runs. Other promising directions include exploring the impact of different classifiers and numerical representation models, developing adaptive or self-calibrating strategies for selecting parameters such as the label space size k and the number of iterations t , improving robustness under imbalanced class distributions, and extending LSR to multi-label classification to increase its general applicability to real-world classification tasks.

Appendix A: Data requirement analysis

We examine the data requirements by varying the size of the unlabeled dataset used for test-time training while maintaining the complete test set for consistent evaluation. Using stratified sampling to preserve the original class distributions, we create three representative subsets at 25%, 50%, and 75% of the initial dataset. For each subset, we reserve 20% as a validation set to prevent overfitting during training, following the methodology.

For balanced datasets (BANKING77, CRIME, DBPEDIA), we observe a clear positive correlation between data availability and performance improvement, as shown in Table 5. However, the marginal benefits vary across datasets. BANKING77 and CRIME show diminishing returns when data availability increases from 75 to 100%, indicating that our methodology can be effective even with relatively small amounts of data. Notable improvements are observed with as little as 25% of the full dataset (approximately 10 samples per class). In contrast, DBPEDIA demonstrates consistent performance gains with increased data availability (see Fig. 6), suggesting that larger unlabeled datasets could yield further improvements.

For unbalanced datasets, the relationship between data availability and performance is more complex. When minority classes have very few samples, sometimes just single instances or none after splitting the subset, performance metrics can show significant variability due to poor generalization as highlighted in Sect. 6.3. Although LSR still shows positive improvements in such cases, we recommend ensuring a minimum number of

Table 5 Effect of data size on LSR. Results show the gain in macro-F1 scores of LLaMA-3.1-70B between zero-shot predictions and after 15 iterations with $k = 2$

Data Size	Amazon Massive Scenario	Amazon Massive Intent	Banking77	Mtop Domain	Mtop Intent	Crime	DBpedia
LLaMA-3.1-70B w/ LSR	.719	.778	.670	.945	.633	.765	.555
25%	+0.034	+0.034	+0.082	+0.017	+0.046	+0.051	+0.023
50%	+0.056	+0.041	+0.105	+0.017	+0.049	+0.071	+0.072
75%	+0.060	+0.027	+0.122	+0.015	+0.005	+0.076	+0.111
100%	+0.050	+0.027	+0.129	+0.014	+0.050	+0.077	+0.142

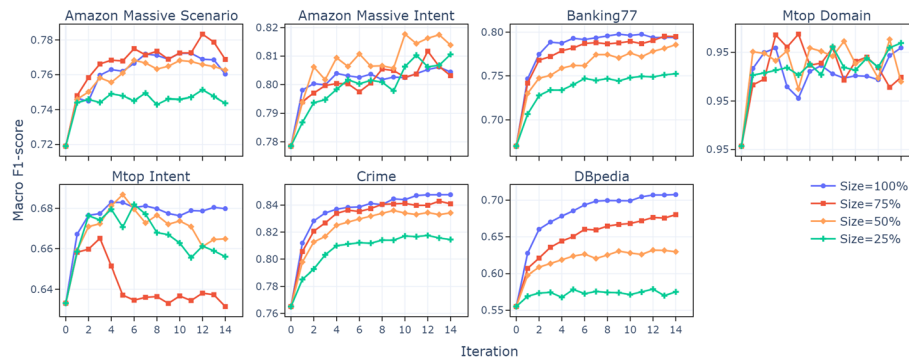


Fig. 6 Results of LSR with varying unlabeled subset sizes for training. Lines show macro-F1 scores over 15 iterations, with $k = 2$ using LLAMA-3.1-70B

instances per class before applying LSR. The key focus should be on adequately representing the underlying sample distribution rather than achieving a specific total sample size.

Appendix B: Prompt template

Prompt template for general classification tasks

```

### Context ###
Your goal is to predict the correct category given the context for each case.
The categories are: {labels}

### Instructions ###
1. Write down your thinking in a step-by-step way.
2. You MUST pick one of the suggested categories.
3. Your output must be in JSON format structured as follows:
  {"predictions": [{"Case": 0, "Analysis": "...", "Label": "..."}, ...]}
4. You must analyze all cases individually.

### Cases ###
Case 0: {feature_value_pairs}, suggestions: {subset_of_labels}
...
(Note: "suggestions" only appear after initial classification)

```

Example input Crime dataset

```

feature_value_pairs:
  Descript: FALSE IMPRISONMENT, DayOfWeek: Sunday, PdDistrict: TARAVAL,
  Resolution: NONE, Address: 2400 Block of 28TH AV, Hour: 19

subset_of_labels:
  ['KIDNAPPING', 'ASSAULT', 'OTHER OFFENSES', 'SEX OFFENSES, FORCIBLE',
  'SUSPICIOUS OCC']

```

Example input Banking77 dataset

```

feature_value_pairs:
  text: Can I make sure my card is delivered on a specific day?

subset_of_labels:
  ['card_delivery_estimate', 'card_arrival', 'get_physical_card',
  'order_physical_card', 'getting_spare_card']

```

Author contributions

N.V. conceived and designed the study, conducted the research, and developed the manuscript outline. B.S., F.O., and S.V.H. contributed to the review and critical revision of the manuscript. All authors reviewed and approved the final manuscript.

Funding

This project received funding from the European Union's Active and Assisted Living Programme (AAL) via the Emilio project (Grant Agreement No. HBC.2021.1138) and was partially funded by the Flemish Government (AI Research Program).



Co-funded by
the European Union

Data availability

The datasets analyzed during the study are publicly available from the sources cited in the Datasets section.

Declarations

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors have no conflict of interest to declare that are relevant to the content of this article.

Received: 28 January 2026 / Accepted: 27 July 2026

Published online: 06 August 2026

References

1. Wang W, Zheng VW, Yu H, Miao C. A survey of zero-shot learning: settings, methods, and applications. *ACM Trans Intell Syst Technol*. 2019. <https://doi.org/10.1145/3293318>.
2. Xian Y, Lampert CH, Schiele B, Akata Z. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Trans Pattern Anal Mach Intell*. 2017;41:2251–65.
3. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, et al. Language models are few-shot learners. *CoRR*. 2020; [arXiv:2005.14165](https://arxiv.org/abs/2005.14165).
4. Bommasani R, Hudson DA, Adeli E, Altman R, Arora S, Arx S, Bernstein MS, Bohg J, Bosselut A, Brunskill E, et al. On the opportunities and risks of foundation models, 2021. [arXiv: 2108.07258](https://arxiv.org/abs/2108.07258).
5. Wei J, Bosma M, Zhao V, Guu K, Yu AW, Lester B, Du N, Dai AM, Le QV. Finetuned language models are zero-shot learners. In: *International Conference on Learning Representations 2022*. <https://openreview.net/forum?id=gEZrGCozdqR>
6. Peysakhovich A, Lerer A. Attention sorting combats recency bias in long context language models. 2023. [arXiv:abs/2310.01427](https://arxiv.org/abs/2310.01427).
7. An C, Zhang J, Zhong M, Li L, Gong S, Luo Y, Xu J, Kong L. Why does the effective context length of LLMs fall short?, 2024. [arXiv:2410.18745](https://arxiv.org/abs/2410.18745).
8. Kamradt G. Needle In A Haystack - pressure testing LLMs. GitHub repository 2023. https://github.com/gkamradt/LLMTest_NeedleInAHaystack.
9. Hsieh C-P, Sun S, Krizan S, Acharya S, Rekish D, Jia F, Ginsburg B. RULER: What's the real context size of your long-context language models? In: *First Conference on Language Modeling 2024*. <https://openreview.net/forum?id=kloBbc765y>.
10. Liu NF, Lin K, Hewitt J, Paranjape A, Bevilacqua M, Petroni F, et al. Lost in the middle: how language models use long contexts. *Trans Assoc Comput Linguis*. 2024;12:157–73. https://doi.org/10.1162/tacl_a_00638.
11. LangChain: Multi-Needle in a Haystack. *LangChain Blog* 2023. <https://blog.langchain.dev/multi-needle-in-a-haystack/>.
12. Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2023. [arxiv:2201.11903](https://arxiv.org/abs/2201.11903).
13. Kojima T, Gu SS, Reid M, Matsuo Y, Iwasawa Y. Large language models are zero-shot reasoners. 2023. [arxiv:2205.11916](https://arxiv.org/abs/2205.11916).
14. Wu W, Wang Y, Xiao G, Peng H, Fu Y. Retrieval head mechanistically explains long-context factuality. 2024.
15. Kumar V, Pujari AK, Padmanabhan V, Kagita VR. Group preserving label embedding for multi-label classification. *CoRR*. 2018; [arXiv:1812.09910](https://arxiv.org/abs/1812.09910).
16. Huang J, Zhang P, Zhang H, Li G, Rui H. Multi-label learning via feature and label space dimension reduction. *IEEE Access*. 2020;8:1–1. <https://doi.org/10.1109/ACCESS.2020.2969238>.
17. Moyano JM, Luna JM, Ventura S. Reducing the label space a predefined ratio for a more efficient multilabel classification. *IEEE Access*. 2022. <https://doi.org/10.1109/ACCESS.2022.3192642>.
18. Tai F, Lin H-T. Multilabel classification with principal label space transformation. *Neural Comput*. 2012;24:2508–42. https://doi.org/10.1162/NECO_a_00320.
19. Charte F, Rivera AJ, Jesús MJ, Herrera F. Li-mlc: A label inference methodology for addressing high dimensionality in the label space for multilabel classification. *IEEE Trans Neural Netw Learn Syst*. 2014;25:1842–54.
20. Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, Dai Y, Sun J, Wang M, Wang H. Retrieval-augmented generation for large language models: a survey 2024. [arxiv: 2312.10997](https://arxiv.org/abs/2312.10997).
21. Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, et al. Learning transferable visual models from natural language supervision. *CoRR*; 2021. [arXiv:2103.00020](https://arxiv.org/abs/2103.00020).

22. Alayrac J-B, Donahue J, Luc P, Miech A, Barr I, Hasson Y, et al. Flamingo: a visual language model for few-shot learning. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22. Red Hook, NY, USA: Curran Associates Inc; 2022.
23. Abdelhamed A, Affi M, Go A. What do you see? enhancing zero-shot image classification with multimodal large language models. 2024. arXiv preprint. [arXiv:2405.15668](https://arxiv.org/abs/2405.15668).
24. Hegselmann S, Buendia A, Lang H, Agrawal M, Jiang X, Sontag D. Tabllm: Few-shot classification of tabular data with large language models. In: International Conference on Artificial Intelligence and Statistics. 2023;5549–81 PMLR.
25. Zhang X, Chowdhury RR, Gupta RK, Shang J. Large language models for time series: a survey. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, 2024;8335–8343. International Joint Conferences on Artificial Intelligence Organization, Jeju, South Korea <https://doi.org/10.24963/ijcai.2024/921>
26. Latif S, Shoukat M, Shamshad F, Usama M, Ren Y, Cuayáhuil H, et al. Sparks of large audio models: a survey and outlook. 2023. [arxiv: 2308.12792](https://arxiv.org/abs/2308.12792).
27. Murzaku J, Rambow O. Omnivox: Zero-shot emotion recognition with omni-llms. 2025. ArXiv [arXiv:2503.21480](https://arxiv.org/abs/2503.21480).
28. Kojima T, Gu SS, Reid M, Matsuo Y, Iwasawa Y. Large language models are zero-shot reasoners. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22. Red Hook, NY, USA: Curran Associates Inc; 2022.
29. Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, et al. Chain-of-thought prompting elicits reasoning in large language models. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22. Red Hook, NY, USA: Curran Associates Inc; 2022.
30. Sahoo P, Singh AK, Saha S, Jain V, Mondal SS, Chadha A. A systematic survey of prompt engineering in large language models: Techniques and applications. 2024. ArXiv [arXiv:abs/2402.07927](https://arxiv.org/abs/2402.07927).
31. Chung HW, Hou L, Longpre S, Zoph B, Tay Y, Fedus W, et al. Scaling instruction-finetuned language models. 2022. ArXiv [arXiv:abs/2210.11416](https://arxiv.org/abs/2210.11416).
32. Wang X, Wei J, Schuurmans D, Le Q, Chi E, Narang S, et al. Self-consistency improves chain of thought reasoning in language models. 2023. [arxiv: 2203.11171](https://arxiv.org/abs/2203.11171).
33. Chowdhury JR, Caragea C. Zero-shot verification-guided chain of thoughts. 2025.
34. Bhatia K, Jain H, Kar P, Varma M, Jain P. In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R, editors. Advances in neural information processing systems, sparse local embeddings for extreme multi-label classification, vol. 28. Red Hook, NY, USA: Curran Associates Inc; 2015.
35. Mittal A, Dahiya K, Malani S, Ramaswamy J, Kuruvilla S, Ajmera J, Chang K-H, Agarwal S, Kar P, Varma M. Multi-modal extreme classification. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022;12383–12392. IEEE, New Orleans, LA, USA. <https://doi.org/10.1109/CVPR52688.2022.01207>.
36. Zhu Y, Zamani H. ICXML: an in-context learning framework for zero-shot extreme multi-label classification. 2024. [arxiv: 2311.09649](https://arxiv.org/abs/2311.09649).
37. Chang W-C, Yu H-F, Zhong K, Yang Y, Dhillon I. Taming pretrained transformers for extreme multi-label text classification. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020;3163–3171
38. Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. *Inf Process Manag.* 1988;24(5):513–23.
39. Robertson S, Zaragoza H. The probabilistic relevance framework: Bm25 and beyond. *Found Trends Inf Retr.* 2009;3(4):333–89.
40. Reimers N, Gurevych I. Sentence-bert: Sentence embeddings using siamese bert-networks. 2019. arXiv preprint. [arXiv:1908.10084](https://arxiv.org/abs/1908.10084).
41. You R, Zhang Z, Wang Z, Dai S, Mamitsuka H, Zhu S. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Adv Neural Inf Process Syst.* 2019;32.
42. Jiang T, Liu D, Yang L. Lightxml: transformer with dynamic negative sampling for high-performance extreme multi-label text classification. *Proc AAAI Conf Artif Intell.* 2021;35(14):13252–60.
43. Zhang J, Ullah N, Babbar R. Zero-shot learning over large output spaces: Utilizing indirect knowledge extraction from large language models. 2024. ArXiv [arXiv:2406.09288](https://arxiv.org/abs/2406.09288).
44. Fidler J, Goldberg Y. Controlling linguistic style aspects in neural language generation. In: Brooke J, Solorio T, Koppel M, editors. Proceedings of the Workshop on Stylistic Variation. Copenhagen, Denmark: Association for Computational Linguistics; 2017;94–104. <https://doi.org/10.18653/v1/W17-4912>.
45. Holtzman A, Buys J, Du L, Forbes M, Choi Y. The curious case of neural text degeneration. In: International Conference on Learning Representations 2020. <https://openreview.net/forum?id=rygGQyFvH>.
46. Fan A, Lewis M, Dauphin Y. Hierarchical neural story generation. 2018. [arxiv: 1805.04833](https://arxiv.org/abs/1805.04833).
47. Nguyen M, Baker A, Neo C, Roush A, Kirsch A, Shwartz-Ziv R. Turning up the heat: min-p Sampling for Creative and coherent llm outputs. 2024. [arxiv: 2407.01082](https://arxiv.org/abs/2407.01082).
48. Sen J, Sengupta S, Dasgupta S. Advancing decoding strategies: enhancements in locally typical sampling for LLMs. *arXiv (2025)*. [arxiv:2506.05387](https://arxiv.org/abs/2506.05387).
49. Muennighoff N, Tazi N, Magne L, Reimers N. MTEB: Massive text embedding benchmark. 2023. [arxiv:2210.07316](https://arxiv.org/abs/2210.07316).
50. Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, et al. Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web.* 2015;6(2):167–95.
51. Zhang X, Zhao J, LeCun Y. Character-level convolutional networks for text classification. In: Advances in Neural Information Processing Systems, 2015;28.
52. Llama Team: The Llama 3 Herd of Models (2024). [arxiv: 2407.21783](https://arxiv.org/abs/2407.21783).
53. Team G. Gemma 2: Improving open language models at a practical size. 2024. [arxiv: 2408.00118](https://arxiv.org/abs/2408.00118).
54. Qwen Team: Qwen2.5: a party of foundation models 2024. <https://qwenlm.github.io/blog/qwen2.5/>.
55. Anthropic: Claude-3.5-Sonnet Technical Report, 2024. https://www-cdn.anthropic.com/fed9cc193a14b84131812372d8d5857f8f304c52/Model_Card_Claude_3_Addendum.pdf.
56. Xiao S, Liu Z. BGE: BAAI general embedding. GitHub repository 2023. <https://github.com/FlagOpen/FlagEmbedding>.
57. Reimers N, Gurevych I. Sentence-bert: Sentence embeddings using siamese bert-networks. 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). Hong Kong, China: Association for Computational Linguistics; 2019;3982–92.

58. Shakir A, Aarsen T, Lee S. Binary and scalar embedding quantization for significantly faster & cheaper retrieval. Hugging Face Blog 2024. <https://huggingface.co/blog/embedding-quantization>.
59. Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A. Catboost: unbiased boosting with categorical features. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, editors. *Advances in Neural Information Processing Systems*, vol. 31. Red Hook, NY, USA: Curran Associates Inc; 2018.
60. Borisov V, Leemann T, Seßler K, Haug J, Pawelczyk M, Kasneci G. Deep neural networks and tabular data: a survey. *IEEE Trans Neural Netw Learn Syst*. 2021;35:7499–519.
61. Karpukhin V, Oguz B, Min S, Lewis P, Wu L, Edunov S, et al. Dense passage retrieval for open-domain question answering. In: Webber B, Cohn T, He Y, Liu Y, editors. *In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics; 2020;6769–81. <https://doi.org/10.18653/v1/2020.emnlp-main.550>.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.