Hierarchical Query Classification in E-commerce Search

Bing He
bhe46@gatech.edu
Georgia Institute of Technology
Atlanta, GA, USA

Sreyashi Nag
sreyanag@amazon.com
Amazon
Palo Alto, CA, USA

Limeng Cui
culimeng@amazon.com
Amazon
Palo Alto, CA, USA

Suhang Wang
szw494@psu.edu
The Pennsylvania State University
University Park, PA, USA

Zheng Li
amzzhe@amazon.com
Amazon
Palo Alto, CA, USA

Rahul Goutam
rgoutam@amazon.com
Amazon
Palo Alto, CA, USA

Zhen Li
amzzhn@amazon.com
Amazon
Palo Alto, CA, USA

Haiyang Zhang
hhaiz@amazon.com
Amazon
Palo Alto, CA, USA

ABSTRACT

E-commerce platforms typically store and structure product information and search data in a hierarchy. Efficiently categorizing user search queries into a similar hierarchical structure is paramount in enhancing user experience on e-commerce platforms as well as news curation and academic research. The significance of this task is amplified when dealing with sensitive query categorization or critical information dissemination, where inaccuracies can lead to considerable negative impacts. The inherent complexity of hierarchical query classification is compounded by two primary challenges: (1) the pronounced class imbalance that skews towards dominant categories, and (2) the inherent brevity and ambiguity of search queries that hinder accurate classification.

To address these challenges, we introduce a novel framework that leverages hierarchical information through (i) enhanced representation learning that utilizes the contrastive loss to discern fine-grained instance relationships within the hierarchy, called "instance hierarchy", and (ii) a nuanced hierarchical classification loss that attends to the intrinsic label taxonomy, named "label hierarchy". Additionally, based on our observation that certain unlabeled queries share typographical similarities with labeled queries, we propose a neighborhood-aware sampling technique to intelligently select these unlabeled queries to boost the classification performance. Extensive experiments demonstrate that our proposed method is better than state-of-the-art (SOTA) on the proprietary Amazon dataset, and comparable to SOTA on the public datasets of Web of Science and RCV1-V2. These results underscore the efficacy of our proposed solution, and pave the path toward the next generation of hierarchy-aware query classification systems.

WWW '24 Companion, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0172-6/24/05.
https://doi.org/10.1145/3589335.3648332

CCS CONCEPTS

• **Computing methodologies** → **Semi-supervised learning settings**; **Natural language processing**; • **Information systems** → **Web mining**.

KEYWORDS

Query Classification, Hierarchical Text Classification, Semi-supervised Learning

ACM Reference Format:
Bing He, Sreyashi Nag, Limeng Cui, Suhang Wang, Zheng Li, Rahul Goutam, Zhen Li, and Haiyang Zhang. 2024. Hierarchical Query Classification in E-commerce Search. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3589335.3648332

1 INTRODUCTION

Hierarchical query classification is a vital task in the domain of e-commerce and search, playing a crucial role in driving customer obsession [8]. As users interact with online services, they input various queries to search for products, services, or information. Accurately classifying these queries is pivotal in ensuring that users are presented with the most relevant and valuable results. One significant application of the hierarchical query classifier in industry is categorizing sensitive queries that follow a predefined hierarchy in e-commerce. For example, given a query, it can be classified as harmful, adult-oriented, or non-sensitive products (Here, for illustration, we define these categories as parent categories). Furthermore, for harmful products, there are two child categories: self-harm and harm to others. The child categories for the adult-oriented category can be adult products and adult content. Since these queries contain offensive content or pertain to unregulated goods, and different categories need to be handled differently, mis-classification of such queries can lead to unpleasant or even detrimental user experiences, potentially damaging a service's reputation and user trust. Moreover, presenting inappropriate or restricted content could lead to legal ramifications for the service. Hence, building an accurate hierarchical query classification framework is of paramount importance,

not just for user satisfaction, but also for the overall compliance and integrity of the service [31].

Various machine learning techniques are employed to identify the appropriate hierarchical category for each query [8, 41, 42], and sense the context and nuances each query presents [38]. However, these algorithms usually require large-scale high-quality annotated data, which is challenging to obtain. Instead, the more practical semi-supervised setting has gained popularity [3, 17], where unlabeled queries are used to boost the classification performance. When executed effectively, a well-performing hierarchical query classifier enhances the user experience, fostering a smoother and more productive interaction between users and the service.

However, building an accurate hierarchical query classification framework in real-world is non-trivial due to two challenges: (1) *severe class imbalance*. Take Amazon as an example, sensitive queries are infrequent, accounting for less than 0.05%~0.15% of all queries. Even worse, when training the classification model, only a small initial set of sensitive queries is accessible. This greatly hinders the development of a high-quality classifier. (2) *typically short and ambiguous query text in search queries*. The average search query is about three words [11], leading to a weaker semantic understanding of queries for correct classification.

To overcome these two challenges, we propose a semi-supervised machine learning framework utilizing the instance hierarchy and label hierarchy to enhance query representation learning and classification performance. Particularly,

(1) for the class imbalance challenge, we use contrastive learning to learn representations that attend to the minor classes through instance hierarchy. Intuitively, even if the number of queries under a child category can be small, the number of queries under the corresponding parent category is large and these queries are close to each other. To leverage them, we adopt contrastive learning where we create positive pairs from the queries under the same child category while negative pairs from cross-child categories. We formulate it as an intra-class hierarchy in instance hierarchy, and further extend it to the inter-class hierarchy, where we consider positives as queries across child categories and negatives as queries across parent categories, as shown in Figure 1. This instance hierarchy helps capture fine-grained information for model training.

(2) for the ambiguous and short text challenge, we utilize the information from three parts to enhance the understanding of a query — i.e., (i) the neighboring queries that are under the same child category. Motivated by the fact that not all queries are short and ambiguous, especially, some neighboring queries are clear and distinguishable, we can leverage this similarity between neighboring queries by the aforementioned intra-class hierarchical contrastive learning; (ii) the neighboring child categories that share the same parent category. The intuition is that when implicitly aligning the query to different child categories under the same parent category, the model learns the semantics of the query from other queries. This is achieved by adding a hierarchy-aware loss in our classification task; (iii) the text information in the label usually contains useful signals like the semantic meaning that contributes to the downstream classification. Based on this, we first adopt BERT [9] to encode the label text into an embedding vector to capture the contextualized semantic embedding. We concurrently create a "label" graph to take hierarchies/relationships between labels into account and employ the previously generated embedding vectors as node features for downstream graph representation learning. We finally combine the representation vector with the query embedding vector to form the finalized feature vector of a query for the final classification task. Since we use hierarchical label information in this step, we define this process as label hierarchy. Altogether, the designed instance hierarchy and label hierarchy components aim to address the aforementioned challenges for better classification.

In the real-world e-commerce search, we have one observation that there exist many unlabeled queries that share the typographical similarity to the annotated queries of the same category due to potential typos. These queries, when used as training examples, potentially assist the classifier by improving the robustness against mis-typed queries, as they are typographically close to the queries to augment the dataset. Based on this finding, we deploy the self-training learning stage in our pipeline, i.e., we use the pseudo labels of classified queries to retrain the model. When selecting queries for self-training, inspired by the aforementioned observation, we develop neighborhood-aware sampling to effectively identify high-quality similar queries. We argue that the observation of topographical similarity can be extended to generic semantic similarity. The proposed self-training pipeline can be adapted as well. Besides, we can interpret this observation from the adversarial learning perspective where we adversarially generate similar queries or identify similar queries from our unlabeled queries for model training to improve the robustness of the classifier. Overall, we deploy the self-training in our framework to utilize the crucial unannotated queries for classification performance gain.

To evaluate the proposed method, we examine it on proprietary Amazon data and public Web of Science and RCV1-V2 datasets using Micro-F1 and Macro-F1 scores. Our proposed method achieves the best performance in most cases across all compared methods and datasets, except for Micro-F1 on Web of Science and RCV1-V2 dataset. However, Micro-F1 is less critical than Macro-F1 in real-world applications since we have an imbalanced class distribution and need to focus on minority classes, which is attended to by Macro-F1. Our result demonstrates the efficacy of our proposed method, especially on the Amazon dataset. Our method is generalizable to solve hierarchical query classification tasks in all domains and paves the path toward the next generation of hierarchy-aware query classification. The main contributions of our work are:

- We propose a new algorithm that utilizes the instance and label hierarchy through contrastive learning-enhanced representation learning, which allows us to leverage hierarchical information in a fine-grained manner to improve classification performance.
- We propose a neighborhood-aware sampling technique to selectively choose high-quality unlabeled data points for self-training boost.
- Extensive experimental results on both proprietary and public datasets demonstrate the superiority of our proposed method in most cases.

2 RELATED WORKS

In this section, we briefly introduce relevant research areas.

2.1 Hierarchical Query Classification

Hierarchical query classification aims at classifying queries into a category within a given taxonomy to understand user intent and facilitate downstream recommendation tasks. It can be formulated as a text classification problem where the input text is a combination of short keywords [41]. Existing conventional methods employ either a single flattened multi-class classifier or multiple binary classifiers. Based on extracted query features, these works can be categorized into two groups: (1) N-gram-based features [6]: Since the query keywords are indicative of the category it belongs to, the count of keywords can serve as features; (2) Embedding-based features [9, 29]: Due to the advances in deep learning and natural language processing, some researchers use word embeddings (e.g., Glove) and contextualized embeddings (e.g., BERT) to represent the query for classification. Later, researchers designed advanced classification models and learning diagrams utilizing additional information from queries to enhance classifiers. For instance, Liu et al. [23] proposed a mixture of conventional neural network and Naive Bayes as a classifier [23] while Wang et al. [36] incorporated the hierarchy of label information by a graph encoder into the text encoder [36]. Besides, the context-aware session information [5] and searcher engagement data [14] are explored as well. Different from these existing efforts relying on extra information or overlooking abundant unlabeled data, we aim to boost performance using only easily accessible query and label data combined with unlabeled queries.

2.2 Imbalanced Learning

Class imbalance is a common issue in text classification, especially prominent in the hierarchical setting [7]. To address it, one commonly-used solution is re-sampling, which involves oversampling the minority class, under-sampling the majority class, or combining both to achieve a balanced class distribution [24]. Another strategy is cost-sensitive learning [33], where higher costs are assigned to the misclassification of minority classes during model training, eventually making the model more sensitive to the minority class. The Synthetic Minority Over-sampling Technique is another notable approach that generates synthetic instances of the minority class to balance the class distribution [7]. For instance, Pereira et al. [30] utilized the path and depth information to oversample and undersample data points to improve the classifier. Different from the previous works, we utilize the unlabeled queries that are predicted as minority classes to augment datasets.

2.3 Contrastive Learning

Contrastive learning has emerged as a powerful paradigm in unsupervised and self-supervised learning techniques [15] by significantly reducing the performance gap between supervised and unsupervised learning. At its core, contrastive learning aims to learn similar representations for semantically similar instances and dissimilar representations for distinct ones. It accomplishes this by distinguishing the "positive" pairs (two similar data points) from the "negative" pairs (two dissimilar data points). Especially by leveraging large amounts of unlabeled data, it opens up new avenues for model training in scenarios where labeled data is scarce or expensive to obtain. The effectiveness of this approach has been showcased in numerous applications, such as image, speech recognition, and natural language processing [21].

2.4 Semi-supervised Text Classification

Semi-supervised learning is a promising research direction since it utilizes both the labeled and unlabeled data points in machine learning [34], which alleviates the high cost of data annotation. Among the semi-supervised learning techniques, self-training is widely used where a model is initially trained on a limited set of labeled data points, and then iteratively expands the training datasets by using classified unlabeled data points [38]. Different selection methods are developed to choose which data points for augmentation, including probability-based and uncertainty-based solutions [1, 1, 32]

3 PROBLEM DEFINITION

In this section, we provide the mathematical definition of the hierarchical query classification problem. Note that, for the sake of simplicity in presentation, we assume the problem space is a two-level category hierarchy, but, the proposed method is extensible to accommodate a multi-level category hierarchy.

We have a set of queries $Q = \{q_1, q_2, ..., q_N\}$, where q_i is the i-th query. A query is a sequence of words, represented as $q_i = x_1, x_2, ..., x_i, ...$, where x_i is the i-th word in a query. We can divide the query set Q into two groups: unlabeled queries $Q_U = \{q_1^U, q_2^U, q_3^U, ...\}$ and labeled queries $Q_L = \{q_1^L, q_2^L, q_3^L ...\}$. For one labeled query, we have its child category c_k and parent category p_j where p_j denotes the parent category from a set of parent categories $P = \{p_1, p_2, p_3, ...\}$ and each parent category p_j consists of a set of child categories $p_j = \{c_1, c_2, c_3, ...\}$. In this case, we have $c_k \in p_j$ The goal is to leverage the information in both labeled and unlabeled queries to learn a function $\mathcal{F}(q_i) \rightarrow c_k, p_j$, where $q_i \in c_k, q_i \in p_j$, and $c_k \in p_j$.

4 PROPOSED METHOD

In this section, we provide the details of the proposed semi-supervised hierarchical query classification framework to accurately classify a query for a given taxonomy. Specifically, the framework has three major components: i) It utilizes the hierarchical label information to enhance initial query embeddings. ii) It attends to fine-grained instance hierarchy by modeling intra-class and inter-class relationships. The resultant contrastive loss boosts the query embedding learning, which is finally combined with a classification loss to train the classifier. iii) Through the proposed neighborhood-aware sampling technique, it selectively chooses high-quality unlabeled data points with pseudo labels to augment existing labeled data for model re-training. An overview of our proposed method is presented in Figure 1.

4.1 Label Hierarchy

Given a query q_i, we pass it to BERT to get the textual embedding emb_{q_i}, following existing works to generate feature vectors [9, 26].

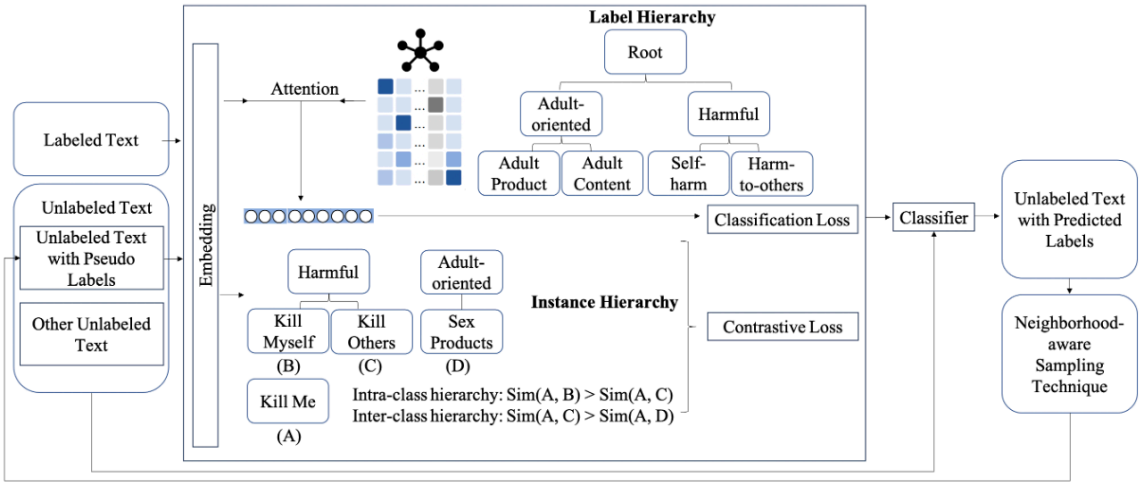

Figure 1: The overview of the proposed framework.

To attend to the hierarchy, we first create a label graph $G = (V, E)$ representing the taxonomic hierarchy, where V is the vertex set of labels and E is the edge set of connections between parent and child labels. Because the label text (e.g., "self-harm", "adult products") contains useful semantic information for the downstream classification, we follow similar approaches to transfer text into embedding vectors [13, 28], passing the label text to BERT to get the textual embedding as the node feature vector. For the root node, we use the average of all label embedding vectors. Due to the advances in graph neural networks for graph representation [39], in practice, we pass the graph to a conventional two-layer graph convolutional network [19] to get the embedding as:

$$emb_G = \text{GCN}(G)$$

Note that, not all label embeddings are equally relevant for a query. Motivated by the attention mechanism [35], we compute the attention score between a query and each label embedding. Formally, we have an attention matrix:

$$A_{q_i} = \text{Softmax}((emb_{q_i} \cdot W) \cdot emb_G^T)$$

where W is a matrix for feature dimension alignment between emb_{q_i} and emb_G during matrix multiplication. Then, we compute the attention-weighted label features, denoted as

$$emb^l_{q_i} = A_{q_i} \times emb_G.$$

We finally concatenate it with query textual embedding emb_{q_i} derived by BERT and form the final representation $emb^f_{q_i} = [emb_{q_i}, emb^l_{q_i}]$ for the downstream classification task. Particularly, for query q_i, we predict its label as:

$$\hat{y}_i = f(emb^f_{q_i}).$$

In addition, to leverage the hierarchical label information, we use the neighboring child label information to assist the classification by aligning the predicted child category to the neighboring child category. Such signal is incorporated into the model by a loss derived from between the current child category and the neighboring child category as:

$$\sum_{j \in p_i} \log(\hat{y}_j)$$

where p_i denotes the parent category of c_i. By combing the two loss together, we have the final classification loss as:

$$\mathcal{L}_{\text{Classification}} = \sum_i [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) + \lambda \cdot \sum_{j \in p_i} \log(\hat{y}_j)]$$

where y_i is the label of the query and λ adjusts the importance of between the two losses. In implementation, we empirically set $\lambda = 1$.

4.2 Instance Hierarchy

Given query q_i, to learn its comprehensive representation in the context of its hierarchical structure, we use contrastive learning at two levels: (1) the intra-class hierarchy; and (2) the inter-class hierarchy. Particularly, for the intra-class level, we randomly sample two queries within the same child category as one positive pair $(q_i, q_j), q_i \in c_i, q_j \in c_i$. For the query pairs that are in the same parent category but from different child categories, we treat them as negative pairs $(q_i, q_k), q_i \in c_i, q_k \notin c_i, q_i \in p_k, q_j \in p_k$. The contrastive objective is defined as:

$$\mathcal{L}_{\text{intra-class}} = -\log\left(\frac{\exp(\text{sim}(emb_{q_i}, emb_{q_k}))}{\sum \exp(\text{sim}(emb_{q_i}, emb_{q_j}))}\right),$$

where $\text{sim}(u, v) = \frac{u^T v}{||u||_2 ||v||_2}$ denotes the cosine similarity between two vectors. Similarly, for the inter-class level, we have the predefined (q_i, q_k) query pair as the positive and the query pairs that are in different parent categories as negatives $(q_i, q_l), q_i \in p_k, q_l \notin p_k$. The corresponding contrastive loss is:

$$\mathcal{L}_{\text{inter-class}} = -\log\left(\frac{\exp(\text{sim}(emb_{q_i}, emb_{q_l}))}{\sum \exp(\text{sim}(emb_{q_i}, emb_{q_k}))}\right).$$

Combining them together, the contrastive loss is given as:

$$\mathcal{L}_{\text{contrastive}} = w_{\text{intra-class}} \cdot \mathcal{L}_{\text{intra-class}} + (1 - w_{\text{intra-class}}) \cdot \mathcal{L}_{\text{inter-class}}$$

where $w_{\text{intra-class}}$ denotes the weight of the intra-class hierarchy in the contrastive loss.

4.3 Objective Function and Model Training

Finally, we combine the classification and contrastive loss together, which arrives at:

$$\mathcal{L} = w_{\text{contrastive}} \cdot \mathcal{L}_{\text{Contrastive}} + (1 - w_{\text{contrastive}}) \cdot \mathcal{L}_{\text{Classification}}$$

where $w_{\text{contrastive}}$ indicates the weight of contrastive loss in the final loss computation.

When training the model, we minimize the loss for optimization through back-propagation using the Adam optimizer [18].

4.4 Neighborhood-aware Sampling

After training, we apply the classifier $\mathcal{F}$ to predict labels of the unlabeled data Q_U and then use the classified high-confidence data points to retrain the classifier. Motivated by our aforementioned observation that queries with similar labels tend to share similar typographical representations, we develop a neighborhood-aware sampling algorithm containing the following steps:

Given an unlabeled query $q_i^U \in Q_U$, after inference by $\mathcal{F}(q_i^U)$, we have the predicted child category and parent category:

$$\hat{c}_{q_i^U}, \hat{p}_{q_i^U} = \mathcal{F}(q_i^U)$$

.

Step I: We use the K-Nearest Neighbors (KNN) to find the labeled queries similar to the unlabeled queries in the feature space. Here, motivated by the aforementioned observation, the feature space can be the simple N-length string space and we compute Levenshtein Distance (Edit Distance) neighborhood search [4]. More generally, we use the previously generated BERT embedding to represent the feature space due to its powerful semantic representation in a broader case. Formally, we have:

$$\text{Neighbor}_{q_i^U} = \{q_j | q_j \in \text{KNN}(q_i^U), q_j \in Q_L\}$$

where q_j is from the labeled query set Q_L and its child category is c_j and parent category is p_j. In practice, we utilize the hierarchical navigable small world method for the indexing and search process because of its efficiency in high-dimensional data spaces, making it a suitable choice for large-scale and high-dimensional datasets [27]. To measure the similarity between queries, we adopt the cosine similarity metric. By using these two off-the-shelf solutions, we aim to achieve an efficient and accurate KNN search process.

Step II: After getting the neighboring labeled queries, we need to compute their distribution for the downstream sampling. To achieve this goal, we leverage the child category information between labeled and unlabeled queries. The intuition is that if the unlabeled query shares the same child category of the labeled query, chances are high that this predicted child category is correct. Specifically, we compute KL divergence scores between the child category of the labeled query q_j^L, denoted as $c_{q_j^L}$, and the predicted child category of unlabeled query q_i^U, denoted as $\hat{c}_{q_i^U}$. This score is denoted as KL distance:

$$KL_{L-U_{\text{child}}} = \text{KL}(c_{q_j^L}, \hat{c}_{q_i^U})$$

Since there are multiple labels in the neighborhood of unlabeled query q_i^U, it is necessary to take the quality of these queries into consideration during the sampling. Formally, for $q_j^L \in \text{Neighbor}_{q_i^U}$, the average child category label is given as:

$$\bar{c}_{\text{Neighbor}_{q_i^U}} = \frac{1}{|\text{Neighbor}_{q_i^U}|} \sum c_{q_j}^L$$

We then compute the divergence score as:

$$KL_{L-L_{\text{child}}} = \text{KL}(c_{q_j^L}, \bar{c}_{\text{Neighbor}_{q_i^U}}).$$

Finally, we add the scores from the child category information together as the final distribution:

$$\text{Dist}(\text{Neighbor}_{q_i^U})_{\text{child}} = KL_{L-U_{\text{child}}} + KL_{L-L_{\text{child}}}$$

Step III: Similarly, we compute the distribution based on the parent category information and get $KL_{L-U_{\text{parent}}} = \text{KL}(p_{q_j^L}, \hat{p}_{q_i^U})$ and $KL_{L-L_{\text{parent}}} = \text{KL}(p_{q_j^L}, \bar{p}_{\text{Neighbor}_{q_i^U}})$, where $\bar{p}_{\text{Neighbor}_{q_i^U}} = \frac{1}{|\text{Neighbor}_{q_i^U}|} \sum p_{q_j^L}$. After addition, we have

$$\text{Dist}(\text{Neighbor}_{q_i^U})_{\text{parent}} = KL_{L-U_{\text{parent}}} + KL_{L-L_{\text{parent}}}$$

Finally, we have the sampling distribution as:

$$\begin{aligned}\text{Dist}(\text{Neighbor}_{q_i^U}) = {} & w_{\text{child}} \cdot \text{Dist}(\text{Neighbor}_{q_i^U})_{\text{child}} \\ & + (1 - w_{\text{child}}) \cdot \text{Dist}(\text{Neighbor}_{q_i^U})_{\text{parent}}\end{aligned}$$

Step IV: We sample the unlabeled queries following:

$$\text{Prob} \propto \text{Dist}(\text{Neighbor}_{q_i^U})$$

We then add the sampled data points: $\{(q_i^U, \hat{c}_{q_i^U}, \hat{p}_{q_i^U})\}$ to our existing labeled queries:

$$Q_L = Q_L + \{(q_i^U, \hat{c}_{q_i^U}, \hat{p}_{q_i^U})\}$$

to retrain the classifier. After re-training, we run the neighborhood-aware sampling again to select new queries to augment the existing labeled datasets. We repeat these steps until the model converges.

5 EXPERIMENTAL EVALUATION

In this section, we examine the performance of the proposed framework by conducting extensive experiments. Specifically, we aim to answer the following research questions:

- RQ1: How effective is our proposed method when compared to other methods?
- RQ2: What is the contribution of each component in the proposed framework?
- RQ3: How sensitive is the model performance when we change the parameters?

5.1 Datasets

We evaluate the proposed framework on both public and proprietary datasets.

5.1.1 Public Datasets. We adopt two benchmark datasets widely used for hierarchical text classification, including Web-of-Science and ECV1-V2. The data statistics is shown in Table 1.

- Web-of-Science (WoS) [20]: WOS dataset contains keywords and abstracts of academic papers across several disciplines, e.g., economy and science. For one paper, it also has a hierarchical domain-area label, representing the hierarchical nature of the discipline

Table 1: Data statistics of public datasets.

Public Dataset Name	# of Classese	# of Hierarchy
Web of Science (WoS)	141	2
RCV1-V2	103	4

(e.g., Computer Vision domain under Computer Science category). This makes WOS suitable for the hierarchical query classification task where we treat keywords, area, and domain as query, child category, and parent category, respectively.

- RCV1-V2 [22]: The RCV1-V2 dataset is a benchmark corpus for text categorization research where each document has metadata such as date and title, in addition to the content of the news story. It comprises an archive of over 800,000 manually categorized newswire stories from Reuters Ltd. Its hierarchical categorization scheme includes four main topics (Corporate/Industrial, Economics, Government/Social, and Markets), which are further divided into subtopics, leading to over 100 leaf-level categories. Thus, it is an excellent dataset for hierarchical classification tasks. We use extracted nouns from titles, subtopic, and main topic as query, child category, and parent category, respectively.

5.1.2 Proprietary Dataset. For proprietary dataset, we use Amazon search queries as our testbed for examination on real-world application settings. We sampled 9~10 million user queries to create the dataset. In the Amazon dataset, a substantial portion consists of unlabeled queries, making up 40% to 50% of the total. Labeled non-sensitive queries also represent a significant segment, comprising 45% to 58% of the data. Within the labeled sensitive categories, queries related to adult-oriented products form 3% to 6%, while adult content is less common, constituting only 0.3% to 0.5%. The dataset also includes a small fraction of queries that are potentially harmful, with those related to self-harm and harm to others present in 0.003% to 0.005% and 0.01% to 0.03%, respectively. The remaining sensitive queries count for 0.04% to 0.07%.

5.2 Evaluation Setup

5.2.1 Evaluation Metrics. Since it is a standard imbalanced data classification task, we follow the existing related works and measurement: the Micro and Macro F1 score [36, 37].

5.2.2 Compared Methods. We compare with the standard multi-class text classifiers using fine-tuned BERT. Besides, several state-of-the-art (SOTA) hierarchical text classifiers using transfer learning and prompt learning are examined [2, 16, 25, 36, 40]. Namely, we compare with (1) HPT [37], where prompt tuning on pre-trained language model is utilized to handle hierarchical classification from a multi-label masked language model perspective; (2) HGCLR [36], where new queries will be generated by the label hierarchy to enhance the query representation learning using the contrastive loss; and (3) HiTIN [40], where the label hierarchy is converted into an unweighted tree structure to enhance the query representation.

In implementation, to utilize the unlabeled queries for a fair comparison, we add the widely-used confidence-based sampling strategy in the self-training stage for each compared method. This ensures that all methods are compared in the same semi-supervised setting. For the train/val/testing split, we follow the existing split for the public Web of Science and RCV1-V2 datasets, and an 80-10-10 split for the Amazon dataset. For unlabeled data points, we use the existing unlabeled data points in the Amazon dataset and 10% of the whole public Web of Science and RCV1-V2 datasets.

Table 2: Comparison of hierarchical query classification performance. The best algorithm in each row is colored in dark blue and the second best is light blue. Note that we present the baseline result on Amazon as "0" for relative comparison (Here, the baseline is BERT), and ± indicates that the corresponding method is above or below the baseline.

Dataset	Metric	BERT	HPT	HGCLR	HiTIN	Ours
Amazon	Micro-F1	0	-0.31	+2.90	+2.91	+3.26
	Macro-F1	0	+0.67	+2.56	+3.35	+4.10
WoS	Micro-F1	47.98	44.27	51.73	54.21	53.19
	Macro-F1	42.93	43.71	48.92	47.93	50.54
RCV1-V2	Micro-F1	74.96	73.07	76.57	76.95	76.91
	Macro-F1	58.49	58.78	59.25	60.84	61.48

5.3 Evaluation Results

5.3.1 RQ1: Effectiveness of the proposed framework. As we see the comparison results in Table 2, our proposed method is the best in most cases except Micro-F1 on Web of Science and RCV1-V2 dataset.

This demonstrates the efficacy of our proposed method, especially on the Amazon dataset. In detail, we find that our proposed method beats the baseline fine-tuned BERT with the largest margin, indicating the necessity of designing sophisticated approaches for performance gain (i.e., instance hierarchy, label hierarchy, and neighborhood-aware sampling). We also beat the advanced HPT and HGCLR solutions. The reason may be that we explicitly consider the instance hierarchy to model the relationship between queries while HPT and HGCLR focus more on the hierarchical label structure. Our neighborhood-based sampling technique also contributes by selecting high-quality data points for self-training. Even if we are weaker than HiTIN regarding Micro-F1 on Web of Science and RCV1-V2, we are better in Macro-F1, which is more crucial. This is because in the real-world application setting, like the sensitive query classification on Amazon, critical categories have fewer data points and we should treat each category equally rather than each data point equally during evaluation. This is achieved by the Macro-F1 score.

5.3.2 RQ2: Ablation studies. To examine the contribution of each component in the proposed framework (i.e., label hierarchy, instance hierarchy, and neighborhood-based sampling in the self-training stage), we first remove one component in the framework. Then, we retrain the model and measure the classification performance.

As shown in Table 3, our proposed method with all components is better than any revised method that is removing one component. This demonstrates the necessity of each component in the pipeline. Interestingly, we find removing the label hierarchy leads to the largest performance drop, possibly because the additional information from the label text shares certain similarities with the

Table 3: Ablation studies of our proposed framework. Note that we present our method on the Amazon dataset as "0" for relative comparison, and ± indicates that the corresponding ablated method is above or below our method.

Dataset	Metric	Ours	Ours w/o label hierarchy	Ours w/o instance hierarchy	Ours w/o self-training
Amazon	Micro-F1	0	-4.92	-1.56	-1.06
	Macro-F1	0	-6.25	-1.04	-1.94
WoS	Micro-F1	53.19	48.17	51.41	52.27
	Macro-F1	50.54	47.95	48.02	49.16

query text embedding, thereby contributing to the model training. The instance hierarchy and self-training stage contribute to the performance to a similar degree.

5.3.3 RQ3: Hyperparameter tuning. In this section, we examine the effect of hyperparameters on the model performance. Here, we focus on three major parameters $w_{\text{intra-class}}$, $w_{\text{contrastive}}$, and w_{child}. The value ranges from 0.01 to 1 and we report the performance in Table 4.

Table 4: Effects of varying weights on Amazon dataset. Note that we present the result by our method (i.e., $w_{intra-class} = 0.1$, $w_{contrastive} = 0.01$, and $w_{child} = 0.1$) on the Amazon dataset as the baseline, denoted as "0" for relative comparison, and ± indicates that the corresponding method configured with certain parameter values is above or below the compared method. Δ Micro-F1 means the difference in the Micro-F1 score and Δ Macro-F1 means the difference in the Micro-F1 score.

Parameter	Value	Δ Micro-F1	Δ Macro-F1
$w_{intra-class}$	0.1	0	0
	0.3	-0.78	-0.64
	0.5	+0.10	+1.67
	0.7	+0.19	+1.02
	0.9	+0.62	+2.66
	1	+0.12	+2.04
$w_{contrastive}$	0.01	0	0
	0.1	+1.47	+2.25
	0.3	+0.95	+0.98
	0.5	-0.42	-1.01
	0.7	+0.28	-2.64
	0.9	-3.66	-6.49
w_{child}	0.1	0	0
	0.3	+0.69	+0.25
	0.5	+0.35	-0.15
	0.7	+0.24	-0.19
	0.9	+0.25	-0.21
	1	-0.14	-0.34

As we can see, (1) for $w_{\text{inter-class}}$, the best value is 0.9 and the higher value leads to better performance except 1, which indicates that the intra-class hierarchy contributes more than the inter-class hierarchy. The reason can be that intra-class helps the model learn the representation better for the downstream query classification; (2) for $w_{\text{contrastive}}$, we find when we add the contrastive loss to the classification loss, it helps improve the classification performance. But, the weight should not be large. 0.1 works best in the current setup. (3) for w_{child}, we see that 0.3 is the best. It implies when utilizing both child and parent category information in the sampling stage, we should carefully choose and tune the weight. All these results indicate the contribution of each corresponding component. When using them together, we should take caution and exhaustively test different values to find the proper hyperparameter set for the specific application setting.

6 APPLICATION IN PRACTICE

We launched the proposed model in our sensitive query detection platform on Amazon, which is used in the search module. We compare the proposed method with our previous rule-based production model. We sample queries detected as positive by the model, and ask the human labeling team to measure precision before and after the launch. The results show that our method is better.

7 CONCLUSION AND LIMITATION

In this work, we propose a novel hierarchical query classification framework to effectively classify short queries into different groups. In essence, we first utilize label and instance hierarchy patterns to derive the classification and contrastive loss to train the model. We then design a neighborhood-aware sampling method to intelligently utilize unlabeled queries with pseudo labels to boost the model performance for self-training. Extensive results on both proprietary and public datasets demonstrate the effectiveness of our model.

However, our work still suffers from a few limitations. First, since our method is a multi-stage framework rather than an automatic end-to-end solution. Manual configuration and monitoring of the model training are needed, especially, during the self-training stage to determine the number of high-quality data points for the model retraining. Second, in the use case of sensitive query classification, users can purposefully write queries to bypass or attack the classifier such that the classification performance drops [10, 12]. We plan to explore this in future work. Third, Large Language Models have gained popularity in recent years due to their promising results across multiple applications including text classification and text generation. Our method can beat ChatGPT during our preliminary examination of the Web of Science dataset. The potential reason is that our task is more challenging than the conventional text classification due to the complex label hierarchy structure. But, more efforts are required to accomplish a thorough comparison. Fourth, our method still requires a large number of annotated data points. In the real-world application, especially the sensitive query classification on Amazon, there are only few annotated data points for certain categories. Few-shot learning can be a possible direction for future research.

REFERENCES

[1] Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Emilie Devijver, and Yury Maximov. 2022. Self-training: A survey. *arXiv preprint arXiv:2202.12040* (2022).
[2] Siddhartha Banerjee, Cem Akkaya, Francisco Perez-Sorrosal, and Kostas Tsioutsiouliklis. 2019. Hierarchical transfer learning for multi-label text classification.

In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 6295–6300.

[3] Steven M Beitzel, Eric C Jensen, Ophir Frieder, David D Lewis, Abdur Chowdhury, and Aleksander Kolcz. 2005. Improving automatic query classification via semi-supervised learning. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 8–pp.

[4] Bonnie Berger, Michael S Waterman, and Yun William Yu. 2020. Levenshtein distance, sequence comparison and biological database search. *IEEE transactions on information theory* 67, 6 (2020), 3287–3294.

[5] Huanhuan Cao, Derek Hao Hu, Dou Shen, Daxin Jiang, Jian-Tao Sun, Enhong Chen, and Qiang Yang. 2009. Context-aware query classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 3–10.

[6] William B Cavnar, John M Trenkle, et al. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, Vol. 161175. Las Vegas, NV, 14.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[8] Shui-Lung Chuang and Lee-Feng Chien. 2002. Towards automatic generation of query taxonomy: A hierarchical query clustering approach. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* IEEE, 75–82.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751* (2017).

[11] Rand Fishkin. 2017. The State of Searcher Behavior Revealed Through 23 Remarkable Statistics. https://moz.com/blog/state-of-searcher-behavior-revealed

[12] Bing He, Mustaque Ahamad, and Srijan Kumar. 2021. Petgen: Personalized text generation attack on deep sequence embedding-based classification models. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 575–584.

[13] Bing He, Caleb Ziems, Sandeep Soni, Naren Ramakrishnan, Diyi Yang, and Srijan Kumar. 2021. Racism is a virus: Anti-Asian hate and counterspeech in social media during the COVID-19 crisis. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 90–94.

[14] Yunzhong He, Cong Zhang, Ruoyan Kong, Chaitanya Kulkarni, Qing Liu, Ashish Gandhe, Amit Nithianandan, and Arul Prakash. 2023. HierCat: Hierarchical Query Categorization from Weakly Supervised Data at Facebook Marketplace. In *Companion Proceedings of the ACM Web Conference 2023*. 331–335.

[15] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A survey on contrastive self-supervised learning. *Technologies* 9, 1 (2020), 2.

[16] Ashiqur R KhudaBukhsh, Paul N Bennett, and Ryen W White. 2015. Building effective query classifiers: a case study in self-harm intent detection. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 1735–1738.

[17] Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Scalable semi-supervised query classification using matrix sketching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 8–13.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

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

[21] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. 2020. Contrastive representation learning: A framework and review. *Ieee Access* 8 (2020), 193907–193934.

[22] David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research* 5, Apr (2004), 361–397.

[23] Xianjing Liu, Hejia Zhang, Mingkuan Liu, and Alan Lu. 2019. System Design of Extreme Multi-label Query Classification using a Hybrid Model.. In *eCOM@ SIGIR*.

[24] Rushi Longadge and Snehalata Dongre. 2013. Class imbalance problem in data mining review. *arXiv preprint arXiv:1305.1707* (2013).

[25] Qianwen Ma, Chunyuan Yuan, Wei Zhou, and Songlin Hu. 2021. Label-specific dual graph neural network for multi-label text classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 3855–3864.

[26] Yingchen Ma, Bing He, Nathan Subrahmanian, and Srijan Kumar. 2023. Characterizing and Predicting Social Correction on Twitter. In *Proceedings of the 15th ACM Web Science Conference 2023*. 86–95.

[27] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[28] Nicholas Micallef, Bing He, Srijan Kumar, Mustaque Ahamad, and Nasir Memon. 2020. The role of the crowd in countering misinformation: A case study of the COVID-19 infodemic. In *2020 IEEE international Conference on big data (big data)*. IEEE, 748–757.

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

[30] Rodolfo M Pereira, Yandre MG Costa, and Carlos N Silla Jr. 2021. Toward hierarchical classification of imbalanced data using random resampling algorithms. *Information Sciences* 578 (2021), 344–363.

[31] Dou Shen, Ying Li, Xiao Li, and Dengyong Zhou. 2009. Product query classification. In *Proceedings of the 18th ACM conference on information and knowledge management*. 741–750.

[32] Jafar Tanha, Maarten Van Someren, and Hamideh Afsarmanesh. 2017. Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics* 8 (2017), 355–370.

[33] Nguyen Thai-Nghe, Zeno Gantner, and Lars Schmidt-Thieme. 2010. Cost-sensitive learning methods for imbalanced data. In *The 2010 International joint conference on neural networks (IJCNN)*. IEEE, 1–8.

[34] Jesper E Van Engelen and Holger H Hoos. 2020. A survey on semi-supervised learning. *Machine learning* 109, 2 (2020), 373–440.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[36] Zihan Wang, Peiyi Wang, Lianzhe Huang, Xin Sun, and Houfeng Wang. 2022. Incorporating hierarchy into text encoder: a contrastive learning approach for hierarchical text classification. *arXiv preprint arXiv:2203.03825* (2022).

[37] Zihan Wang, Peiyi Wang, Tianyu Liu, Binghuai Lin, Yunbo Cao, Zhifang Sui, and Houfeng Wang. 2022. HPT: Hierarchy-aware prompt tuning for hierarchical text classification. *arXiv preprint arXiv:2204.13413* (2022).

[38] Zhiquan Ye, Yuxia Geng, Jiaoyan Chen, Jingmin Chen, Xiaoxiao Xu, SuHang Zheng, Feng Wang, Jun Zhang, and Huajun Chen. 2020. Zero-shot text classification via reinforced self-training. In *Proceedings of the 58th annual meeting of the association for computational linguistics*. 3014–3024.

[39] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI open* 1 (2020), 57–81.

[40] Jie Zhou, Chunping Ma, Dingkun Long, Guangwei Xu, Ning Ding, Haoyu Zhang, Pengjun Xie, and Gongshen Liu. 2020. Hierarchy-aware global model for hierarchical text classification. In *Proceedings of the 58th annual meeting of the association for computational linguistics*. 1106–1117.

[41] Sanduo Zhou, Kefei Cheng, and Lijun Men. 2017. The survey of large-scale query classification. In *AIP conference proceedings*, Vol. 1834. AIP Publishing.

[42] Lvxing Zhu, Kexin Zhang, Hao Chen, Chao Wei, Weiru Zhang, Haihong Tang, and Xiu Li. 2023. HCL4QC: Incorporating Hierarchical Category Structures Into Contrastive Learning for E-commerce Query Classification. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 3647–3656.