

TELEClass: Taxonomy Enrichment and LLM-Enhanced Hierarchical Text Classification with Minimal Supervision

Yunyi Zhang
University of Illinois
Urbana-Champaign, IL, USA
yzhan238@illinois.edu

Ruozhen Yang*
University of Illinois
Urbana-Champaign, IL, USA
ruozhen2@illinois.edu

Xueqiang Xu*
University of Illinois
Urbana-Champaign, IL, USA
xx19@illinois.edu

Jinfeng Xiao
University of Illinois
Urbana-Champaign, IL, USA
jxiao13@illinois.edu

Jiaming Shen
Google Research
NY, USA
jmshen@google.com

Jiawei Han
University of Illinois
Urbana-Champaign, IL, USA
hanj@illinois.edu

ABSTRACT

Hierarchical text classification aims to categorize each document into a set of classes in a label taxonomy. Most earlier works focus on fully or semi-supervised methods that require a large amount of human annotated data which is costly and time-consuming to acquire. To alleviate human efforts, in this paper, we work on hierarchical text classification with the minimal amount of supervision: using the sole class name of each node as the only supervision. Recently, large language models (LLM) show competitive performance on various tasks through zero-shot prompting, but this method performs poorly in the hierarchical setting, because it is ineffective to include the large and structured label space in a prompt. On the other hand, previous weakly-supervised hierarchical text classification methods only utilize the raw taxonomy skeleton and ignore the rich information hidden in the text corpus that can serve as additional class-indicative features. To tackle the above challenges, we propose TELEClass, Taxonomy Enrichment and LLM-Enhanced weakly-supervised hierarchical text Classification, which (1) automatically enriches the label taxonomy with class-indicative topical terms mined from the corpus to facilitate classifier training and (2) utilizes LLMs for both data annotation and creation tailored for the hierarchical label space. Experiments show that TELEClass can outperform previous weakly-supervised hierarchical text classification methods and LLM-based zero-shot prompting methods on two public datasets.

KEYWORDS

Weakly-Supervised Text Classification, Hierarchical Text Classification, Taxonomy Enrichment, Large Language Model

1 INTRODUCTION

Hierarchical text classification, aiming to classify documents into one or multiple classes in a label taxonomy, is an essential and fundamental task in text mining and natural language processing. Compared with standard text classification where label space is flat and relatively small (e.g., less than 20 classes), hierarchical text classification is more challenging given the larger and more structured label space and the existence of fine-grained and long-tail classes in the taxonomy. Most earlier works tackle this task in fully supervised [6, 15, 44] or semi-supervised settings [4, 13], and

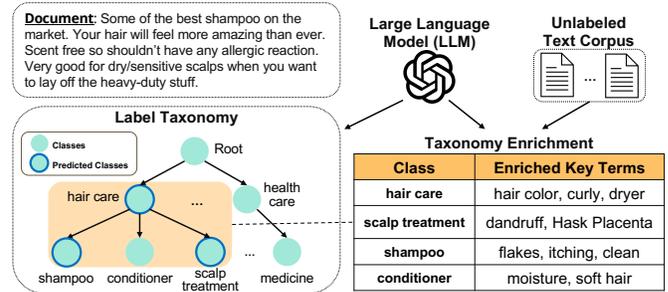


Figure 1: An example document tagged with 3 classes. We automatically enrich each node with class-indicative terms and utilize LLMs to facilitate classification.

different models are proposed to learn from a substantial amount of human-labeled data. However, acquiring human annotation is often costly, time-consuming, and not scalable.

Recently, large language models (LLM) such as GPT-4 [25] and Llama2 [37] have demonstrated strong performance in flat text classification through zero-shot or few-shot prompting [35]. However, applying LLMs in hierarchical settings, with large and structured label spaces, remains challenging. Directly including hundreds of classes in prompts is ineffective, leading to structural information loss, increased computational costs, and diminished clarity for LLMs in focusing on critical information. Along another line of research, Meng et al. [23] propose to train a moderate-size text classifier by utilizing a small set of keywords or labeled documents for each class. However, compiling keyword lists for hundreds of classes and obtaining representative documents for each specific and niche category still demand significant human efforts. Shen et al. [31] study the hierarchical text classification with *minimal supervision*, which takes the class name as the only supervision signal. Specifically, they introduce TaxoClass, a method that generates pseudo-labels with a textual entailment model for classifier training. However, this method overlooks additional class-relevant features in the text and suffers from unreliable pseudo-label selection due to the entailment model's domain shift.

In this study, we advance minimally supervised hierarchical text classification by integrating corpus-based taxonomy enrichment and leveraging LLMs tailored for the hierarchical label structure.

*Equal Contribution.

First, we enrich the existing label taxonomy skeleton with automatically extracted class-specific topical terms from the corpus, enhancing supervision signals and thus improving the pseudo-label quality for classifier training. Second, we explore the application of LLMs in hierarchical text classification from two perspectives. Specifically, we enhance LLM annotation efficiency and effectiveness through a taxonomy-guided candidate search and optimize LLM-generated document accuracy by using taxonomy paths to create more precise pseudo-data.

Leveraging the above ideas, we introduce TELEClass: Taxonomy Enrichment and LLM-Enhanced weakly-supervised hierarchical text Classification. TELEClass consists of four major steps: (1) *LLM-Enhanced Core Class Annotation*, where we identify document “core classes” (i.e., fine-grained classes that most accurately describe the documents) by first using a textual entailment model for top-down candidate search on the taxonomy and then prompting an LLM to select the most precise core classes. (2) *Corpus-Based Taxonomy Enrichment*, where we analyze the taxonomy structure and the roughly classified documents to identify class-indicative topical terms through semantic and statistical analysis, enriching the label taxonomy with more features. For example, the “conditioner” class in Figure 1 is enriched with key terms like “moisture” and “soft hair”, which distinguish it from its sibling classes. (3) *Core Class Refinement with Enriched Taxonomy*, where we embed documents and classes based on the enriched label taxonomy and refine the initially selected core classes by identifying the most similar classes for each document. (4) *Text Classifier Training with Path-Based Data Augmentation*, where we sample label paths from the taxonomy and guide the LLM to generate pseudo documents most accurately describing these fine-grained classes. Finally, we train the text classifier on two types of pseudo-labels, the core classes and the generated data, with a simple text-matching network and multi-label training strategy.

The contributions of this paper are summarized as follows: (1) We propose TELEClass, a new method for minimally supervised hierarchical text classification, which requires only the class names of the label taxonomy as supervision to train a multi-label text classifier. (2) We propose to enrich the label taxonomy with class-indicative topical terms mined from the text corpus, based on which we utilize an embedding-based document-class matching method to improve the pseudo-label quality. (3) We study two ways of adopting large language models to the hierarchical text classification setting, which can improve the pseudo-label quality and solve the data scarcity issue for fine-grained classes. (4) Extensive experiments on two datasets show that TELEClass can outperform strong weakly-supervised hierarchical text classification methods and zero-shot LLM prompting methods.¹

2 PROBLEM DEFINITION

Following Shen et al. [31], we define the minimally-supervised hierarchical text classification task as to train a text classifier that can categorize each document into multiple nodes on a label taxonomy by using the name of each node as the only supervision. For example, in Figure 1, the input document is classified as “hair care”, “shampoo”, and “scalp treatment”.

Formally, the task input includes an unlabeled text corpus $\mathcal{D} = \{d_1, \dots, d_{|\mathcal{D}|}\}$ and a directed acyclic graph $\mathcal{T} = (C, \mathcal{R})$ as the label taxonomy. Each $c_i \in C$ represents a target class in the taxonomy, coupled with a unique textual surface name s_i . Each edge $\langle c_i, c_j \rangle \in \mathcal{R}$ indicates a hypernymy relation, where class c_j is a sub-class of c_i . For example, one such edge in Figure 1 is between $s_i = \text{“hair care”}$ and $s_j = \text{“shampoo”}$. Then, the goal of our task is to train a multi-label text classifier $f(\cdot)$ that can map a document d into a binary encoding of its corresponding classes, $f(d) = [y_1, \dots, y_{|C|}]$, where $y_i = 1$ represents d belongs to class c_i , otherwise $y_i = 0$.

3 METHODOLOGY

In this section, we will introduce TELEClass consisting of the following modules: (1) LLM-enhanced core class annotation, (2) corpus-based taxonomy enrichment, (3) core class refinement with enriched taxonomy, and (4) text classifier training with path-based data augmentation. Figure 2 shows an overview of TELEClass.

3.1 LLM-enhanced Core Class Annotation

Inspired by previous studies, we first tag each document with its “core classes”, which are defined as a set of classes that can describe the document most accurately [31]. This process also mimics the process of human performing hierarchical text classification: first, select a set of most essential classes for the document and then trace back to their ancestors to complete the labeling. For example, in Figure 1, by first tagging the document with “shampoo” and “scalp treatment”, we can easily find its complete set of classes.

To mine the core classes for each document, existing methods utilize a textual entailment model (e.g., RoBERTa-large-MNLI) and combine tree-based top-down candidate searching with corpus-level statistical analysis for core class selection. Specifically, the similarity score between a document and a class is defined as the entailment score from the document to a class-specific hypothesis such as “*This example is [Class Name]*”. These similarity scores facilitate a top-down search in the taxonomy to shortlist candidate classes, which are then evaluated across the corpus to determine the most reliable core classes.

However, there are two limitations of this method. First, the textual entailment model, not being directly trained for document classification, offers inconsistent zero-shot performance, particularly in distinguishing closely related, fine-grained classes within a hierarchical structure. Second, given a pair of premise (document) and hypothesis, the entailment model is trained to predict a distribution over three classes, namely entailment, neutral, and contradiction. Therefore, it is questionable whether the generated scores can support corpus-level comparison, because the scores between different pairs may not be directly comparable. Both limitations will lead to noisy core class selection and adversely impact the classifier performance.

In this work, we propose to utilize an LLM to enhance the core class annotation step. These LLMs are trained on very large textual data and human preference data and shown to have superior zero-shot ability for the document classification task with a small and flat label space [35]. However, LLMs cannot effectively understand the hierarchical label space, which is large and structured and cannot be easily represented in a prompt. Therefore, we precede the LLM

¹Code and datasets are available upon request now and will later be available online.

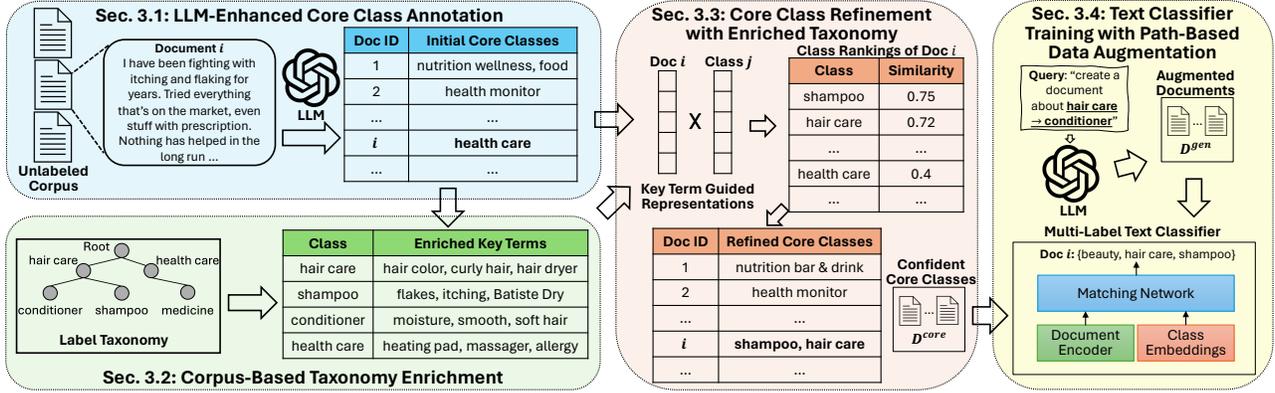


Figure 2: Overview of the TELEClass framework.

annotation with the top-down candidate search method using a textual entailment model.

Specifically, for each document d_i , we start from the Root node at level 0 of the taxonomy by adding it into a queue. For each class at level l in the queue, we select $l + 3$ of its children that have the highest similarity to document d_i , where the similarity score $sim^{entail}(c, d_i)$ is obtained from a textual entailment model. Then, among all the selected children at level $l + 1$, we keep the top $(l + 2)^2$ of them with the highest path-based similarity scores and add to the queue, where the path-based similarity $sim^{path}(c, d_i)$ is defined recursively as,

$$sim^{path}(\text{Root}, d) = 1,$$

$$sim^{path}(c, d) = \max_{c_p \in \text{Par}(c)} sim^{path}(c_p, d_i) \cdot sim^{entail}(c, d_i),$$

where $\text{Par}(c)$ is the set of all parents of c . This process finishes when there is no class in the queue, and all the nodes (other than Root) ever pushed into the queue are treated as candidate core classes for document d_i . To use an LLM to annotate core classes, we first construct an instruction for each document, which asks the LLM to select the most accurate classes for the document from its candidate core classes. Then, the document is fed to the LLM as a query and it will generate a list of core classes according to the document and candidate core classes.

Our LLM-enhanced core class annotation step will assign an initial set of core classes (denoted as C_i^0) for each document $d_i \in \mathcal{D}$. In the next two steps, we introduce how TELEClass leverages the initial core classes to enrich the label taxonomy with class-indicative features. This enriched taxonomy is subsequently employed to refine the initial core classes, utilizing embedding-based similarity scores that are more comparable across documents.

3.2 Corpus-Based Taxonomy Enrichment

Due to the hierarchical nature of taxonomy, fined-grained classes at lower levels are often hard to distinguish with only class label names. Therefore, we propose to enrich the weak supervision, i.e. the label taxonomy, with a set of *class-indicative topical terms* for each class. These enriched terms can help better interpret the class and thus facilitate text classification. For example, in Figure 2, “shampoo” and “conditioner” are two fine-grained classes that are similar to each other. We can effectively separate the two classes by identifying

a set of class-specific terms such as “flakes” for “shampoo” and “moisture” for “conditioner”.

In this work, we combine statistical document occurrence features and semantic similarity and further contrast a class with its siblings to identify a set of key terms that can distinguish it from its most similar nodes in the taxonomy. Notice that, because the input taxonomy \mathcal{T} is a directed acyclic graph rather than a simple tree, a node c can have multiple sets of siblings, each of which corresponds to one of c 's parent nodes. Besides, because we have already tagged each document with its core classes, we can use them as rough classification results and form a document cluster for each class to help the enrichment step.

Formally, given a class $c \in \mathcal{C}$ and its siblings corresponding to one of its parents c_p , $Sib(c, c_p) = \{c' \in \mathcal{C} | \langle c_p, c' \rangle \in \mathcal{R}\}$, $c_p \in \text{Par}(c)$, we find a set of *class-indicative topical terms* of c corresponding to c_p , denoted as $T(c, c_p) = \{t_1, t_2, \dots, t_k\}$. Each term in $T(c, c_p)$ can signify the class c and distinguish it from its siblings under c_p . To identify these class-indicative terms, we construct a set of documents D_c^0 for each class c , containing all the documents whose initial core classes contain c or its descendants,

$$D_c^0 = \{d_i \in \mathcal{D} | \exists c' \in C_i^0, c' \in \{c\} \cup \text{Des}(c)\}, \quad (1)$$

where $\text{Des}(c)$ is the set of all descendants of c . We consider the following 3 factors for class-indicative term selection: *popularity*, *distinctiveness*, and *semantic similarity*.

Popularity. First, a class-indicative term of c should be frequently mentioned by the documents about c , denoted as its popularity within c . We calculate the popularity of a term t within a class c as,

$$pop(t, c) = \log(1 + df(t, D_c^0)), \quad (2)$$

where the document frequency $df(t, D)$ stands for the number of documents in D that mention t .

Distinctiveness. Second, a class-indicative term t for a class c should also be infrequent in its siblings so that a document containing t likely indicates that it is about c instead of c 's siblings. We define the distinctiveness score of a term t for a class c , corresponding to its siblings under c_p , as,

$$dist(t, c, c_p) = \frac{\exp(BM25(t, D_c^0))}{1 + \sum_{c' \in Sib(c, c_p)} \exp(BM25(t, D_{c'}^0))}, \quad (3)$$

where $BM25(\cdot, \cdot)$ denotes the BM25 relevance function [29].

Semantic Similarity. Third, we also require a class-indicative term to be semantically similar to the class name. We use a small pretrained language model (e.g., BERT-base-uncased [7]) to encode each term and class name. Specifically, for each w , either a class name or a term from the corpus, we feed it into a PLM as [CLS] w [SEP]. Because w may be tokenized into multiple word pieces by the PLM, we take the average of the final layer embeddings of all its pieces as the embedding of w , denoted as \mathbf{t} for a term t and \mathbf{c} for a class c . Then, cosine similarity is used to measure the semantic similarity between a term t and a class c .

Finally, we define the affinity score between a term t and a class c corresponding to parent p to be the geometric mean of the above scores, with a weight parameter α ,

$$aff(t, c, c_p) = pop(t, c)^\alpha \cdot dist(t, c, c_p)^{1-\alpha} \cdot \cos(\mathbf{t}, \mathbf{c}). \quad (4)$$

Taxonomy Enrichment. We first apply a phrase mining tool, AutoPhrase [30], to mine quality single-token and multi-token phrases from the corpus to serve as candidates for taxonomy enrichment. Then, for each class c and each of its parents p , we select the top- k terms with the highest affinity scores with c corresponding to p , denoted as $T(c, c_p)$. Then, we aggregate the terms corresponding to each parent of c to get the final enriched class-indicative topical terms for class c as follows

$$T_c = \bigcup_{c_p \in Par(c)} T(c, c_p). \quad (5)$$

3.3 Core Class Refinement with Enriched Taxonomy

With the enriched class-indicative terms for each class, we propose to further utilize them to refine the initial core classes. In this paper, we adopt an embedding-based document-class matching method, which has two distinct advantages in the hierarchical setting. (1) By pre-computing the document and class representations, calculating their similarity is much more efficient than the textual entailment-based method, promoting a more comprehensive score comparison on the entire taxonomy instead of just on a sub-tree. (2) While textual entailment model is trained to predict an entailment score for each pair of document and class, which follows an unknown distribution and is unreliable for corpus-level comparison, the embedding similarity is more general and easier to compare across classes and documents.

Unlike previous methods in flat text classification [39] that use keyword-level embeddings to estimate document and class representations, here, because we already have a set of roughly classified documents for each class, D_c^0 , we can directly define their representations based on document-level embeddings. To obtain document representations, we utilize a pretrained Sentence Transformer model [28] to encode the entire document, which we denote as \vec{d} . Then, for each class c , we identify a subset of its assigned documents which explicitly mention at least one of the class-indicative keywords and thus most confidently belong to this class, $D_c = \{d \in D_c^0 | \exists w \in T_c, w \in d\}$. Then, we use the average of their document embeddings as the class representation, $\vec{c} = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{d}$. Finally, we compute the document-class matching score as the cosine similarity between their representations.

Based on the document-class matching scores, we make an observation that the true core classes often have much higher matching scores with the document compared to other classes. Therefore, we use the largest ‘‘similarity gap’’ for each document to identify its core classes. Specifically, for each document $d_i \in \mathcal{D}$, we first get a ranked list of classes according to the matching scores, denoted as $[c_1^i, c_2^i, \dots, c_{|C|}^i]$, where $diff^i(j) := \cos(\vec{d}_i, \vec{c}_j^i) - \cos(\vec{d}_i, \vec{c}_{j+1}^i) > 0$ for $j \in \{1, \dots, |C| - 1\}$. Then, we find the position m_i with the highest similarity difference with its next one in the list. After that, we treat the classes ranked above this position as this document’s refined core classes \mathbb{C}_i , and the corresponding similarity gap as the confidence estimation $conf_i$.

$$conf_i = diff^i(m_i), \quad \mathbb{C}_i = \{c_1^i, c_2^i, \dots, c_{m_i}^i\}, \\ m_i = \arg \max_{j \in \{1, \dots, |C| - 1\}} diff^i(j). \quad (6)$$

Finally, we select the set of documents and their refined core classes with the top 50% confidence scores $conf_i$, to be the final set of refined core classes for classifier training. Formally, $\mathcal{D}^{core} = \{d_i\}$, where each $d_i \in \mathcal{D}^{core}$ satisfies $|\{d_j | conf_j < conf_i\}| \geq \frac{1}{2} |\mathcal{D}|$.

3.4 Text Classifier Training with Path-Based Data Augmentation

The final step of TELEClass is to train a hierarchical text classifier using the confident refined core classes. One straightforward way is to directly use the selected core classes as a complete set of pseudo-labeled documents and train a text classifier in a common supervised way. However, such a strategy is ineffective, because the core classes are not comprehensive enough and cannot cover all the classes in the taxonomy. The reason is two-fold. First, the core class selection step is document-centric, meaning that it only focuses on selecting the most accurate core classes for each document, while ignoring whether each class in the taxonomy is ever selected as a core class. Second, the label space in the hierarchical text classification task naturally contains some fine-grained and long-tail classes, so they are often not guaranteed to be selected as core classes due to their low frequency. Empirically, for the two datasets we use in our experiments, Amazon and DBPedia, the missing rates of the selected core classes are 11.6% and 5.4%, respectively. These missing classes will never be used as positive classes in the training process, if we only train the classifier with the selected core classes.

Therefore, to overcome this issue, we further utilize the language generation ability of LLMs to augment the current pseudo-labels. We propose the idea of *path-based document generation by LLMs* to generate augmented documents for each path from a level-1 node to a leaf node in the taxonomy. By adding the generated documents to the pseudo-labeled data, we can ensure the coverage of the pseudo-data. Moreover, we use a path instead of a single class to guide the LLM generation, because the meaning of lower-level classes is often conditioned on their parents. For example, in Figure 2, a path ‘‘hair care’’ \rightarrow ‘‘shampoo’’ can guide the LLM to generate text about hair shampoo instead of pet shampoo or carpet shampoo that are in different paths.

More specifically, for each path from a level-1 node to a leaf node in the taxonomy, $p = (c_1, \dots, c_l)$, we query an LLM to generate a small number of documents (e.g., $q = 5$) corresponding to the

path p and denote the resulting documents as d_i^p , $i = 1, \dots, q$. By generating q documents for each path and treating all the classes in the query path as the pseudo-labels for the generated documents, we get a set of generated data $\mathcal{D}^{\text{gen}} = \{d_i^p\}_{p \in \text{Path}(\mathcal{T}), i=1, \dots, q}$, where $\text{Path}(\mathcal{T})$ denotes all such paths in the taxonomy \mathcal{T} , and \mathbb{C}_p is the set of classes in the path p . \mathcal{D}^{gen} can serve as augmented data to train the text classifier together with the selected refined core classes $\mathcal{D}^{\text{core}}$. Notice that, here we generate pseudo documents for *each* path instead of just generating for the classes that have no or very few assigned documents. The reason is that, because our core classes are obtained based on static keyword-based features (i.e., the enriched class-indicative topical terms), it may induce certain bias of the matching process into the pseudo-data and affect the learning process [9]. Therefore, we use an LLM to generate pseudo documents for each path in the taxonomy, which can also serve as regularization of the potential bias.

Now, with two sets of pseudo-data prepared, we are ready to introduce the classifier architecture and the training process.

Classifier architecture. Because we do not focus on proposing a new model for the hierarchical text classification task, we choose to use a standard architecture as our text classifier, including a document encoder, class embeddings, and a matching network.

- Document encoder: we initialize the document encoder $g(\cdot)$ with a pretrained BERT-base model [7] and use the hidden representation of the [CLS] token from the last layer as the document encoding, $\mathbf{d}_i = g(d_i)$.
- Class embeddings: we initialize the class embeddings with the class surface name embeddings obtained by a pretrained BERT model (c.f. Section 3.2), denoted by c_j . Notice that, here we detach the embeddings from the base model, so only the embeddings will be updated during the training process without back-propagation to the backbone model.
- Matching Network: we utilize a log-bilinear matching network to calculate the probability of document d_i belonging to class c_j :

$$p(c_j|d_i) = \mathcal{P}(y_j = 1|d_i) = \sigma(\exp(\mathbf{c}_j^T \mathbf{W} \mathbf{d}_i)),$$

where σ stands for the sigmoid function and \mathbf{W} is a learnable interaction matrix.

Training process. For each document d_i tagged with core classes \mathbb{C}_i , we construct its positive classes $\mathbb{C}_{i,+}^{\text{core}}$ as the union of its core classes and their ancestors in the label taxonomy, and its negative classes $\mathbb{C}_{i,-}^{\text{core}}$ as the ones that are not positive classes or descendants of any core class. This is because the ancestors of confident core classes are also likely to be true labels, and the descendants may not all be negative given that the automatically generated core classes are not optimal. Formally,

$$\begin{aligned} \mathbb{C}_{i,+}^{\text{core}} &= \mathbb{C}_i \cup (\cup_{c \in \mathbb{C}_i} \text{Anc}(c)), \\ \mathbb{C}_{i,-}^{\text{core}} &= \mathcal{C} - \mathbb{C}_{i,+}^{\text{core}} - \cup_{c \in \mathbb{C}_i} \text{Des}(c), \end{aligned}$$

where $\text{Anc}(c)$ and $\text{Des}(c)$ denote the set of ancestors and descendants and class c , respectively. For the LLM-generated documents, we are more confident in its pseudo labels, so we simply treat all the classes in the corresponding path as positive classes and all other classes as negative,

$$\mathbb{C}_{p,+}^{\text{gen}} = \mathbb{C}_p, \quad \mathbb{C}_{p,-}^{\text{gen}} = \mathcal{C} - \mathbb{C}_p.$$

Algorithm 1: TELEClass

Input: A corpus \mathcal{D} , a label taxonomy \mathcal{T} , a textual entailment model \mathcal{M} , a sentence encoder \mathcal{S} , an LLM \mathcal{G} .
Output: A text classifier F that can classify each document into a set of classes in \mathcal{T} .

- 1 // LLM-Enhanced Core Class Annotation;
- 2 **for** $d_i \in \mathcal{D}$ **do**
- 3 $\mathbb{C}_i^0 \leftarrow$ use the LLM \mathcal{G} to select core classes from candidates retrieved by \mathcal{M} ;
- 4 // Corpus-Based Taxonomy Enrichment;
- 5 **for** $c \in \mathcal{C}$ **do**
- 6 $D_c^0 \leftarrow$ a set of roughly classified documents (Eq. 1);
- 7 **for** $c_p \in \text{Par}(c)$ **do**
- 8 $T(c, c_p) \leftarrow$ top terms ranked by affinity (Eq. 4);
- 9 $T_c \leftarrow$ aggregate class-indicative terms with Eq. 5;
- 10 // Core Class Refinement with Enriched Taxonomy;
- 11 $\vec{d} \leftarrow$ document representation $\mathcal{S}(d)$;
- 12 **for** $c \in \mathcal{C}$ **do**
- 13 $D_c \leftarrow$ confident documents by matching T_c ;
- 14 $\vec{c} \leftarrow$ average document representation in D_c ;
- 15 **for** $d_i \in \mathcal{D}$ **do**
- 16 $\mathbb{C}_i, \text{conf}_i \leftarrow$ refined core classes using $\cos(\vec{d}, \vec{c})$ and Eq. 6;
- 17 $\mathcal{D}^{\text{core}} \leftarrow$ confident refined core classes;
- 18 // Text Classifier Training with Path-Based Data Augmentation;
- 19 $\mathcal{D}^{\text{gen}} \leftarrow$ generate q documents for each path using \mathcal{G} ;
- 20 $F \leftarrow$ train classifier with $\mathcal{D}^{\text{core}}$, \mathcal{D}^{gen} , and Eq. 7;
- 21 **Return** F ;

Then, we train a multi-label classifier with the binary cross entropy loss (BCE):

$$\begin{aligned} \mathcal{L}^{\text{core}} &= - \sum_{d_i \in \mathcal{D}^{\text{core}}} \left(\sum_{c_j \in \mathbb{C}_{i,+}^{\text{core}}} \log p(c_j|d_i) + \sum_{c_j \in \mathbb{C}_{i,-}^{\text{core}}} \log (1 - p(c_j|d_i)) \right) \\ \mathcal{L}^{\text{gen}} &= - \sum_{d_i^p \in \mathcal{D}^{\text{gen}}} \left(\sum_{c_j \in \mathbb{C}_{p,+}^{\text{gen}}} \log p(c_j|d_i^p) + \sum_{c_j \in \mathbb{C}_{p,-}^{\text{gen}}} \log (1 - p(c_j|d_i^p)) \right) \\ \mathcal{L} &= \mathcal{L}^{\text{core}} + \frac{|\mathcal{D}^{\text{core}}|}{|\mathcal{D}^{\text{gen}}|} \cdot \mathcal{L}^{\text{gen}}, \end{aligned} \quad (7)$$

where $\frac{|\mathcal{D}^{\text{core}}|}{|\mathcal{D}^{\text{gen}}|}$ serves as a scaling factor to balance two sets of pseudo-data and can be exactly calculated as $\frac{|\mathcal{D}|}{2q \cdot |\text{Path}(\mathcal{T})|}$.

We train the text-matching model with \mathcal{L} to get the final classifier. Notice that, we do not continue to train the classifier with self-training that is commonly used in previous studies [23, 31], because we want to keep the classifier training part simple. Using self-training may further improve the model performance, which we leave for future exploration. Algorithm 1 summarizes TELEClass.

4 EXPERIMENTS

4.1 Experiment Setup

4.1.1 Datasets. We use two public datasets in different domains for evaluation. (1) **Amazon-531** [20] consists of Amazon product reviews and a three-layer label taxonomy of 531 product types, with 29,487 unlabeled training samples and 19,685 testing samples. (2) **DBpedia-298** [18] consists of Wikipedia articles with a three-layer

Table 1: Experiment results on Amazon-531 and DBPedia-298 datasets, evaluated by Example-F1, P@k, and MRR. The best score among zero-shot and weakly-supervised methods is boldfaced. “†” indicates the numbers for these baselines are directly from previous paper [31]. “—” means the method cannot generate a ranking of predictions and thus MRR cannot be calculated.

| Supervision Type | Methods | Amazon-531 | | | | DBPedia-298 | | | |
|-------------------|-----------------------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| | | Example-F1 | P@1 | P@3 | MRR | Example-F1 | P@1 | P@3 | MRR |
| Zero-Shot | Hier-0Shot-TC [†] | 0.4742 | 0.7144 | 0.4610 | — | 0.6765 | 0.7871 | 0.6765 | — |
| | GPT-3.5-turbo | 0.5164 | 0.6807 | 0.4752 | — | 0.4816 | 0.5328 | 0.4547 | — |
| Weakly-Supervised | Hier-doc2vec [†] | 0.3157 | 0.5805 | 0.3115 | — | 0.1443 | 0.2635 | 0.1443 | — |
| | WeSHClass [†] | 0.2458 | 0.5773 | 0.2517 | — | 0.3047 | 0.5359 | 0.3048 | — |
| | TaxoClass-NoST [†] | 0.5431 | 0.7918 | 0.5414 | 0.5911 | 0.7712 | 0.8621 | 0.7712 | 0.8221 |
| | TaxoClass [†] | 0.5934 | 0.8120 | 0.5894 | 0.6332 | 0.8156 | 0.8942 | 0.8156 | 0.8762 |
| | TELEClass | 0.6330 | 0.8439 | 0.6269 | 0.6664 | 0.8684 | 0.9293 | 0.8684 | 0.8880 |
| Fully-Supervised | | 0.8843 | 0.9524 | 0.8758 | 0.9085 | 0.9786 | 0.9945 | 0.9786 | 0.9826 |

label taxonomy of 298 categories, with 196,665 unlabeled training samples and 49,167 testing samples.

4.1.2 Compared Methods. We compare the following methods on the weakly-supervised hierarchical text classification task.

- **Hier-0Shot-TC** [43] is a zero-shot approach, which utilizes a pretrained textual entailment model to iteratively find the most similar class at each level for a document.
- **GPT-3.5-turbo** is a zero-shot approach that queries GPT-3.5-turbo by directly providing all classes in the prompt.
- **Hier-doc2vec** [17] is a weakly-supervised approach, which first trains document and class representations in the same embedding space, and then iteratively selects the most similar class at each level.
- **WeSHClass** [23] is a weakly-supervised approach using a set of keywords for each class. It first generates pseudo documents to pretrain text classifiers and then performs self-training.
- **TaxoClass** and its variant **TaxoClass-NoST** [31] is a weakly-supervised approach that only uses the class name of each class. It first uses a textual entailment model with a top-down search and corpus-level comparison to select core classes, which are then used as pseudo-training data. We include both its full model, TaxoClass, and its variation TaxoClass-NoST that does not apply self-training on the trained classifier, which is the same as ours.
- **TELEClass** is our newly proposed weakly-supervised approach that only uses the class name for each class.
- **Fully-Supervised** is a fully-supervised baseline that uses the entire labeled training data to train the text-matching network used in TELEClass.

4.1.3 Evaluation Metrics. Following previous studies [31], we utilize the following evaluation metrics: Example-F1, Precision at k (P@k) for $k = 1, 3$, and Mean Reciprocal Rank (MRR). See Appendix A for definitions of these metrics.

4.1.4 Implementation Details. We use RoBERTa-large-MNLI as the textual entailment model for core class candidate search. We query GPT-3.5-turbo-0613 for both core class annotation and path-based generation. For taxonomy enrichment, we get the term and class name embeddings using a pre-trained BERT-base-uncased. The weight parameter α is set to 0.2, and we select top $k = 20$

enriched terms for each class (c.f. Section 3.2). For core class refinement, we get the document and class embeddings using Sentence Transformer all-mpnet-base-v2 (c.f. Section 3.3). We generate $q = 5$ documents with path-based generation for each class. The document encoder in the final classifier is initialized with BERT-base-uncased. We train the classifier using AdamW optimizer with a learning rate $5e-5$, and the batch size is 64. We include prompts we used for LLM annotation and generation in Appendix B.

4.2 Experimental Results

Table 1 shows the evaluation results of all the compared methods. We make the following observations. (1) Overall, TELEClass achieves the best performance among the compared strong zero-shot and weakly-supervised baselines. (2) By comparing with other weakly-supervised methods, we find that TELEClass significantly outperforms TaxoClass-ST, which is the strongest baseline that also does not use self-training as TELEClass. This further demonstrates the effectiveness of TELEClass. (3) Although LLM (e.g., GPT-3.5-turbo) show superior power in many tasks, naively prompting GPT-3.5-turbo in the hierarchical text classification task does not yield promising performance and underperforms strong weakly-supervised text classifier by a large margin. We will conduct a more detailed comparison of LLM prompting for hierarchical text classification in Section 4.4.

4.3 Ablation Studies

To better understand if each component of TELEClass contributes to the entire method, we additionally conduct ablation studies by removing part of TELEClass and reporting the performance. Table 2 shows the results of the following ablations of TELEClass:

- **Gen-Only** only uses the augmented documents produced by path-based LLM generation to train the final classifier.
- **TELEClass-NoEnrich** does not perform taxonomy enrichment, so directly uses the initial core classes annotated by LLM plus augmented documents to train the final classifier.
- **TELEClass-NoGen** does not use the augmented documents produced by path-based LLM generation, so only uses the refined core classes by taxonomy enrichment and embedding-based matching to train the final classifier.

Table 2: Performance of TELEClass and its ablations on Amazon-531 and DBpedia-298 datasets. The best score is boldfaced.

| Methods | Amazon-531 | | | | DBpedia-298 | | | |
|--------------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| | Example-F1 | P@1 | P@3 | MRR | Example-F1 | P@1 | P@3 | MRR |
| Gen-Only | 0.5151 | 0.7477 | 0.5096 | 0.5357 | 0.7930 | 0.9421 | 0.7930 | 0.8209 |
| TELEClass-NoEnrich | 0.6324 | 0.8375 | 0.6263 | 0.6730 | 0.6868 | 0.7726 | 0.6868 | 0.7182 |
| TELEClass-NoGen | 0.6084 | 0.8238 | 0.6024 | 0.6361 | 0.8364 | 0.9146 | 0.8364 | 0.8599 |
| TELEClass | 0.6330 | 0.8439 | 0.6269 | 0.6664 | 0.8684 | 0.9293 | 0.8684 | 0.8880 |

Table 3: Performance comparison of zero-shot LLM prompting. We only report Example-F1 and P@k, because it is not straightforward to get ranking of classes predicted by LLMs for MRR calculation. We also report estimated cost in US dollar (as of the pricing in Feb 2024) and running time in minutes for each method on the entire test set. “‡” indicates that we report the performance on a 1,000-document subset of test data.

| Methods | Amazon-531 | | | | | DBpedia-298 | | | | |
|-----------------------|------------|--------|--------|-----------|-----------|-------------|--------|--------|-----------|------------|
| | Example-F1 | P@1 | P@3 | Est. Cost | Est. Time | Example-F1 | P@1 | P@3 | Est. Cost | Est. Time |
| GPT-3.5-turbo | 0.5164 | 0.6807 | 0.4752 | \$60 | 240 mins | 0.4816 | 0.5328 | 0.4547 | \$80 | 400 mins |
| GPT-3.5-turbo (level) | 0.6621 | 0.8574 | 0.6444 | \$20 | 800 mins | 0.6649 | 0.8301 | 0.6488 | \$60 | 1,000 mins |
| GPT-4‡ | 0.6994 | 0.8220 | 0.6890 | \$800 | 400 mins | 0.6054 | 0.6520 | 0.5920 | \$2,500 | 1,000 mins |
| TELEClass | 0.6330 | 0.8439 | 0.6269 | <\$1 | 3 mins | 0.8684 | 0.9293 | 0.8684 | <\$1 | 7 mins |

We find that the full model TELEClass achieves the overall best performance among the compared methods, showing the effectiveness of each of its components. Interestingly, we find that taxonomy enrichment and path-based data generation make different levels of contribution on two data sets: path-based data generation brings more improvement on Amazon-531, while taxonomy enrichment contributes more on DBpedia-298. The reasons could be as follows. Amazon-531 has a large label space and more skewed class distribution. We find that around 11.6% of classes are not selected as core classes in Amazon, while this number for DBpedia is 5.4%. This necessitates data augmentation to provide training data that covers each class in the taxonomy. On the other hand, although DBpedia has a smaller label space, its class names are more subtle to distinguish, which can also be demonstrated by the lower performance of zero-shot LLM prompting on DBpedia compared to Amazon-531 (c.f. GPT-3.5-turbo in Table 1). Therefore, by mining class-indicative terms, TELEClass can enrich each class with more clues to facilitate better classification.

4.4 Comparison with Zero-Shot LLM Prompting

In this section, we further compare TELEClass with zero-shot LLM prompting. Because it is not straightforward to get ranked predictions by LLMs, we only report Example-F1 and P@k as performance evaluations. Additionally, we report the estimated cost and time for each method on the entire test set. The cost is reported in US dollar based on the pricing in February 2024. The inference time is reported in minutes, and please be aware that this is just a rough estimation as the actual running time is also dependent on the server condition.

We include the following settings to compare with TELEClass:

- **GPT-3.5-turbo**: We include all the classes in the prompt and ask GPT-3.5-turbo model to provide 3 most appropriate classes for a given document.
- **GPT-3.5-turbo (level)**: We perform level-by-level prompting using GPT-3.5-turbo. Starting from the root node, we ask the

model to return one most appropriate class for a given document, and we iteratively prompt the model with the child nodes of the selected node at each level. Notice that this method can only generate a path in the taxonomy, but in the actual multi-label hierarchical classification setting, the true labels may not sit in the same path in the label taxonomy.

- **GPT-4**: Similar to the first one, we include all the classes in the prompt and ask GPT-4 to provide 3 most appropriate classes for a given document. Given the limited budget, we only test it on a sampled subset of the test split, which contains 1,000 documents for each dataset.

Table 3 shows the experiment results. We find that TELEClass consistently outperforms all compared methods on DBpedia, while on Amazon, TELEClass underperforms GPT-3.5-turbo (level) and GPT-4 but still being comparable. As for the cost, once trained, TELEClass does not require an additional cost on inference and also has substantially shorter inference time. Prompting LLMs takes longer time and can be prohibitively expensive (e.g., using GPT-4), and the cost will scale up with an increasing size of test data. Also, we find that the level-by-level version of GPT-3.5-turbo consistently outperforms the naive version, demonstrating the necessity of taxonomy structure. It saves the cost because of the much shorter prompts, but takes longer time due to more queries made per document.

4.5 Case studies

To better understand the TELEClass framework, we show some intermediate results of two documents in Table 4, including the core classes selected by (1) the original TaxoClass [31] method, (2) TELEClass’s initial core classes selected by LLM, and (3) refined core classes of TELEClass. Besides, we also include the true labels of the documents and the taxonomy enrichment results of the corresponding core class in the table. Overall, we can see that TELEClass’s refined core class is the most accurate. For example, for the first Wikipedia article about a library, TaxoClass selects “village” as the core class, while TELEClass’s initial core class finds

Table 4: Intermediate results on two documents, including selected core classes by different methods, true labels, and the corresponding taxonomy enrichment results. The optimal core class in the true labels is marked with ©.

| Dataset | Document | Core Classes by... | True Labels | Corr. Enrichment |
|---------|---|--|---|--|
| DBPedia | The Lindenhurst Memorial Library (LML) is located in Lindenhurst, New York, and is one of the fifty six libraries that are part of the Suffolk Cooperative Library System ... | TaxoClass: village TELEClass initial: building TELEClass refined: library | library©, agent, educational institution | Class: library Top Enrichment: national library, central library, collection, volumes... |
| Amazon | Since mom (89 yrs young) isn't steady on her feet, we have place these grab bars around the room. It gives her the stability and security she needs. | TaxoClass: personal care, health personal care, safety TELEClass initial: daily living aids, medical supplies equipment, safety, TELEClass refined: bathroom aids safety | health personal care, medical supplies equipment, bathroom aids safety© | Class: bathroom aids safety Top Enrichment: seat, toilet, shower, safety, handles... |

a closer one “building” thanks to the power of LLMs. Then, with the enriched classes-indicative features as guidance, TELEClass’s refined core class correctly identifies the optimal core class, which is “library”. In the other example, TELEClass also pinpoints the most accurate core class “bathroom aids safety” while other methods can only find more general or partially relevant classes.

5 RELATED WORK

5.1 Weakly-Supervised Text Classification

Weakly-supervised text classification trains a classifier with limited guidance, aiming to reduce human efforts while maintaining high proficiency. Various sources of weak supervision have been explored, including distant supervision [5, 11, 33] like knowledge bases, keywords [1, 22, 24, 36, 39, 46], and heuristic rules [2, 27, 32]. Later, extremely weakly-supervised methods are proposed to solely rely on class label names for generating pseudo-labels and training classifiers. LOTClass [24] utilizes MLM-based PLM as a knowledge base for extracting class-indicative keywords. XClass [39] extracts keywords for creating static class representations through clustering. PIEClass [49] employs PLMs’ zero-shot prompting to obtain pseudo labels with noise-robust iterative ensemble training. MEGClass [16] acquires contextualized sentence representations to capture topical information at the document level. WOTClass [38] focuses on mining and ranking class-indicative keywords from generated classes and extracting classes with overlapping keywords.

5.2 Hierarchical Text Classification

A hierarchy provides a systematic top-to-down structure with inherent semantic relations that can assist in text classification. Typical hierarchical text classification can be categorized into two groups: local approaches and global approaches. Local approaches train multiple classifiers for each node or local structures [3, 19, 41]. Global approaches, learn hierarchy structure into a single classifier through recursive regularization [12], a graph neural network (GNN)-based encoder [15, 26, 31, 51], or a joint document label embedding space [6].

Weak supervision is also studied for hierarchical text classification. WeSHClass [23] uses a few keywords or example documents per class and pretrains classifiers with pseudo documents followed by self-training. TaxoClass [31] follows the same setting as ours

which uses the sole class name of each class as the only supervision. It identifies core classes for each document using a textual entailment model, which is then used to train a multi-label classifier. Additionally, MATCH [50] and HiMeCat [48] study how to integrate associated metadata into the label hierarchy for document categorization with weak supervision.

5.3 LLMs as Generators and Annotators

Large language models (LLMs) have demonstrated impressive performance in many downstream tasks and are explored to help low-resource settings by synthesizing data as generators or annotators [8]. For data generation, few-shot examples [40] or class-conditioned prompts [21] are explored for LLM generation and the generated data can be used as pseudo-training data to further fine-tune a small model as the final classifier [42]. Recently, Yu et al. [45] propose an attribute-aware topical text classification method that incorporates ChatGPT to generate topic-dependent attributes and topic-independent attributes to reduce topic ambiguity and increase topic diversity for generation. For data annotation, previous works utilize LLMs for unsupervised annotation [10], Chain-of-Thought annotation with explanation generation [14], and active annotation [47].

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a new method, TELEClass, for the minimally-supervised hierarchical text classification task with two major contributions. First, we enrich the input label taxonomy with more class-indicative topical terms for each class, which can serve as additional features to understand the classes and facilitate classification. Second, we explore the utilization of LLMs in the hierarchical text classification in two directions: data annotation and data creation. On two public datasets, TELEClass can outperform existing baseline methods substantially, and we further demonstrate its effectiveness through ablation studies. We also conduct a comparative analysis on performance and cost for zero-shot LLM prompting for the hierarchical text classification task. For future works, we plan to generalize TELEClass’s idea into other low-resource text mining tasks with hierarchical label spaces, such as fine-grained entity typing. We also plan to explore how to extend TELEClass into a more complicated label space, such as an extremely large space with millions of labels.

REFERENCES

- [1] Eugene Agichtein and Luis Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *Digital library*.
- [2] Sonia Badene, Kate Thompson, Jean-Pierre Lorré, and Nicholas Asher. 2019. Data Programming for Learning Discourse Structure. In *ACL*.
- [3] Siddhartha Banerjee, Cem Akkaya, Francisco Perez-Sorrosal, and Kostas Tsoutsoulis. 2019. Hierarchical Transfer Learning for Multi-label Text Classification. In *ACL*.
- [4] David Berthelot, Nicholas Carlini, Ian Goodfellow, Avital Oliver, Nicolas Papernot, and Colin Raffel. 2019. MixMatch: a holistic approach to semi-supervised learning. In *NeurIPS*.
- [5] Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of Semantic Representation: Dataless Classification. In *AAAI*.
- [6] Haibin Chen, Qianli Ma, Zhenxi Lin, and Jiangyue Yan. 2021. Hierarchy-aware Label Semantics Matching Network for Hierarchical Text Classification. In *ACL-IJCNLP*.
- [7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.
- [8] Bosheng Ding, Chengwei Qin, Linlin Liu, Yew Ken Chia, Boyang Li, Shafiq Joty, and Lidong Bing. 2023. Is GPT-3 a Good Data Annotator?. In *ACL*.
- [9] Chengyu Dong, Zihan Wang, and Jingbo Shang. 2023. Debiasing Made State-of-the-art: Revisiting the Simple Seed-based Weak Supervision for Text Classification. In *EMNLP*.
- [10] Xiaohong Feng, Xiaocheng Feng, Libo Qin, Bing Qin, and Ting Liu. 2021. Language Model as an Annotator: Exploring DialoGPT for Dialogue Summarization. In *ACL-IJCNLP*.
- [11] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing Semantic Relatedness Using Wikipedia-Based Explicit Semantic Analysis. In *IJCAI*.
- [12] Siddharth Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *KDD*.
- [13] Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. 2019. Variational Pretraining for Semi-supervised Text Classification. In *ACL*.
- [14] Xingwei He, Zheng-Wen Lin, Yeyun Gong, Alex Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2023. AnnoLLM: Making Large Language Models to Be Better Crowdsourced Annotators. *ArXiv abs/2303.16854* (2023).
- [15] Wei Huang, Enhong Chen, Qi Liu, Yuying Chen, Zai Huang, Yang Liu, Zhou Zhao, Dan Zhang, and Shijin Wang. 2019. Hierarchical Multi-label Text Classification: An Attention-based Recurrent Network Approach. In *CIKM*.
- [16] Priyanka Kargupta, Tanay Komarlu, Susik Yoon, Xuan Wang, and Jiawei Han. 2023. MEGClass: Extremely Weakly Supervised Text Classification via Mutually-Enhancing Text Granularities. In *Findings of EMNLP*.
- [17] Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*.
- [18] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, S. Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6 (2015), 167–195.
- [19] Tieyan Liu, Yiming Yang, Hao Wan, Huajun Zeng, Zheng Chen, and Weiyang Ma. 2005. Support vector machines classification with a very large-scale taxonomy. In *KDD*.
- [20] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*.
- [21] Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. Generating Training Data with Language Models: Towards Zero-Shot Language Understanding. In *NeurIPS*.
- [22] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-Supervised Neural Text Classification. In *CIKM*.
- [23] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2019. Weakly-supervised hierarchical text classification. In *AAAI*.
- [24] Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. 2020. Text Classification Using Label Names Only: A Language Model Self-Training Approach. In *EMNLP*.
- [25] OpenAI. 2023. GPT-4 Technical Report. *ArXiv abs/2303.08774* (2023).
- [26] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Yangqiu Song, and Qiang Yang. 2018. Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN. In *WWW*.
- [27] Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data Programming: Creating Large Training Sets, Quickly. In *NIPS*.
- [28] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*.
- [29] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (apr 2009), 333–389.
- [30] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, and Jiawei Han. 2018. Automated Phrase Mining from Massive Text Corpora. *TKDE* 30, 10 (2018), 1825–1837.
- [31] Jiaming Shen, Wenda Qiu, Yu Meng, Jingbo Shang, Xiang Ren, and Jiawei Han. 2021. TaxoClass: Hierarchical Multi-Label Text Classification Using Only Class Names. In *NAACL*.
- [32] Kai Shu, Subhabrata Mukherjee, Guoqing Zheng, Ahmed Hassan Awadallah, Milad Shokouhi, and Susan Dumais. 2020. Learning with Weak Supervision for Email Intent Detection. In *SIGIR*.
- [33] Yangqiu Song and Dan Roth. 2014. On Dataless Hierarchical Text Classification. In *AAAI*.
- [34] T. Sørensen. 1948. *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons*. Munksgaard in Komm.
- [35] Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text Classification via Large Language Models. In *Findings of EMNLP*.
- [36] Fangbo Tao, Chao Zhang, Xiusi Chen, Meng Jiang, Tim Hanratty, Lance Kaplan, and Jiawei Han. 2018. Doc2Cube: Allocating Documents to Text Cube Without Labeled Data. In *ICDM*.
- [37] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shrutit Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *ArXiv abs/2307.09288* (2023).
- [38] Tianle Wang, Zihan Wang, Weitang Liu, and Jingbo Shang. 2023. WOT-Class: Weakly Supervised Open-world Text Classification. In *CIKM*.
- [39] Zihan Wang, Dheeraj Mekala, and Jingbo Shang. 2021. X-Class: Text Classification with Extremely Weak Supervision. In *NAACL*.
- [40] Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021. Towards Zero-Label Language Learning. *ArXiv abs/2109.09193* (2021).
- [41] Jonas Wehrmann, Ricardo Cerri, and Rodrigo C. Barros. 2018. Hierarchical Multi-label Classification Networks. In *ICML*.
- [42] Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. ZeroGen: Efficient Zero-shot Learning via Dataset Generation. In *EMNLP*.
- [43] Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach. In *EMNLP-IJCNLP*.
- [44] Ronghui You, Zihan Zhang, Ziyi Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. In *NeurIPS*.
- [45] Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023. Large Language Model as Attributed Training Data Generator: A Tale of Diversity and Bias. In *NeurIPS*.
- [46] Lu Zhang, Jiandong Ding, Yi Xu, Yingyao Liu, and Shuigeng Zhou. 2021. Weakly-supervised Text Classification Based on Keyword Graph. In *EMNLP*.
- [47] Ruoyu Zhang, Yanzeng Li, Yongliang Ma, Ming Zhou, and Lei Zou. 2023. LLMaAA: Making Large Language Models as Active Annotators. In *Findings of EMNLP*.
- [48] Yu Zhang, Xiusi Chen, Yu Meng, and Jiawei Han. 2021. Hierarchical Metadata-Aware Document Categorization under Weak Supervision. In *WSDM*.
- [49] Yunyi Zhang, Minhao Jiang, Yu Meng, Yu Zhang, and Jiawei Han. 2023. PIEClass: Weakly-Supervised Text Classification with Prompting and Noise-Robust Iterative Ensemble Training. In *EMNLP*.
- [50] Yu Zhang, Zhihong Shen, Yuxiao Dong, Kuansan Wang, and Jiawei Han. 2021. MATCH: Metadata-Aware Text Classification in A Large Hierarchy. In *WWW*.
- [51] Jie Zhou, Chunping Ma, Dingkun Long, Guangwei Xu, Ning Ding, Haoyu Zhang, Pengjun Xie, and Gongshe Liu. 2020. Hierarchy-Aware Global Model for Hierarchical Text Classification. In *ACL*.

A EVALUATION METRICS

Let $\mathbb{C}_i^{\text{true}}$ and $\mathbb{C}_i^{\text{pred}}$ denote the set of true labels and the set of predicted labels for document $d_i \in \mathcal{D}$, respectively. If a method can generate rankings of classes within $\mathbb{C}_i^{\text{pred}}$, we further denote its top- k predicted labels as $\mathbb{C}_{i,k}^{\text{pred}} \subset \mathbb{C}_i^{\text{pred}}$. The evaluation metrics are defined as follows:

- **Example-F1** [34], which is also called micro-Dice coefficient, evaluates the multi-label classification results without ranking,

$$\text{Example-F1} = \frac{1}{|\mathcal{D}|} \sum_{d_i \in \mathcal{D}} \frac{2 \cdot |\mathbb{C}_i^{\text{true}} \cap \mathbb{C}_i^{\text{pred}}|}{|\mathbb{C}_i^{\text{true}}| + |\mathbb{C}_i^{\text{pred}}|}. \quad (8)$$

- **Precision at k** , or $P@k$, is a ranking-based metric that evaluates the precision of top- k predicted classes,

$$P@k = \frac{1}{k} \sum_{d_i \in \mathcal{D}} \frac{|\mathbb{C}_i^{\text{true}} \cap \mathbb{C}_{i,k}^{\text{pred}}|}{\min(k, |\mathbb{C}_i^{\text{true}}|)}. \quad (9)$$

- **Mean Reciprocal Rank**, or MRR, is another ranking-based metric, which evaluates the multi-label predictions based on the inverse of true labels' ranks within predicted classes,

$$\text{MRR} = \frac{1}{|\mathcal{D}|} \sum_{d_i \in \mathcal{D}} \frac{1}{|\mathbb{C}_i^{\text{true}}|} \sum_{c_j \in \mathbb{C}_i^{\text{true}}} \frac{1}{\min\{k | c_j \in \mathbb{C}_{i,k}^{\text{pred}}\}}. \quad (10)$$

B PROMPTS FOR LLM

- **Core class annotation for Amazon-531**

Instruction: You will be provided with an Amazon product review, and your task is to select its product types from the following categories: [Candidate Classes]. Just give the category names as shown in the provided list.

Query: [Document]

- **Path-based generation for Amazon-531**

Instruction: Suppose you are an Amazon Reviewer, please generate 5 various and reliable passages following the requirements below:

1. Must generate reviews following the themes of the taxonomy path: [Path].
2. Must be in length about 100 words.
3. The writing style and format of the text should be a product review.
4. Should keep the generated text to be diverse, specific, and consistent with the given taxonomy path. You should focus on [The Leaf Node on the Path].

- **Core class annotation for DBpedia-298**

Instruction: You will be provided with a Wikipedia article describing an entity at the beginning, and your task is to select its types from the following categories: [Candidate Classes]. Just give the category names as shown in the provided list.

Query: [Document]

- **Path-based generation for DBpedia-298**

Instruction: Suppose you are a Wikipedia Contributor, please generate 5 various and reliable passages following the requirements below:

1. Must generate reviews following the themes of the taxonomy path: [Path].
2. Must be in length about 100 words.
3. The writing style and format of the text should be a Wikipedia page.
4. Should keep the generated text to be diverse, specific, and consistent with the given taxonomy path. You should focus on [The Leaf Node on the Path].