XMC-AGENT : Dynamic Navigation over Scalable Hierarchical Index for Incremental Extreme Multi-label Classification

Yanjiang Liu[1,2], Tianyun Zhong[1,2], Yaojie Lu[1]*, Hongyu Lin[1], Ben He[1,2], Shuheng Zhou[3], Huijia Zhu[3], Weiqiang Wang[3], Zhongyi Liu[3], Xianpei Han[1], Le Sun[1]

[1]Chinese Information Processing Laboratory Institute of Software, Chinese Academy of Sciences, Beijing, China

[2]University of Chinese Academy of Sciences, Beijing, China

[3]Ant Group

{zhongtianyun2023,luyaojie,hongyu,xianpei,sunle}@iscas.ac.cn
{shuheng.zsh,weiqiang.wwq,zhongyi.lzy}@antgroup.com
{liuyanjiang2021,benhe}@ucas.edu.cn, huijia.zhj@antfin.com

Abstract

The eXtreme Multi-label Classification (XMC) aims to accurately assign a large number of labels to instances, presenting challenges in learning, managing, and predicting across a vast and rapidly growing set of labels. Traditional XMC methods, such as one-vs-all and tree-based methods struggle with the increasing set of labels due to their static label assumptions, while embedding-based methods face difficulties with complex mapping relationships due to their late-interaction paradigm. In this paper, we propose a large language model (LLM) powered agent framework for extreme multi-label classification – XMC-AGENT, which can effectively learn, manage and predict an extremely large and dynamically increasing set of labels. Specifically, XMC-AGENT models the extreme multi-label classification task as a dynamic navigation problem, employing a scalable hierarchical label index to effectively manage the unified label space. Additionally, we design two algorithms to enhance the dynamic navigation capabilities of XMC-AGENT: a self-construction algorithm for building the scalable hierarchical index, and an iterative feedback learning algorithm for adjusting the agent to specific tasks. Experiments demonstrate that XMC-AGENT achieves the state-of-the-art performance on three datasets.

1 Introduction

The eXtreme Multi-label Classification (XMC) task aims to classify instances to relevant labels from an extremely large label candidate space (Bhatia et al., 2015; Bengio et al., 2019; Prabhu et al., 2018). XMC is a widely used technique in many real-world applications, such as assigning appropriate tags to products in e-commerce platforms (Medini et al., 2019; Chang et al., 2021), recommending of interest in recommendation systems (McAuley and Leskovec, 2013), and facilitating search queries auto-completion in search engines (Agrawal et al., 2013; Yadav et al., 2021).

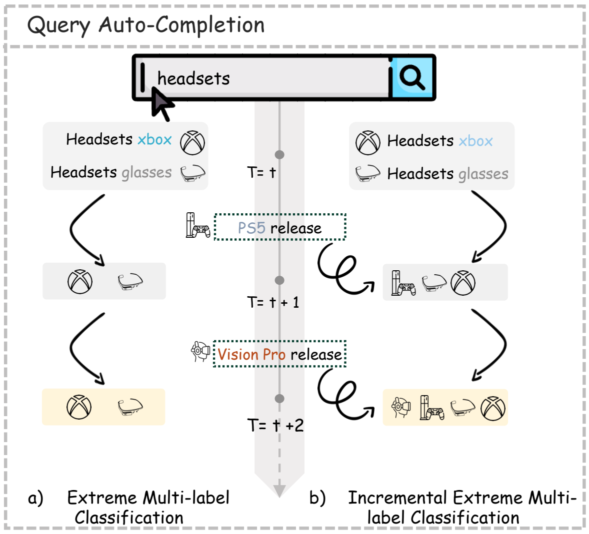

Figure 1: An example of search engine auto-completion is provided, illustrating the two distinct settings of XMC, which differ in whether the label set is fixed. When a user types headsets, standard XMC usually gives predictions from a fixed label set; whereas incremental XMC can dynamically adapt to newly added labels.

Unfortunately, due to the extensive and dynamic growth of the label candidates, XMC is a very challenging task. In real-world XMC problems, the number of potential labels often ranges from tens of thousands to millions (Song et al., 2020). Such a large output space poses significant challenges for modeling, learning, and computing the mapping from instances to large-scale labels, i.e., the scalability problem. For instance, it is difficult to

*Corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2024, pages 5659–5672
August 11-16, 2024 ©2024 Association for Computational Linguistics

directly learn the mapping from `headsets` (instance) in Figure 1 to `xbox` and `glasses` (labels), and computing all instance-label pairs will result in a high computation cost. Furthermore, the label set in real-world XMC scenarios is often dynamically changing and rapidly growing. The evolving labels further raise the challenge of efficient integration of new labels without the necessity for extensive retraining.

Current eXtreme Multi-Label Classification methods are mainly tree-based (Khandagale et al., 2019; Majzoubi and Choromanska, 2020; Zhang et al., 2021; Yu et al., 2022; Kharbanda et al., 2022) and embedding-based approaches (Gupta et al., 2021; Dahiya et al., 2021; Mittal et al., 2021a; Xu et al., 2023; Gupta et al., 2023; Chien et al., 2023). Tree-based approaches organize the labels as a fixed and static label tree, classify instances from root to leaf nodes and gradually narrow down the label candidates. These approaches, while addressing the challenge posed by large-scale label sets, struggle with dynamically growing label sets due to the utilization of prefixed, static label indices. Embedding-based approaches, on the other hand, predict labels by mapping labels and instances into the same vector space and selecting labels based on their vector similarities. However, due to the lack of fine-grained interaction between instances and labels, issues arise when dealing with complex mapping relationships. Moreover, to effectively integrate new labels, a process of re-training or continual training is necessary. However, the extensive label space and large volumes of data make retraining resource-intensive, and continuous learning can result in severe catastrophic forgetting, degrading previously acquired label knowledge.

In this paper, we propose an agent-based framework for extreme multi-label classification – XMC-AGENT, which can effectively learn, manage, and predict the extremely large and dynamically increasing set of labels by leveraging LLMs-powered agents. Specifically, XMC-AGENT models the extreme multi-label classification task as a dynamic navigation problem (i.e., the model searches through the label space to locate the labels corresponding to the instance), and employs a scalable hierarchical label index to effectively manage the extensive label space via transforming them into a tree-like label index. In this way XMC-AGENT can uniformly manage both existing labels and future labels and seamlessly integrate future labels by inserting them at suitable positions in the tree as they emerge, leveraging their connections and associations with existing labels, thereby avoiding disruption of existing structures and the need for extensive retraining. By leveraging the capabilities of LLMs for dynamic navigation within a structured label space, XMC-AGENT offers a novel and effective solution for addressing the scalability and adaptability challenges of XMC.

Given the XMC-AGENT framework, we propose a *self-construction* algorithm for scalable hierarchical label building and a *self-correction* algorithm for the general navigational capabilities of LLMs. Specifically, the *self-construction* algorithm autonomously transforms the large label set into a structured hierarchical index by adopting a self-questioning strategy, i.e., the XMC-AGENT determines comparison relations between labels and recursively merges these relations to build the structured label index. In this way, the self-construction algorithm enables the seamless integration of newly emerged labels. Furthermore, we propose a *self-correction* algorithm, which dynamically obtains feedback signals from previous incorrect navigation trajectories and iteratively adjusts its navigation capability on specific tasks.

Generally, our main contributions are:

- We propose an LLM-powered agent framework named XMC-AGENT. By modeling the XMC problem as a navigation task within the label space, XMC-AGENT can naturally handle the incremental XMC problem and achieve state-of-the-art performance on three standard datasets.
- We design a scalable hierarchical label index construction algorithm named *self-construction*. By discovering the associative relationships between labels, *self-construction* enables the seamless integration of newly emerged labels into an existing label index.
- We design an iterative feedback learning algorithm named *self-correction*, which leverages the navigation trajectory as feedback to effectively align general navigation capability with specific classification scenarios.

2 Methodology

Let $\mathcal{X}$ and $\mathcal{Y}$ represent the sets of input instances and labels respectively, and $\{\mathcal{Y}_0, \mathcal{Y}_1, \cdots, \mathcal{Y}_k\}$ represent the acquired labels at different time. For

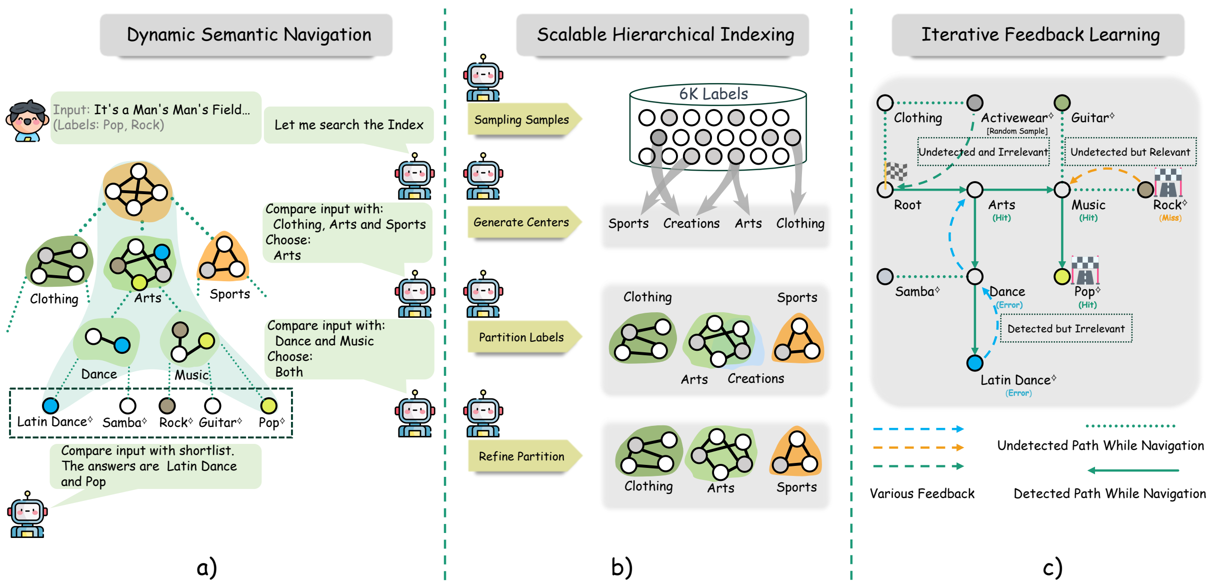

Figure 2: Illustrations of our proposed LLM-powered agent framework. a) Modeling the extreme multi-label classification task as a dynamic navigation problem, and utilizing a two-stage navigation strategy to seek optimal results over a semantic hierarchical label index[1]. b) Employing a *self-construction* algorithm to build a scalable hierarchical label index by adopting a self-questioning strategy. c) Employing a *self-correction* algorithm to enhance the general navigational capabilities by iteratively learning feedback signals from previous navigation trajectories.

simplicity, we consider a two-stage incremental setting in this paper, which means $\mathcal{Y} = \mathcal{Y}_0 \cup \mathcal{Y}_1$.

We bring XMC-AGENT to confront the challenges encountered in addressing the incremental XMC, which is achieved by: (1) Constructing a scalable hierarchical label index using LLMs. (2) Employing iterative feedback learning to effectively adjust LLMs with specific tasks.

2.1 Extreme Multi-label Classification as Dynamic Navigation

The essence of multi-label classification lies in searching for multiple relevant labels from the label space, which leads to increased difficulty in directly solving the problem (i.e., one-vs-all approaches) with the increase of the set of labels. Considering this, we propose XMC-AGENT to simplify the problem by incorporating the interrelationships between labels to construct a label index $\mathcal{I}$, which consists of a specialized center c along with multiple sub-indices, denoted as $\mathcal{I} \equiv (c, \{\mathcal{I}^i\})$, and employing an LLM-powered agent to navigate over the index for the optimal results. The main idea of dynamic navigation is illustrated in Figure 2.a.

Specifically, we employ a two-stage navigation strategy to seek the optimal results over the hierarchical index. In the first stage, a breadth-first search is employed to generate a shortlist via the comparison of the instances and centers in the index. The breadth-first search stops when traversing the entire index or reaching a certain number of terminal index (i.e., reaching `Dance` and `Music` in Figure 2. b). The shortlist is composed of the union of all labels from the reached terminal index (i.e., [`Latin Dance`, `Samba`, `Rock`, `Guitar`, `Pop`]). In the second stage, XMC-AGENT selects labels relevant to the instance from the shortlist and outputs them based on the relevance (i.e., XMC-AGENT assign `Latin Dance` and `Pop` to the instance, and regard the former as more relevant).

2.2 Scalable Hierarchical Index Building via Self-construction

To adapt the navigation strategy (comparison among the instance and centers), we adopt a compare-based (Schultz and Joachims, 2003; Haghiri et al., 2017; Emamjomeh-Zadeh and Kempe, 2018; Ghoshdastidar et al., 2019) index building approach; instead of using explicit similarity computations to form a hierarchical label index. Specifically, we utilize LLMs to determine comparison relations between labels and recursively merge these relations to build the structured label index.

[1]Tags with superscript ◇ represent the actual labels, while the others represent centers generated during the construction.

Algorithm 1 Hierarchical Label Indexing of *self-construction*

Input: Label partition $\mathbf{p} = (c, \mathcal{Y})$, Task description $\mathcal{T}$
Output: Hierarchical label index $\mathcal{I}$

if should stop **then** ▷ Pre-defined stop criteria
  return p
end if
repeat
  $\hat{\mathcal{Y}} \leftarrow Sample(\mathcal{Y})$ ▷ Sample a subset labels to represent $\mathcal{Y}$
  $\mathcal{C} \leftarrow GenCenters(T, \hat{\mathcal{Y}})$ ▷ Generate sub-index centers according to $\hat{\mathcal{Y}}$
  for $l_i \in \mathcal{Y}$ **do** ▷ Assign each label to relevant centers
    $\mathcal{C}^i \leftarrow AssignCenter(l_i, \mathcal{C})$
  end for
  $\mathcal{P} \leftarrow Partition(\{(l_i, \mathcal{C}^i)\}_{i=1}^{|\mathcal{Y}_0|})$ ▷ Create sub-partitions according to the assignment
  $\mathcal{P}^{\dagger} \leftarrow Validation(\mathcal{P})$
until $\mathcal{P}^{\dagger} \neq \varnothing$
for $p^i \in \mathcal{P}^{\dagger}$ **do** ▷ Recursive execution
  $\mathcal{I}^i \leftarrow QuickCluster(p^i, \mathcal{T})$ ▷ Algorithm 1
end for
$\mathcal{I} \leftarrow Merge(c, \{\mathcal{I}^i\}_{i=1}^{|\mathcal{P}^{\dagger}|})$ ▷ Algorithm 2
return $\mathcal{I}$

2.2.1 Compare-based Hierarchical Indexing

Considering the label set $\mathcal{Y}_0$ in Figure 2.b, we initially regard it as a partition $p^{\star} = (root, \mathcal{Y}_0)$ and sample a subset $\hat{\mathcal{Y}}$ as representations from $p^{\star}$. Then, a collection of sub-index centers (e.g., `Sports`, `Creations`, `Clothing` and `Arts`) can be generated based on $\hat{\mathcal{Y}}$, using the following prompt :

> Which centers are relevant to the provided product category?

To get the partition of $\mathcal{Y}_0$, each label $l_i \in \mathcal{Y}_0$ is compared with $\mathcal{C}$, assigning l_i to relevant centers $\mathcal{C}^i$, using the following prompt :

> Look through the provided labels of product categories and give a set of cluster centers.

This process generate $k+1$ partitions eventually, denoted as $\mathcal{P} = \{p_1, \cdots, p_k, p_{other}\}$. The first k partitions correspond to the k centers and their assigned labels, while the last partition p_{other}, encompasses labels irrelevant to all centers in $\mathcal{C}$.

We additionally apply a post-refinement to address potential issues existing in the obtained partition (i.e., there is a significant overlap between partition `Arts` and `Creations` in Figure 2.a, retaining both would result in the waste of resources), as $\mathcal{C}$ is generated from a subset of $\mathcal{Y}_0$.

Algorithm 2 Merge Operation of *self-construction*

Input: Sub-index set $\{\mathcal{I}^i\}$, Predecessor center c
Output: Hierarchical label index $\mathcal{I}$

Init: $S_c \leftarrow [\,]$ ▷ Successors of c
for $\mathcal{I}^i \in \{\mathcal{I}^i\}$ **do**
  if $\mathcal{I}^i$ is $other$ **then** ▷ The index for a group of labels assigned to center $other$
    Add successors of $\mathcal{I}^i$ to S_c
  else
    Add $\mathcal{I}^i$ to S_c
  end if
end for
return (c, S_c)

Algorithm 3 Scalable Label Integration of *self-construction*

Input: Hierarchical label index $\mathcal{I}$, New Labels $\mathcal{Y}^{'}$, Task description $\mathcal{T}$
Output: Extended Index $\mathcal{I}$

for $l_i \in \mathcal{Y}^{'}$ **do**
  $\mathcal{P}^i \leftarrow Search(\mathcal{I}, l_i)$ ▷ Compare l_i with centers in $\mathcal{I}$ in a top-down manner
  for $p_j^i \in \mathcal{P}^i$ **do**
    $p_j^i \leftarrow (c_j^i, \mathcal{Y}_j^i \cup \{l_i\})$ ▷ Insert l_i to partition p_j^i
    if p_j^i should split **then** ▷ Pre-defined criteria
      $\mathcal{I}_j^i \leftarrow QuickCluster(p_j^i, \mathcal{T})$ ▷ Algorithm1
      $p_j^i \leftarrow \mathcal{I}_j^i$ ▷ Replace p_j^i with new index
    end if
  end for
end for
return $\mathcal{I}$

We recursively execute the above process for each partition until the stopping criteria are satisfied (i.e., the number of labels within the partition is less than a pre-defined threshold). One noteworthy benefit of using the recursive strategy is that as the recursion depth increases, the label similarities within an obtained partition also increase. This in turn leads to the increase in the specificity of the sub-index center's representation (i.e., `Clothing` -> `Athletic Apparel` -> `Running Apparel`).

As mentioned before, the partition process also generates non-semantic centers, denoted as p_{other}, which block the information circulation over the index. To address this issue, we establish direct connections between the successors and predecessors of these centers, thereby eliminating their impact on the semantic index. The details of the index-building process are shown in Algorithm 1.

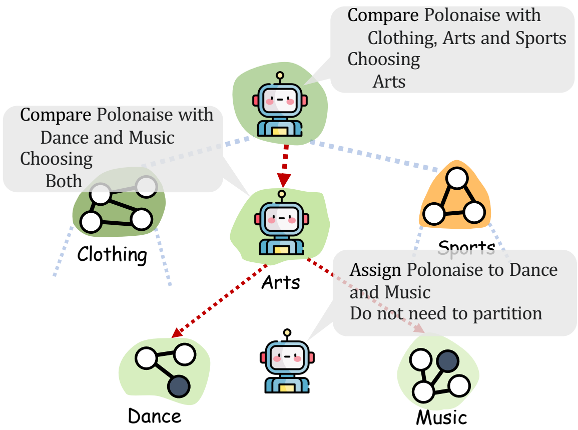

Figure 3: An example of adding a new label *Polonaise*[2] to an existing label index. After a few level-wise comparisons, the new label is inserted into two terminal partitions. Since neither of the partitions requires further subdivision, the insertion is complete.

2.2.2 Integration of Scalable Indexing

To incorporate new labels into an existing index, we propose an *InsertSort* like algorithm. We use an example to illustrate the main idea in Figure 3. For each new label, XMC-AGENT recursively compares it with the centers of the sub-index and assigns it to relevant sub-indices until reaching the terminal index. Once the labels within a terminal index surpassing the pre-defined threshold, we use Algorithm 1 to directly generate fine-grained sub-indices for the terminal index.

2.3 Agent Adaption via Iterative Feedback Learning

To adjust the mapping relationship between instances and labels for a specific application, one approach is to add summarized mapping rules to the context of LLMs. However, due to the inherent challenge of having extensive labels, the summarized rules are incapable of covering all annotated data, which gives rise to inconsistency between classification results and user intent.

Different from using summarized decision criteria, we propose an approach to utilize feedback to inform the navigation process of LLMs. Giving an input instance, LLMs would give several predictions using the self-constructed index, which consists of two distinct label types: **Hit** representing labels both detected and relevant, like `Pop` in Figure 2, and **Error** representing labels detected but irrelevant, indicating inconsistency, like `Latin Dance` in Figure 2. Additionally, there exist labels which are relevant but remain undetected in the search process, denoted as **Miss**, also indicating inconsistency, like `Rock` in Figure 2. Furthermore, based on the three types of labels, we also mark the centers along their search paths with the corresponding type. For example, `Arts` is on the search path of `Pop`, and `Dance` is on the search path of `Latin Dance`, thus they are marked as **Hit** and **Error** respectively.

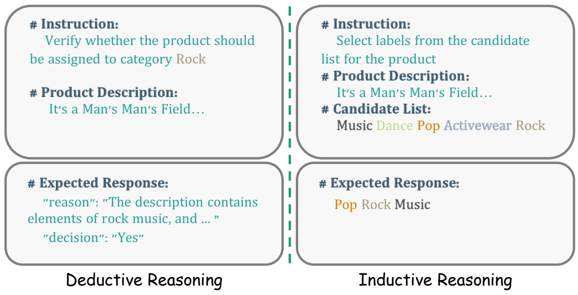

Figure 4: An example of the collected feedback data. Deductive Reasoning is the self-feedback explaining why `Rock` (undetected but relevant) is a relevant label, and Inductive Reasoning is the contrastive feedback used to distinguish relevant labels from a carefully crafted shortlist.

Self-Feedback by Deductive Reasoning To provide feedback using deductive reasoning, we utilize the decision criteria provided by the LLMs themselves for both the two types of inconsistent labels (**Error** and **Miss**). For example, in Figure 2, XMC-AGENT leverages the self-generated decision criteria for the inconsistent label `Rock` (**Miss**) as a feedback signal to adjust its navigational capability.

Contrastive-Feedback by Inductive Reasoning To provide feedback using inductive reasoning, we create a shortlist by randomly sampling the three types of labels along with irrelevant labels without detection, akin to the navigation process, and the expected response are all relevant labels in the list.

When a sufficient amount of feedback, i.e., Figure 4, is collected, we engage in the refinement of LLMs iteratively to align the navigation capability using the feedback data.

3 Experimental Setting

3.1 Datasets and Evaluation

We evaluate our method on the following datasets: AmazonCat-13K (McAuley and Leskovec, 2013)

[2]Polonaise is a dance of Polish origin. Polonaise dance greatly influenced European ballrooms, folk music, and European classical music.

Dataset	Instances		Labels		
	N$_{train}$	N$_{test}$	$\lvert\mathcal{Y}_0\rvert$	$\lvert\mathcal{Y}_1\rvert$	*Avg.*
AmazonCat-13K†	1.1M	307K	6658	6672	2.6/5.1
LF-Amazon-131K†	295K	135K	51378	77067	1.62/2.11
LF-WikiSeeAlso-320K†	693K	118K	124924	187387	2.26/3.05

Table 1: Dataset statistics information. $|\mathcal{Y}_0|$ indicates the label size in the first stage, and $|\mathcal{Y}_1|$ indicates the number of newly added labels in the second stage. *Avg.* represents the average number of labels per instance across the two stages.

in product tagging domain, LF-Amazon-131K (McAuley and Leskovec, 2013) in the recommendation domain and LF-WikiSeeAlso-320K in the wiki-page tagging domain, where 13K, 131K and 320K indicate the total label size. All datasets are available in the extreme classification repository (Bhatia et al., 2016). To evaluate the ability of various methods in an incremental setting, we randomly split the labels into two parts. The statistics of the processed datasets (notated with superscript) are listed in Table 1.

We consider two evaluation setups: Incremental Performance (Inc) and Overall Performance (Overall). The former focus on classification results only on $\mathcal{Y}_1$ and the latter focus on both $\mathcal{Y}_0$ and $\mathcal{Y}_1$. We evaluate the models' performance with Precision@k and Recall@k, where $k \in \{1, 3, 5, 10\}$, which are two commonly-used evaluation metrics in XMC (Xiong et al., 2022; Aggarwal et al., 2023).

3.2 Baselines

We compare our method with the following baselines. **1) BM25** conducts a nearest neighbor retrieval using TF-IDF features. **2) TAS-B** (Hofstätter et al., 2021) ranks labels based on the similarity with the instance by Faiss (Johnson et al., 2019). **3) MACLR** (Xiong et al., 2022) leverages the raw text and self-training with pseudo positive pairs to improve the extreme zero-shot capacity. **4) SemSup-XC** (Aggarwal et al., 2023) use web-collected semantic descriptions to represent labels and facilitate generalization by using a combination of semantic and lexical similarity. **5) ICXML** (Zhu and Zamani, 2023) propose three demonstration selection approaches to create in-context learning prompts for `gpt-3.5-turbo` to generate approximate labels, then using TAS-B mapping these approximate labels to labels set and get final reranking results by `gpt-3.5-turbo`. **6) Linear Search** To assess the efficacy of directly employing LLMs for XMC, we traverse all labels using both zero-shot and few-shot approaches, sorting the labels based on the output logits. Considering the scale of the label sets, we only conducted experiments on AmazonCat-13K†.

4 Results and Analysis

4.1 Main results

In all experiments, we choose Vicuna-13B-v1.5 (Zheng et al., 2023) as the base LLM. The experimental results over three datasets, as presented in Table 2, reveal that:

1) XMC-Agent exhibits a noteworthy improvement in addressing incremental XMC problem. Compared with previous methods, our classification as a navigation approach demonstrates an improved capability in handling new labels on three datasets of different scales. Simultaneously, our approach achieves optimal performance under the overall setup, exemplifying a commendable balance between utility and generalization.

2) XMC-Agent enhances its dynamic navigation capability by integrating the proposed components. Compared with the Linear Search results on AmazonCat-13K†, our approach achieves an acceptable time cost while exhibiting superior navigation performance under both setups (i.e., 9.3% P@1 improvement in Inc and 45.9% P@1 improvement in Overall), which indicates the effectiveness of the proposed components.

3) XMC-Agent demonstrates a stable performance across various application scenarios. In our experiments, we found that previous methods have varying applicability across scenarios. For instance, TAS-B exhibits a better performance in scenarios with longer label length (e.g., LF-Amazon-131K† and LF-WikiSeeAlso-320K†), ICXML performs better in cases where the mapping relationship between instances and labels is complex (e.g., LF-WikiSeeAlso†), and SemSup-XC demonstrates better capabilities in scenarios where the mapping relationship is more direct (e.g., AmazonCat-13K† and LF-Amazon-131K†). Our approach, which utilizes an LLM to uniformly manage the label space and learn mapping relationships from feedback rather than integrating them into embedding, enables effective handling of various applications.

4.2 Analysis

To understand the impact of various key components on the results, we conduct ablation studies on

Method	Inc								Overall							
	Precision				Recall				Precision				Recall			
	P@1	P@3	P@5	P@10	R@1	R@3	R@5	R@10	P@1	P@3	P@5	P@10	R@1	R@3	R@5	R@10
AmazonCat-13K†																
BM25	8.7	5.6	4.3	2.9	3.5	6.8	8.6	11.7	16.8	11.2	8.7	6.0	3.2	6.5	8.3	11.4
TAS-B (Hofstätter et al., 2021)	10.1	6.5	5.0	3.3	4.1	7.9	10.1	13.6	19.3	12.9	10.1	7.0	3.8	7.5	9.7	13.3
MACLR (Xiong et al., 2022)	7.4	5.0	4.0	2.8	2.7	5.5	7.4	10.6	15.2	10.3	8.2	5.8	2.7	5.6	7.4	10.4
SemSup-XC (Aggarwal et al., 2023)	25.6	17.2	13.3	9.0	11.0	23.6	30.7	41.3	**86.5**	62.5	47.3	29.4	19.4	37.3	45.1	54.4
ICXML (Zhu and Zamani, 2023)	14.8	10.6	8.4	5.3	5.4	12.4	15.8	20.6	32.0	20.9	16.5	10.7	6.0	11.8	15.4	19.4
Linear Search (Zero-Shot)	16.0	13.8	12.3	9.7	9.2	23.3	33.7	49.7	21.6	21.0	20.2	16.5	5.6	19.7	30.9	49.7
Linear Search (3-Shot)	17.0	15.2	12.8	9.5	9.9	23.7	35.8	50.3	34.2	28.2	24.5	18.2	12.0	27.5	38.9	55.3
XMC-AGENT (*ours*)	**36.3**	**29.2**	**24.1**	**15.3**	**24.1**	**37.5**	**43.4**	**50.6**	80.1	**64.2**	**50.3**	**33.3**	**22.8**	**39.6**	**51.0**	**62.7**
LF-Amazon-131K†																
BM25	10.2	8.8	6.8	4.3	7.2	17.8	22.3	27.6	13.8	12.2	9.5	6.1	7.1	17.4	22.0	27.3
TAS-B (Hofstätter et al., 2021)	11.5	9.6	7.4	4.7	8.1	19.3	24.2	30.0	15.9	13.4	10.5	6.7	8.2	19.2	24.1	29.9
MACLR (Xiong et al., 2022)	11.6	9.6	7.5	4.8	8.0	19.3	24.5	30.8	15.9	13.6	10.7	6.9	8.1	19.4	24.6	31.1
SemSup-XC (Aggarwal et al., 2023)	21.5	15.3	11.2	6.7	10.0	31.2	37.2	43.7	19.1	17.5	**13.8**	8.7	10.1	25.9	32.6	40.2
ICXML (Zhu and Zamani, 2023)	19.0	12.7	9.5	5.5	14.0	26.4	32.2	37.5	**24.6**	17.1	12.7	7.6	13.4	**26.3**	31.7	37.3
XMC-AGENT (*ours*)	**24.8**	**18.3**	**13.1**	**8.1**	**21.4**	**32.0**	**39.3**	**45.5**	22.7	**18.9**	13.7	**10.2**	**26.1**	25.7	**34.3**	**46.5**
LF-WikiSeeAlso-320K†																
BM25	10.4	7.8	6.1	4.0	7.1	14.6	18.0	22.6	13.8	10.9	8.6	5.8	7.1	14.5	17.9	22.5
TAS-B (Hofstätter et al., 2021)	13.2	10.1	7.9	5.2	9.3	**19.4**	23.9	29.9	17.4	14.0	11.1	7.4	9.3	19.3	23.8	29.8
MACLR (Xiong et al., 2022)	7.5	7.2	5.9	4.1	5.1	12.7	16.5	21.6	10.6	10.7	8.8	6.1	5.4	13.5	17.3	22.5
SemSup-XC (Aggarwal et al., 2023)	13.4	13.5	12.1	9.2	5.5	14.4	20.1	28.3	10.6	14.1	13.4	11.3	3.1	10.1	14.9	23.0
ICXML (Zhu and Zamani, 2023)	15.0	10.9	9.0	6.6	5.3	10.4	13.1	18.5	21.6	17.2	14.3	10.5	4.9	10.6	13.5	19.2
XMC-AGENT (*ours*)	**15.8**	**14.3**	**12.6**	**9.9**	**10.3**	16.0	**25.3**	**32.5**	**24.3**	**18.4**	**15.6**	**13.0**	**12.4**	**19.9**	**26.3**	**33.0**

Table 2: Main results of XMC-AGENT on three datasets, where **Inc** measures the performance on $\mathcal{Y}_1$ and **Overall** measures the performance on both $\mathcal{Y}_0$ and $\mathcal{Y}_1$. The best and second-best performing score in each column are highlighted with bold and underline, respectively. Considering the scale of the label sets, we only experiment with Linear Search on AmazonCat-13K†.

the key components of XMC-AGENT and further provide qualitative analysis of the performance of previous methods with continual fine-tuning.

4.2.1 Ablating the Label Index

To investigate the impact of label index on the final performance, we replaced the index used in XMC-AGENT with two alternative methods. The first one uses K-Means to recursively partition the label set (with k=16) as mentioned in PECOS (Yu et al., 2022). The second one employs Faiss (Johnson et al., 2019) as a retriever, to identify the Top 500 similar labels with the instances as a shortlist. Both the two approaches use TAS-B as the text embedder. From results presented in Table 3, we can observe that :

1) Replacing with K-Means results in significant performance degradation. This is partly due to the cascading error propagation in the index, as each label only appears once in the K-Means index. Additionally, to navigate over the index, each cluster requires a description as representation. However, due to the limitations of LLMs' context window and long-text processing capabilities, the generated descriptions cannot fully cover labels within the cluster, resulting in the inability to find relevant labels based on the center during navigation.

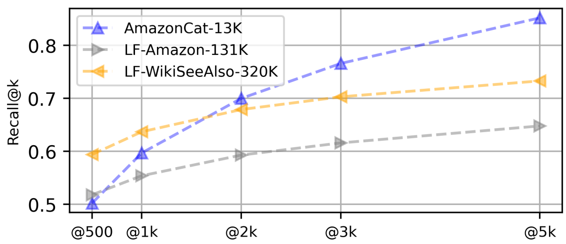

Figure 5: Recall@k performance using TAS-B as the text embedder and Faiss as the retriever on three datasets.

2) Replacing with a shortlist is more effective than K-Means, but still inferior to our approach. This is due to the retrieval method can only detect a fixed portion of relevant labels (as shown in Figure 5, even at R@3000, only 60%-70% of the relevant labels can be detected), thereby restricting the exploration space for subsequent feedback learning.

4.2.2 Ablating Feedback Learning

To investigate the influence of the feedback mechanisms, we separately employ one at a time. From the results presented in Table 3, we can observe that both mechanisms contribute to the final per-

Method	Components			AmazonCat-13K†				LF-Amazon-131K†			
	LLM Index	Inductive Reasoning	Deductive Reasoning	Inc		Overall		Inc		Overall	
				P@1	R@10	P@1	R@10	P@1	R@10	P@1	R@10
Ablating Label Index											
XMC-Agent	✓	✓	✓	**36.3**	**50.6**	**80.1**	**62.7**	**24.8**	**45.5**	**22.7**	**46.5**
Replace LLM Index with K-Means Index	✗	✓	✓	17.3	24.4	15.6	25.3	19.9	34.6	17.1	25.2
Replace LLM Index with Faiss Top 500	✗	✓	✓	22.4	34.0	56.0	53.3	20.2	34.1	20.0	36.9
Ablating Feedback Learning											
XMC-Agent	✓	✓	✓	**36.3** $_{13.0\uparrow}$	**50.6** $_{8.4\uparrow}$	**80.1** $_{35.8\uparrow}$	**62.7** $_{20.2\uparrow}$	**24.8** $_{7.2\uparrow}$	**45.5** $_{5.9\uparrow}$	**22.7** $_{5.4\uparrow}$	**46.5** $_{5.7\uparrow}$
Adopt Inductive Reasoning	✓	✓	✗	$26.6_{3.3\uparrow}$	$49.3_{7.1\uparrow}$	$57.5_{13.2\uparrow}$	$58.1_{15.6\uparrow}$	$21.6_{4.0\uparrow}$	$42.8_{3.2\uparrow}$	$19.5_{2.2\uparrow}$	$44.4_{3.6\uparrow}$
Adopt Deductive Reasoning	✓	✗	✓	$31.5_{8.2\uparrow}$	$47.5_{5.3\uparrow}$	$60.4_{16.1\uparrow}$	$56.7_{14.2\uparrow}$	$22.4_{4.8\uparrow}$	$42.1_{2.5\uparrow}$	$19.0_{1.7\uparrow}$	$43.4_{2.6\uparrow}$
Adopt None (base performance)	✓	✗	✗	23.3	42.2	44.3	42.5	17.6	39.6	17.3	40.8

Table 3: Component-wise ablation of XMC-Agent. Ablating Label Index refers to replacing the self-construct label index with a K-Means index and a shortlist composed of the Top 500 labels retrieved by Faiss to investigate the impact of label index on the final performance. Ablating Feedback Learning represents separately employing one feedback mechanism during iterative feedback learning to investigate the influence of the feedback mechanism.

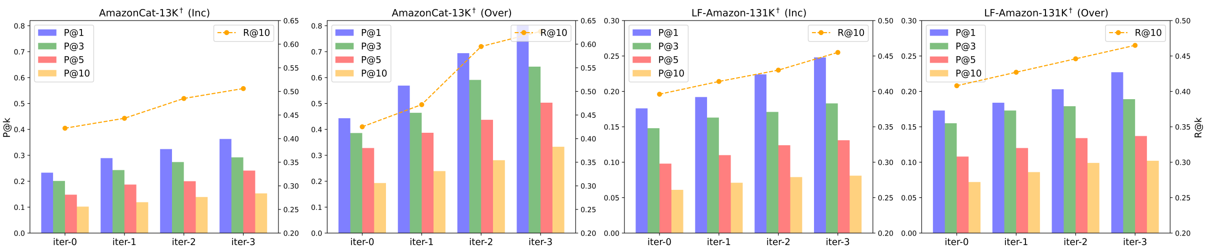

Figure 6: Precision $@\{1, 3, 5, 10\}$ and Recall@10 results at different iterations. Iter-0 stands for the model without feedback learning. The various metrics of XMC-Agent have all shown improvement during the iterative process, and there is also an enhancement in the metrics on $\mathcal{Y}_1$ (Inc), indicating our method exhibits good generalization performance and does not merely learn the corresponding relationships within the training set.

formance, but the emphasis on the improvements differs between the two mechanisms. Employing feedback based on inductive reasoning solely leads to a greater improvement in recall. while solely employing feedback based on deductive reasoning leads to a greater improvement in precision.

This discrepancy arises from the inherent nature of the feedback signals in the two mechanisms. When using deductive reasoning, the feedback signal originates from the self-correction of the inconsistent label, thereby enhancing the discriminatory ability for one specific label. While using inductive reasoning, the signal comes from the exploration of random candidates, leading to an improvement in the discriminatory ability for overall labels.

Additionally, we assess the impact of iteratively employing the feedback mechanism, as illustrated in Figure 6. Across three rounds of iteration, both metrics on the two datasets exhibit an improvement, suggesting the proposed feedback learning mechanism possesses robust stability and generalization.

4.2.3 Effect of Continual Fine-tuning

As the baselines are not designed for incremental XMC problems, we conduct continual fine-tuning (*CFT*) on the model trained with $\mathcal{Y}_0$ using additional labels to assess their adaptability in dealing with new labels. The corresponding results are shown in Table 4. It can be observed that the model's classification ability for new labels significantly improved after *CFT*. However, the overall performance across the entire labels does not show improvement, suggesting the forgetting of the capabilities learned by previous methods on a fixed label set.

5 Related Works

Previous research on XMC can be divided into two settings: full label coverage (Prabhu et al., 2018; Mittal et al., 2021b,a; Kharbanda et al., 2022; Yu et al., 2022) and weak label coverage (Gupta et al., 2021; Dahiya et al., 2021; Xiong et al., 2022; Gupta et al., 2023), the difference is whether supporting predictions for newly added labels during inference.

Method	Inc		Overall	
	P@1	R@10	P@1	R@10
AmazonCat-13K†				
XMC-AGENT	36.3	50.6	80.1	62.7
MACLR (*CFT*)	15.8	12.3	14.6	9.8
SemSup-XC (*CFT*)	74.3	48.9	41.4	54.7
LF-Amazon-131K†				
XMC-AGENT	24.8	45.5	22.7	46.5
MACLR (*CFT*)	17.3	34.3	15.8	31.8
SemSup-XC (*CFT*)	23.3	47.2	19.8	42.4
LF-WikiSeeAlso-320K†				
XMC-AGENT	15.8	32.5	24.3	33.0
MACLR (*CFT*)	12.3	23.6	11.2	22.8
SemSup-XC (*CFT*)	14.6	28.3	13.5	24.7

Table 4: Results of XMC-AGENT and continue fine-tuning baselines (*CFT*). *CFT* represents previous methods in a continue fine-tuning setting that first train on $\mathcal{Y}_0$ and then continue fine-tuning on $\mathcal{Y}_1$.

A prevalent approach for addressing weak label coverage entails the utilization of a bi-encoder to map labels and instances into the same vector space. SiameseXML (Dahiya et al., 2021) generalizes existing Siamese Networks (Chen et al., 2020) by combining Siamese architectures with per-label extreme classifiers. MACLR (Xiong et al., 2022) constructs label and input text encoders by training a pseudo label-input annotation data through a two-stage process. SemSup-XC (Aggarwal et al., 2023) uses web information to augment label semantics and calculates the similarity between label and input from both semantic and lexicon perspectives.

Unlike previous approaches that transformed the classification task into an end-to-end generation task (Simig et al., 2022) or utilized the in-context learning ability of LLMs to generate approximate labels(Chang et al., 2018; Tay et al., 2022; Kishore et al., 2023; Wang et al., 2023), we model XMC as an LLM-Agent dynamic navigation task(Kishore et al., 2023; Wang et al., 2023), allowing for better handling the dynamically growing extensive labels.

6 Conclusion

In this paper, we propose XMC-AGENT to address the challenge of dynamically expanding label set in extreme multi-label classification. This framework utilizes a self-constructed label index for effective management of the extensive labels. And incorporates an iterative feedback learning mechanism to adjust general navigational capabilities to a specific task. The results on three standard datasets indicate that our approach effectively enhances the classification performance in incremental settings.

Limitations

We identify two limitations in our work that necessitates further investigation. Firstly, we only employ Vicuna-13B-v1.5 as the base model of XMC-AGENT, the impact of using different LLMs on the final performance requires further detailed research. Additionally, we only explore extreme multi-label text classification problem with XMC-AGENT, future works can extend the approach presented in this paper to other domains, like the extreme multi-label image classification problem.

7 Acknowledge

We sincerely thank all anonymous reviewers for their insightful comments and valuable suggestions. This research work is supported by the National Natural Science Foundation of China under Grants no. 62306303, no. 62122077 and no. 62106251. This work is supported by Ant Group Research Fund.

References

Pranjal Aggarwal, Ameet Deshpande, and Karthik Narasimhan. 2023. Semsup-xc: Semantic supervision for zero and few-shot extreme classification. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24.

Samy Bengio, Krzysztof Dembczynski, Thorsten Joachims, Marius Kloft, and Manik Varma. 2019. Extreme Classification (Dagstuhl Seminar 18291). *Dagstuhl Reports*, 8(7):62–80.

K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. 2016. The extreme classification repository: Multi-label datasets and code.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. *Advances in neural information processing systems*, 28.

Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, Japinder Singh, and Inderjit S. Dhillon. 2021. Extreme multi-label learning for semantic matching in product search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 2643–2651, New York, NY, USA. Association for Computing Machinery.

Wei-Cheng Chang, Hsiang-Fu Yu, Inderjit S. Dhillon, and Yiming Yang. 2018. Secseq: Semantic coding for sequence-to-sequence based extreme multi-label classification.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Eli Chien, Jiong Zhang, Cho-Jui Hsieh, Jyun-Yu Jiang, Wei-Cheng Chang, Olgica Milenkovic, and Hsiang-Fu Yu. 2023. Pina: Leveraging side information in extreme multi-label classification via predicted instance neighborhood aggregation. *arXiv preprint arXiv:2305.12349*.

Kunal Dahiya, Ananye Agarwal, Deepak Saini, Gururaj K, Jian Jiao, Amit Singh, Sumeet Agarwal, Purushottam Kar, and Manik Varma. 2021. Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2330–2340. PMLR.

Ehsan Emamjomeh-Zadeh and David Kempe. 2018. Adaptive hierarchical clustering using ordinal queries. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, page 415–429, USA. Society for Industrial and Applied Mathematics.

Debarghya Ghoshdastidar, Michaël Perrot, and Ulrike von Luxburg. 2019. Foundations of comparison-based hierarchical clustering. *Advances in neural information processing systems*, 32.

Nilesh Gupta, Sakina Bohra, Yashoteja Prabhu, Saurabh Purohit, and Manik Varma. 2021. Generalized zero-shot extreme multi-label learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 527–535, New York, NY, USA. Association for Computing Machinery.

Nilesh Gupta, Devvrit Khatri, Ankit S Rawat, Srinadh Bhojanapalli, Prateek Jain, and Inderjit S Dhillon. 2023. Efficacy of dual-encoders for extreme multi-label classification. *arXiv preprint arXiv:2310.10636*.

Siavash Haghiri, Debarghya Ghoshdastidar, and Ulrike von Luxburg. 2017. Comparison-Based Nearest Neighbor Search. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 851–859. PMLR.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 113–122, New York, NY, USA. Association for Computing Machinery.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Sujay Khandagale, Han Xiao, and Rohit Babbar. 2019. Bonsai – diverse and shallow trees for extreme multi-label classification.

Siddhant Kharbanda, Atmadeep Banerjee, Erik Schultheis, and Rohit Babbar. 2022. Cascadexml: Rethinking transformers for end-to-end multi-resolution training in extreme multi-label classification. *Advances in Neural Information Processing Systems*, 35:2074–2087.

Varsha Kishore, Chao Wan, Justin Lovelace, Yoav Artzi, and Kilian Q. Weinberger. 2023. Incdsi: Incrementally updatable document retrieval.

Maryam Majzoubi and Anna Choromanska. 2020. Ldsm: Logarithm-depth streaming multi-label decision trees. In *International Conference on Artificial Intelligence and Statistics*, pages 4247–4257. PMLR.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 165–172, New York, NY, USA. Association for Computing Machinery.

Tharun Kumar Reddy Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. 2019. Extreme classification in log memory using count-min sketch: A case study of amazon search with 50m products. In *Advances in Neural Information Processing Systems*, pages 13244–13254.

Anshul Mittal, Kunal Dahiya, Sheshansh Agrawal, Deepak Saini, Sumeet Agarwal, Purushottam Kar, and Manik Varma. 2021a. Decaf: Deep extreme classification with label features. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 49–57.

Anshul Mittal, Noveen Sachdeva, Sheshansh Agrawal, Sumeet Agarwal, Purushottam Kar, and Manik Varma. 2021b. Eclare: Extreme classification with label graph correlations. In *Proceedings of the Web Conference 2021*, pages 3721–3732.

Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, pages 993–1002.

Matthew Schultz and Thorsten Joachims. 2003. Learning a distance metric from relative comparisons. *Advances in neural information processing systems*, 16.

Daniel Simig, Fabio Petroni, Pouya Yanki, Kashyap Popat, Christina Du, Sebastian Riedel, and Majid Yazdani. 2022. Open vocabulary extreme classification using generative models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1561–1583, Dublin, Ireland. Association for Computational Linguistics.

Liuyihan Song, Pan Pan, Kang Zhao, Hao Yang, Yiming Chen, Yingya Zhang, Yinghui Xu, and Rong Jin. 2020. Large-scale training system for 100-million classification at alibaba. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2909–2930.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Allen Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2023. A neural corpus indexer for document retrieval.

Yuanhao Xiong, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit Dhillon. 2022. Extreme Zero-Shot learning for extreme text classification. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5455–5468, Seattle, United States. Association for Computational Linguistics.

Nan Xu, Fei Wang, Mingtao Dong, and Muhao Chen. 2023. Dense retrieval as indirect supervision for large-space decision making. *arXiv preprint arXiv:2310.18619*.

Nishant Yadav, Rajat Sen, Daniel N. Hill, Arya Mazumdar, and Inderjit S. Dhillon. 2021. Session-aware query auto-completion using extreme multi-label ranking. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 3835–3844, New York, NY, USA. Association for Computing Machinery.

Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S Dhillon. 2022. Pecos: Prediction for enormous and correlated output spaces. *the Journal of machine Learning research*, 23(1):4233–4264.

Jiong Zhang, Wei cheng Chang, Hsiang fu Yu, and Inderjit S. Dhillon. 2021. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. 2023. Pytorch fsdp: Experiences on scaling fully sharded data parallel.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

Yaxin Zhu and Hamed Zamani. 2023. Icxml: An in-context learning framework for zero-shot extreme multi-label classification.

A Experiment Details

The deductive and inductive data used across each iteration of feedback learning is list in Table 5, with a distribution ratio of approximately 1:1. When using feedback data to adjust Vicuna-13B-v1.5, we set epoch to 1, learning rate to 2e-5 with no warm-up and batch size to 256 using FSDP (Zhao et al., 2023). All the experiments are implemented on NVIDIA A100-80GB GPU clusters.

Dataset	**Num**
AmazonCat-13K	$448k$
LF-Amazon-131K	$149k$
LF-WikiSeeAlso-320K	$159k$

Table 5: The instance-label pairs used for training XMC-AGENT

Policy at first stage	**Recall**	**Size**
Faiss (base performance)	53.7	300
BFS w/ *ancestor aug*	60.0	219.7
BFS w/o *ancestor aug*	**68.9**	220.5
DFS w/ *ancestor aug*	53.2	192.0
DFS w/o *ancestor aug*	<u>59.3</u>	179.6

Table 6: Impact of different navigation policies on the shortlist obtained in the first stage.

B Ablating the Navigation Policy

To investigate the impact of navigation policy on the results, we experiment with multiple combinations of strategies on AmazonCat-13K†. Due to the second-stage navigation strategy adopting an end-to-end approach to sequentially generate relevant labels from the shortlist, we only experiment with the first-stage strategy. We evaluate the effectiveness of the navigation policy from two aspects: 1) The recall of the first stage, denoted as **Recall**, where a higher proportion of relevant labels in the shortlist obtained in the first stage implies a smaller performance loss in subsequent processing. 2)The number of labels in the obtained shortlist, denoted as **Size**, where a higher number of labels in the shortlist leads to higher subsequent processing costs.

We employed two distinct navigation policies: 1) Breadth-First Search (**BFS**): This policy explores the label index in a breadth-first manner, employing a queue to store upcoming sub-indices for search initiation upon detection of a terminal index during any iteration, and continuing until completion of the process. 2) Depth-First Search (**DFS**): This policy explores the index in a depth-first manner, utilizing a stack to retain the next sub-indices for search initiation upon detection of a terminal index during any iteration. And we terminate the navigation process upon detecting 20 terminal indices.

When navigating over the label index, we employ two different methods to represent the sub-index currently being compared: 1) Only utilizing the description center of the sub-index currently being confronted (i.e., `Dance`, `Music` or `Sports`). 2) Providing a series of descriptions centers traversed from the root to the current sub-index, denoted as *ancestor aug*, i.e., [`Root -> Arts -> Dance`].

From the results in Table 6 we can observe that compared with retrieved top 300 similar labels using Faiss, employing a breadth-first manner navigation policy achieved a higher recall rate while retrieving fewer labels. Furthermore, despite the additional information offered by ancestor augmentation, it does not enhance the recall rate of navigation results. This phenomenon is attributed to the information from common ancestors enhancing the similarity between different sub-indexes, thus diminishing their distinctiveness.

C Full results for Linear Search

Considering the scale of the label set, we traverse all tags in AmazonCat-13K† in a point-wise manner, sorting the labels based on the output logits. We conducted experiments using both zero-shot and few-shot (k=1, 3, 5) approaches. When using the few-shot approach, for each label, we randomly select k instances related to that label from the training set to construct demonstrations. We then employ the large language models to determine the relevance between the label and the input instance, and we rank all labels based on the logits of the response. The full results are present in Table 7 and the comparison results with the previous method and XMC-AGENT are shown in Figure 7. From the results, it can be observed that employing LLMs in a point-wise manner has achieved comparable recall rates to the previous method, with slightly lower precision rates. However, the Linear Search approach incurs high time costs due to the need to traverse all labels for each instance. XMC-AGENT improves search speed by constructing a scalable

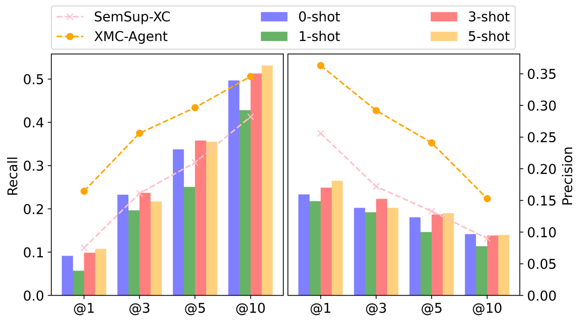

Figure 7: The comparison of Linear Search (k=0, 1, 3, 5) with SemSup-XC and XMC-AGENT on AmazonCat-13K†

hierarchical label index and employing feedback learning to adjust the navigational capability, which simultaneously enhances precision.

D Full results for ablation study

The full results for ablation study are present in Table 8 and Table 9.

Linear Search	Inc								Overall							
	Precision				Recall				Precision				Recall			
	P@1	**P@3**	**P@5**	**P@10**	**R@1**	**R@3**	**R@5**	**R@10**	**P@1**	**P@3**	**P@5**	**P@10**	**R@1**	**R@3**	**R@5**	**R@10**
Zero-Shot	16.0	13.8	12.3	9.7	9.2	23.3	33.7	49.7	21.6	21.0	20.2	16.5	5.6	19.7	30.9	49.7
1-Shot	14.9	13.1	10.0	7.8	5.7	19.7	25.1	42.8	37.8	27.9	23.8	17.9	15.0	28.1	38.7	54.6
3-Shot	17.0	15.2	12.8	9.5	9.9	23.7	35.8	50.3	34.2	28.2	24.5	18.2	12.0	27.5	38.9	55.3
5-Shot	18.1	13.8	13.0	9.6	10.8	21.7	35.5	50.1	37.8	27.9	23.8	17.9	15.0	28.1	38.7	54.6

Table 7: Employ Vicuna-13B-v1.5 in zero-shot and few-shot (k=1, 3, 5) manner to to determine the relevance between the label and the input instance.

Method	**Inc**								**Overall**							
	Precision				**Recall**				**Precision**				**Recall**			
	P@1	**P@3**	**P@5**	**P@10**	**R@1**	**R@3**	**R@5**	**R@10**	**P@1**	**P@3**	**P@5**	**P@10**	**R@1**	**R@3**	**R@5**	**R@10**
Ablating Label Index																
XMC-AGENT	36.3	29.2	24.1	15.3	24.1	37.5	43.4	50.6	80.1	64.2	50.3	33.3	22.8	39.6	51.0	62.7
Replace LLM Index with K-Means Index	17.3	12.7	9.0	6.2	9.6	15.1	20.4	24.4	15.6	13.1	8.7	6.5	10.8	15.7	22.0	25.3
Replace LLM Index with Faiss Top 500	20.2	15.3	10.3	5.7	10.4	16.5	22.5	34.1	20.0	16.5	13.1	8.4	17.0	21.6	28.5	36.9
Ablating Feedback Learning																
XMC-AGENT	36.3	29.2	24.1	15.3	24.1	37.5	43.4	50.6	80.1	64.2	50.3	33.3	22.8	39.6	51.0	62.7
Adopt Inductive Reasoning	26.6	23.7	18.3	13.0	22.8	36.4	42.1	49.3	57.5	45.7	40.1	26.3	18.2	33.3	46.9	58.1]
Adopt Deductive Reasoning	31.5	26.6	19.4	11.9	22.3	36.2	41.2	47.5	60.4	47.8	37.7	26.0	18.3	33.8	43.2	56.7
Adopt None (base performance)	23.3	20.1	14.8	10.2	21.5	35.8	38.4	42.2	44.3	38.6	32.8	19.3	17.3	30.1	39.7	42.5

Table 8: Component-wise ablation results of XMC-AGENT on AmazonCat-13K†. Ablating Label Index refers to replacing the self-construct label index with a K-Means index and a shortlist composed of the top 500 labels retrieved by Faiss to investigate the impact of label index on the final performance. Ablating Feedback Learning represents separately employing one feedback mechanism during iterative feedback learning to investigate the influence of the feedback mechanism.

Method	**Inc**								**Overall**							
	Precision				**Recall**				**Precision**				**Recall**			
	P@1	**P@3**	**P@5**	**P@10**	**R@1**	**R@3**	**R@5**	**R@10**	**P@1**	**P@3**	**P@5**	**P@10**	**R@1**	**R@3**	**R@5**	**R@10**
Ablating Label Index																
XMC-AGENT	24.8	18.3	13.1	8.1	21.4	32.0	39.9	45.5	22.7	18.9	13.7	10.2	26.1	25.7	34.3	46.5
Replace LLM Index with K-Means Index	19.9	8.7	7.8	6.6	10.3	17.7	26.2	34.6	17.1	16.8	8.7	5.5	8.4	14.5	20.7	25.2
Replace LLM Index with Faiss Top 500	20.2	15.3	10.3	5.7	10.4	16.5	22.5	34.1	20.0	16.5	13.1	8.4	17.0	21.6	28.5	36.9
Ablating Feedback Learning																
XMC-AGENT	24.8	18.3	13.1	8.1	21.4	32.0	39.9	45.5	22.7	18.9	13.7	10.2	26.1	25.7	34.3	46.5
Adopt Inductive Reasoning	21.6	16.5	11.3	7.8	20.2	30.7	36.4	42.8	19.5	16.8	12.3	10.0	19.5	16.8	12.3	10.0
Adopt Deductive Reasoning	22.4	17.2	11.1	7.4	20.2	29.5	34.2	42.1	19.0	17.0	12.6	9.7	19.1	25.5	33.2	43.4
Adopt None (base performance)	17.6	14.8	9.8	6.1	16.7	25.7	33.1	39.6	17.3	15.5	10.8	7.2	18.4	25.7	29.4	40.8

Table 9: Component-wise ablation results of XMC-AGENT on LF-Amazon-131K†. Ablating Label Index refers to replacing the self-construct label index with a K-Means index and a shortlist composed of the top 500 labels retrieved by Faiss to investigate the impact of label index on the final performance. Ablating Feedback Learning represents separately employing one feedback mechanism during iterative feedback learning to investigate the influence of the feedback mechanism.