Single-pass Hierarchical Text Classification with Large Language Models

Fabian Schmidt*, Karin Hammerfald†, Henrik Haaland Jahren‡
Amir H. Payberah*, and Vladimir Vlassov*

* KTH Royal Institute of Technology, Sweden
† University of Oslo, Norway
‡Braive AS, Norway
Email: *{fschm, payberah, vladv}@kth.se, † karin.hammerfald@psykologi.uio.no,
‡henrik@braive.com

***Abstract*—Numerous text classification tasks inherently possess hierarchical structures among classes, often overlooked in traditional classification paradigms. This study introduces novel approaches for hierarchical text classification using Large Language Models (LLMs), exploiting taxonomies to improve accuracy and traceability in a zero-shot setting. We propose two hierarchical classification methods, namely (i) *single-path* and (ii) *path-traversal*, which all leverage the hierarchical class structures inherent in the target classes (e.g., a bird is a type of animal that belongs to a species) and improve naïve hierarchical text classification from literature. We implement them as prompts for generative models such as OpenAI GPTs and benchmark them against discriminative language models (BERT and RoBERTa). We measure the classification performance (precision, recall, and F1-score) vs. computational efficiency (time and cost). Throughout the evaluations of the classification methods on two diverse datasets, namely ComFaSyn, containing mental health patients' diary entries, and DBpedia, containing structured information extracted from Wikipedia, we observed that our methods, without any form of fine-tuning and few-shot examples, achieve comparable results to flat classification and existing methods from literature with minimal increases in the prompts and processing time.**

***Index Terms*—Hierarchical text classification, Large Language Models (LLMs), zero-shot classification**

I. Introduction

Text classification is essential in natural language processing, enabling efficient organization of unstructured data and enhancing information retrieval. As digital content grows, automated classification becomes vital for optimizing workflows in healthcare, finance, and customer service, where timely and accurate categorization is crucial.

Conventional text classification often ignores the hierarchical structures present in many datasets. Our work is motivated by the fact that many classification tasks rely on taxonomies, where each level provides greater specificity. Hierarchical text classification (HTC) produces a structured set of labels from different taxonomy levels, offering a more granular understanding of the text's content. These labels capture hierarchical relationships, classifying the text at increasing levels of detail. For example, the model could first categorize a city as a *Place*, then as a *NaturalPlace*, and finally as a *City*.

This work is funded by the Research Council of Norway (grant 321561) and was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation.

We explore text classification methods that leverage the hierarchical nature of class structures, building on existing research on HTC using LLMs by [1]. HTC becomes highly relevant to multiple inheritance taxonomies, where it is not always clear which parent a class belongs to, requiring sophisticated methods to disambiguate relationships and optimize classification performance. In addition to the existing approach, (i) *naïve hierarchical* [1], where a LLM predicts classes at each level and narrows down the search space iteratively, we introduce two methods: (ii) *single-path*, where the LLM generates entire paths through the taxonomy in a single response, and (iii) *path-traversal*, which models the entire hierarchical exploration in a single response. These methods facilitate accurate classification, maintain interpretability by providing insights into the hierarchical reasoning process, and improve computational efficiency.

We demonstrate the effectiveness of our methods by extensively testing them over two diverse datasets, namely (i) ComFaSyn, which contains synthetic mental health patients' diary entries, and (ii) DBpedia, which contains structured information extracted from Wikipedia. We benchmark the models with two types of text classification models: discriminative (e.g., BERT and RoBERTa) and generative language models (e.g., GPT) and compare them to state-of-the-art methods.

Our empirical findings reveal competitive performance compared to state-of-the-art approaches, such as SBERT embeddings with SVM and BERT, particularly on datasets like DBpedia and ComFaSyn. On the ComFaSyn dataset, the flat classification achieved a perfect precision score of 1.0 and an F1 score of 0.9815, while the single-path model achieved a precision of 0.9762 and an F1 score of 0.9653. The path-traversal model performed similarly, with a precision of 0.9722 and an F1 score of 0.9564. On DBpedia, the BERT model achieved the highest accuracy at 0.993, whereas our single-path method reached 0.967 without fine-tuning or few-shot examples. Furthermore, single-path and path-traversal require approximately one-third of the computations of naïve hierarchical methods.

To summarize, this paper makes the following contributions.

- The creation of the taxonomy of common factors, which are the effective elements of treatment shared among various types of psychotherapy, and the ComFaSyn training dataset of synthetic mental health patient diary entries, designed explicitly for evaluating hierarchical classification methods.
- The definition and implementation of novel approaches for HTC using LLMs.
- A comprehensive empirical evaluation of these proposed hierarchical methods to assess their effectiveness.

II. Background and Related Work

Text classification is the task of categorizing a textual input into pre-defined classes [2]. Examples include sentiment analysis [3], spam detection [4], topic labeling [5], and toxicity detection [6]. *Hierarchical* text classification [7]–[9], contrarily to *flat* classification, utilizes the inherent hierarchical structure of the classes. In its simplest form, HTC is repeated flat classification. HTC is especially advantageous when processing a large class space [9] because the classification tasks can be decomposed into multiple smaller problems.

Traditionally, discriminative models such as BERT [10] and RoBERTa [11] have shown strong performance in text classification tasks [12], [13], but they typically require large amounts of training data. In contrast, LLM have demonstrated impressive few-shot learning capabilities across various tasks [14]. With few-shot learning, LLMs can perform tasks with minimal examples in the input prompts, showcasing their ability to understand task structures and logic [15]. This makes LLMs a promising approach for text classification, especially when combined with advanced prompting techniques. One such prompting technique is Chain-of-Thought (CoT) [16], which improves reasoning by breaking tasks down into intermediate steps. Tree-of-Thoughts (ToT) [17] extends this concept by introducing a branching reasoning process that explores multiple reasoning paths. This structured reasoning approach aligns well with hierarchical classification and is part of our core method.

Additionally, LLMs have been used for data augmentation in HTC [18], [19]. [19] build a pipeline to enrich the taxonomy and classify documents using a matching network [20]. More recently, [1] performed HTC with few-shot learning by retrieving the most relevant demonstrations using similarity search. The retrieval model fetches demonstrations iteratively from a retrieval database at each layer, reducing candidate label sets layer by layer. Our approach differs as we classify the whole path along the taxonomy in a single pass.

III. Problem Formulation

Text classification aims to classify the textual input x into corresponding class label $y_i \in Y$, where $Y = \{y_1, y_2, \dots, y_n\}$. In this study, we leverage LLM, denoted as $\mathcal{M}$, as text classifiers that map input text to a distinct class label, $\mathcal{M} : x \to y_i$.

HTC leverages a hierarchy in the classes, also known as taxonomies. We consider taxonomies with single inheritance, such as the taxonomy in Figure 1 consisting of a single root and two main branches: *Place* and *Animal*, representing the two primary categories. The structure forms a tree-like diagram, with each node representing a class or category and arrows indicating *IS_A* relationships between them.

Formally, we can define the taxonomy in the following way. Let $\mathcal{V} = \{r, y_{1,1}, y_{1,2}, \dots, y_{1,m_1}, \dots, y_{n,1}, y_{n,2}, \dots y_{n,m_n}\}$ represent a hierarchy of class labels descending from the root r, where $n \geq 2$ indicates the number of levels in the taxonomy, and m_i denotes the maximum number of classes at that level. Then, a class $y_{i,j}$ signifies the class j at level i. Furthermore, let $V^{(i)}$ be the classes at the i-th level, e.g., in Figure 1, $V^{(1)} = \{Place, Animal\}$. We formulate the taxonomy as a directed acyclic graph $G = (\mathcal{V}, \mathcal{E}, r)$, where

- $\mathcal{V} = \{r, y_{1,1}, y_{1,2}, \dots, y_{1,m_1}, \dots, y_{n,1}, y_{n,2}, \dots, y_{n,m_n}\}$ is the set of vertices representing the classes;
- $\mathcal{E} \subseteq \{(u, v) \in \mathcal{V} \times \mathcal{V} \wedge u \neq v\}$ is a set of edges where (u, v) indicates that u is the superclass of v;
- $r \in \mathcal{V}$ is the root vertex from which all classes descend. The root has no superclass, hence no incoming edges.

Introducing the taxonomy shifts the classification objective by decomposing the classification into n intermediate classifications. We modify the classifier function as $\mathcal{M} : x \to v$, where v is a subset of $\mathcal{V}$, such that $v = (y_{1,a}, y_{2,b}, \dots, y_{n,z})$ is a path from the first level class $y_{1,a}$ to the leaf class $y_{n,z}$ in the taxonomy.

IV. Flat and Hierarchical Classification Methods

In this section, we present our three hierarchical classification methods, (i) *hierarchical*, (ii) *single-path*, and (iii) *hierarchical one-prompt*, to traverse a taxonomy and classify the input using any conversational LLM. Each method approaches the HTC task with subtle variations, though they all build on the same underlying principle. Rather than *flat* classification, these methods decompose the task into subtasks aligned with the levels in the hierarchy, aiming to predict a sequence of class labels $\hat{v} = (y_{1,a}, y_{2,b}, \dots, y_{n,z})$ that represents a path from the root to a leaf node within the taxonomy.

Algorithm 1 details the steps in a naïve hierarchical classification. The hierarchical process necessitates an input x, an LLM $\mathcal{M}$, the number of levels n, the classes $\mathcal{V}$, the demonstrations $\mathcal{D}$. A demonstration $d \in \mathcal{D}$ is an example of an input text for the candidate classes to guide the model in the classification process. For instance, at the first level ($i = 1$), the candidate classes comprise $\{Place, Animal\}$, and a sample in d_1 might be *"Geese are migrating for hundreds of miles."* which belongs to the class *Animal*. The LLM $\mathcal{M}$ classifies the input data at the first level, given the candidate classes and demonstrations, producing the predictions $\hat{z}_i$ and optionally an explanation that explains why the model predicted z_i. A scoring function $\phi(\cdot) \to \mathcal{R} \in [0, 1]$ takes z_i as input to quantify confidence in each class at that level producing the likeliest class $\hat{y}_i$. There are several techniques to score the predictions, including uncertainty estimation [21]. We prompt $\mathcal{M}$ to elicit confidence similar to [22], but seamlessly integrating other techniques is also possible. This process is repeated for the n levels. In each iteration, the predictions from the previous level are evaluated and used to refine the candidate classes for the next level by pruning unlikely classes based on the scores of the predictions.

Algorithm 1 Naïve hierarchical classification

```
Input: input $x$, LLM $\mathcal{M}$, levels $n$, classes $\mathcal{V}$,
    demonstrations $\mathcal{D}$
Output: $\hat{v}$ the predicted sequence of classes
    $(y_{1,a}, y_{2,b}, \dots, y_{n,z})$
1: Let $\hat{v}$ be the empty sequence        ▷ Initialize the sequence
2: for $i \leftarrow 1$ to $n$ do
3:     $d_i \leftarrow$ SAMPLE$(\mathcal{D}, \mathcal{V}^{(i)}, k)$ ▷ Sample $k$ demonstrations
4:     $\hat{z}_i \leftarrow \mathcal{M}(x, \mathcal{V}^{(i)}, d_i)$            ▷ Classify input
5:     $\hat{y}_i \leftarrow argmax_{\hat{z}_i \in V} \phi(\hat{z}_i)$  ▷ Get the top-scoring class
6:     $\mathcal{V} \leftarrow$ PRUNING$(x, \hat{z}_i)$   ▷ Prune the irrelevant classes
7:     Append $\hat{y}_i$ to $\hat{v}$               ▷ Add to class sequence
8: end for
9: return $\hat{v}$                  ▷ Final classification at level $n$
```

Figure 1 (top) shows an example of a taxonomy with two levels represented as a graph. There are two classes at the first level, *Place* and *Animal*, and five subclasses at the second level. For instance, *Building* is a subclass of the *Place* superclass. We assume that each class has only one superclass and each input belongs to exactly one class. Ultimately, the aim is to classify input into leaf classes at the last level. For example, given an input *"The Grand Central Terminal is located in Midtown Manhattan."*, the flat classification model $\mathcal{M}$ should return *Station*. The hierarchical classification methods should result in a fully qualified class name that contains all classes on the path from the root to the leaf class in the taxonomy, e.g., *(Place, Station)*. With multi-class classification, the most likely option, *Station* out of the possible options, is returned.

HTC methods offer several advantages. They enhance traceability for debugging, allowing effective error analysis throughout the classification process. By decomposing the problem into manageable sub-problems, these methods align with the CoT and ToT paradigms [16], [17]. Additionally, they allow for assessing prediction and explanation quality at each stage, providing a comprehensive performance evaluation. Unlike naïve multi-step approaches, our HC paradigm runs in a single pass—using one prompt and response—saving computational resources while maintaining the benefits of hierarchical classification. Below, we highlight the differences between flat and our three HTC methods: hierarchical, single-path, and path-traversal.

a) Flat classification.: Traditional multi-class text classification, commonly referred to as flat classification (as depicted in Figure 1a), is the prevalent method for categorizing textual content [23]. In this approach, the classifier model $\mathcal{M}$ receives an input text x and generates a prediction $\hat{y}$, representing, for instance, the sentiment of a paragraph or the topical category of a news article. Effectively, flat classification disregards any information on the class topology, i.e., *Building is a Place and not an Animal*, and directly classifies the leaf class y_n. In this study, flat classification is one of the baselines in the experimental analysis.

b) Naïve hierarchical classification.: To leverage the information inherent in the taxonomy, we consider a hierarchical approach, also used by [1], to classify the input. Algorithm 1 and Figure 1b) illustrate the approach. This method is the naive implementation of Algorithm 1. At each level, we prompt the model to generate the top three classes in JSON format, similar to a beam search, and prune irrelevant classes and sub-classes before moving to the next level.

c) Single-path classification.: The hierarchical approach queries the LLM at each layer. To improve efficiency, we introduce single-pass prompting techniques. Aligned with the CoT approach [16], our single-path classification leverages a step-by-step reasoning paradigm, deconstructing complex problems into manageable intermediate steps. This strategy has been further adapted by [15] for text classification, wherein tasks such as sentiment classification are dissected into a series of incremental steps. A single-path is the sequence of classes $(y_{1,a}, y_{j,b}, \dots, y_{n,z})$, a path through the taxonomy from root to leaf classes. We prompt the model to generate this path in one response, significantly reducing the total token length while preserving the step-by-step thinking process. For instance, the model receives the input *"The Grand Central Terminal is located in Midtown Manhattan."* and should generate the sequence *(Place, Building)*.

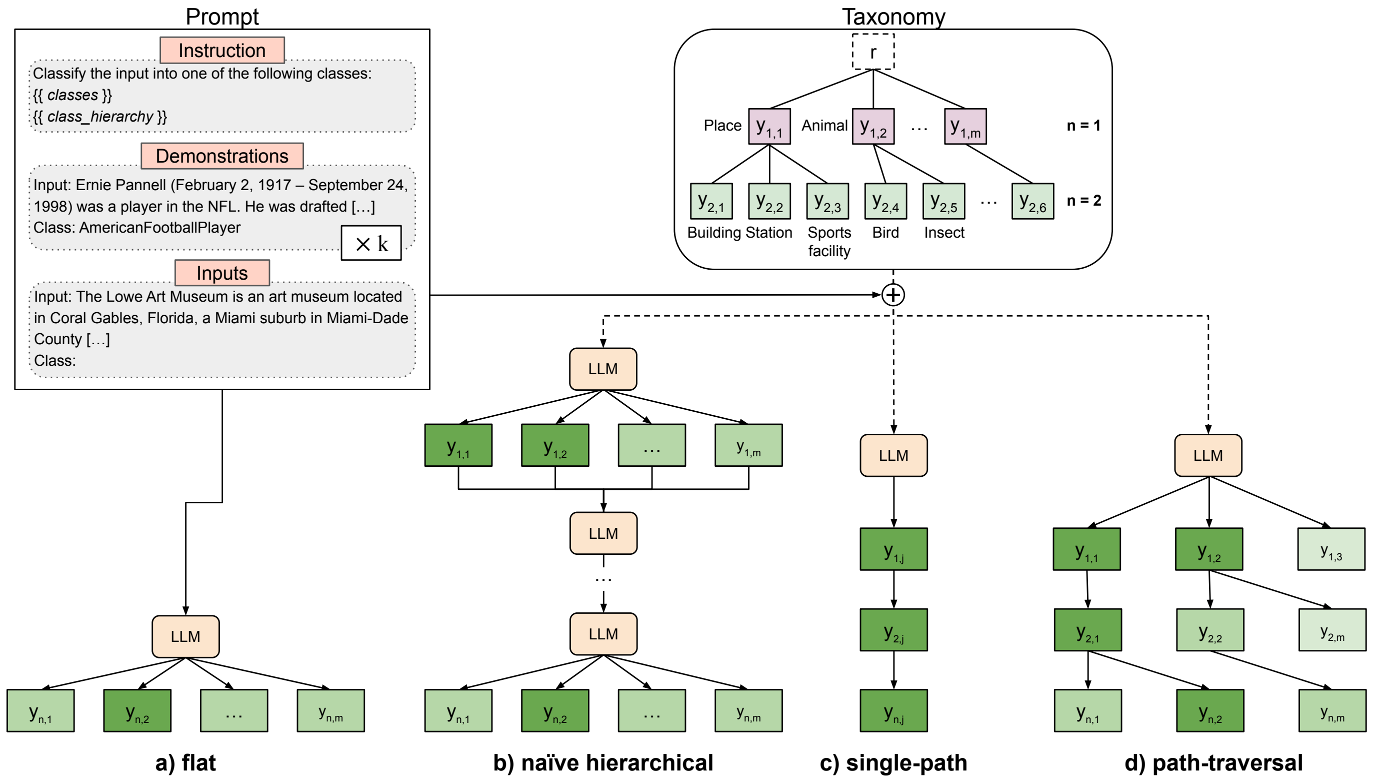
d) Path-traversal classification.: The path-traversal method aligns with the ToT approach [17] and follows the same principle as the hierarchical method. The aim is to explore multiple paths down the taxonomy. However, instead of having a multi-turn conversation by querying the model n times, i.e., the number of levels, the LLM is queried just once. We prompt the LLM to generate p paths through the taxonomy and assign confidence scores to each path, simulating the hierarchical exploration process in the response. For instance, given the input *"The Grand Central Terminal is located in Midtown Manhattan."*, the model explores the paths (Place, Building), (Place, Station), and (Place, Sports facility). Assuming the scoring function $\phi(\cdot)$ ranks these paths in that order, the final prediction will be $\hat{y} =$ *(Place, Building)*.

V. Experimental Setup

We evaluate the hierarchical classification methods in zero-shot and fine-tuned scenarios on ComFaSyn and DBpedia, using flat, naïve hierarchical and discriminative models as baselines. Performance is measured by accuracy and time efficiency. The following section details the experimental setup, including models, datasets, metrics, and results. Hardware configuration is in Appendix C.

A. Datasets

We used two datasets for multi-class classification: ComFaSyn, a dataset of mental health patient diary entries, and DBpedia, which contains structured information extracted from



Prompt
Instruction
Classify the input into one of the following classes:
{{ classes }}
{{ class_hierarchy }}
Demonstrations
Input: Ernie Pannell (February 2, 1917 – September 24, 1998) was a player in the NFL. He was drafted […]
Class: AmericanFootballPlayer
× k
Inputs
Input: The Lowe Art Museum is an art museum located in Coral Gables, Florida, a Miami suburb in Miami-Dade County […]
Class:
Taxonomy
r
Place
Animal
n = 1
n = 2
Building
Station
Sports facility
Bird
Insect
LLM
a) flat
b) naïve hierarchical
c) single-path
d) path-traversal

Fig. 1: Flat and Hierarchical Classification Methods

Wikipedia. ComFaSyn comprises two levels, and DBpedia has three levels of classes. ComFaSyn and DBpedia taxonomies are presented in Appendix A and Appendix B, respectively. We distinguish between training and test data. The training data is used for (i) in-context learning demonstrations and (ii) fine-tuning an LLM, while the test data is for evaluation. The differences are detailed below.

a) Common-factors-synthetic (ComFaSyn): We developed a taxonomy of common factors in psychotherapy and created the ComFaSyn training dataset, which contains mental health patient diary entries. Common factors represent the effective elements shared across various forms of psychotherapy [24] and can be organized hierarchically, as broad concepts like therapeutic alliance include specific elements such as collaboration. Unlike the general knowledge domain of DBpedia, ComFaSyn focuses on a specialized health domain, making it a valuable dataset for testing classification methods in real-world, domain-specific contexts.

To create the dataset, we manually annotated five one-sentence examples per class from literature sources. This resulted in 65 samples across seven first-level and 12 second-level classes, with an average of 5.42 samples per second-level class. A synthetic test dataset was generated, containing five diary entries per class (60 total), where the model produces text reflecting on a user's therapy experience focused on a specific common factor. The prompt dynamically inserts factor-specific details to ensure personalized, contextually relevant entries.

The entries were generated using the *gpt-3.5-turbo* model via the OpenAI API, with a temperature of 1.2, max tokens of 1024, and a frequency penalty of 0.1. Further details about the taxonomy are in Appendix A.

b) DBpedia: We use a subset of DBpedia [25], a vast knowledge base extracted from Wikipedia. We randomly select ten samples from each first-level class of the validation set for evaluation purposes and sample the corresponding classes from the test set to serve as examples. By leveraging this diverse repository of structured information, we aim to evaluate the performance of our proposed methodology across a broad spectrum of domains and topics. This selection strategy ensures the inclusion of varied entities, relationships, and attributes, thereby enhancing the robustness and generalizability of our findings. The training dataset consists of 440 rows, with an average of 10 rows per class, spanning nine first-level classes and 33 second-level classes, for a total of 44 third-level classes. The test dataset contains a total of 90 samples distributed across 44 third-level classes (9 first-level and 33 second-level classes), with an average of 2.05 samples per class. Details on the DBpedia taxonomy are in Appendix B.

B. Discriminative Language Models

BERT [10] and RoBERTa [11] are transformer-based discriminative language models designed for various natural language processing tasks. BERT pre-trains deep bidirectional representations by jointly conditioning on both left and right context in all layers, making it effective for tasks like sentence classification and token-level predictions. RoBERTa builds on

BERT by training with larger batches and longer sequences and removing the next sentence prediction task to further improve performance. Both models are widely adopted as benchmarks in text classification due to their strong ability to capture contextual information and consistent performance across various classification tasks.

C. Generative Large Language Models

We create prompt templates for the flat and hierarchical classification methods (Algorithm 1 and Figure 1). The prompt includes the instruction, examples, and input to classify. The instruction describes the classification task and lists the possible classes and, in the case of hierarchical classification, their hierarchy in the taxonomy. Examples include the input, corresponding class, and, optionally, an explanation of why that input belongs to that class. Each example is separated by "###" to distinguish between individual instances. Lastly, the inputs to classify are procured for the model, either individually or collectively. We predominantly use the latter approach to optimize efficiency.

To constrict the output generation, we employ a JSON schema. This is essential because generative LLMs are non-deterministic by design. Through experiments with several LLMs, we encountered different degrees of faithfulness to the desired output format when only specified via the prompt. The extraction of the classes is crucial in a task such as classification. The JSON schema enforces consistent output in the desired format, including the predicted class and an explanation. The detailed prompt templates and JSON schema can be found in Appendix D.

D. Metrics

We assess both the classification performance and execution efficiency. For classification performance, we evaluate precision, recall, and F1-score. Efficiency metrics include execution time, token length per sample, and API costs ($). This evaluation explores the trade-off between model size, prompt length, query volume, and associated time/costs, which is particularly important in industrial contexts where cost-saving measures are crucial.

VI. Results

The results in Table I highlight the performance of various classification methods on the ComFaSyn and DBpedia datasets, evaluated using precision, recall, and F1 score. On the DBpedia dataset, BERT models perform exceptionally well, with BERT-large-cased achieving an F1 score of 0.9810 and BERT-base-cased close behind at 0.9746. RoBERTa models, however, show a slightly lower performance than BERT, with RoBERTa-large achieving an F1 score of 0.9788 and RoBERTa-base scoring 0.9697. On the ComFaSyn dataset, both BERT models and RoBERTa models struggle, with BERT-large-cased barely achieving an F1 score of 0.0152 and RoBERTa-large scoring 0.3783. Flat classification shows the strongest performance overall, with F1 scores of 0.9815 on ComFaSyn and 0.9696 on DBpedia. The single-path and path-traversal methods also demonstrate competitive results, particularly on DBpedia, where the path-traversal method achieves an F1 score of 0.9592. On ComFaSyn, the single-path method performs slightly below flat classification, with an F1 score of 0.9653. The naïve hierarchical method shows weaker performance, especially on DBpedia, with an F1 score of only 0.4513.

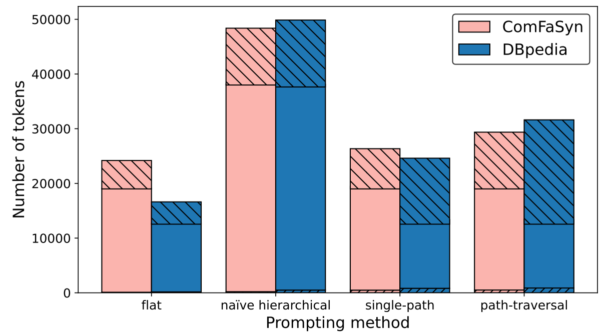


Fig. 2: Token lengths for methods (plain: raw input, hatched: response (top) and instruction (bottom))

Table II compares the results with existing literature. Flat classification achieved an accuracy of 0.978, while the single-path and path-traversal models both reached 0.967. SBERT with SVM attained 0.987, and the char-CNN and BERT models reported by [13] achieved 0.983 and 0.993, respectively. Importantly, these methods were trained on the full DBpedia dataset (560K samples), making our results highly competitive based on a smaller dataset.

VII. Discussion

We focus the discussion on two key findings: the performance of hierarchical classification methods, highlighting their pros and cons, and the efficiency trade-offs between time, cost, and performance.

A. Evaluation of Hierarchical vs. Flat Classification Methods

Hierarchical classification methods perform comparably to flat approaches but offer greater interpretability. Their hierarchical structure breaks down the classification task, providing finer granularity and more precise explanations for misclassifications. In contrast, flat classification lacks this depth, offering fewer insights into the causes of errors. Naïve hierarchical classification is significantly more expensive than our single-path and path-traversal methods, as it requires n queries. Its performance is also worse, as errors at earlier layers are irrecoverable due to the pruning unless the correct classification is within the top-3 predictions. This approach increases the chance of mistakes at each level, and due to pruning, there is no opportunity to recover. For instance, the F1-scores dropped from 0.9541 at the first level to 0.6574 at the second and down to 0.4513 at the third level in DBpedia.

A significant challenge in earlier classification methods was the accurate extraction of relevant classes, which was

TABLE I: Classification performance on ComFaSyn and DBpedia

Model	**ComFaSyn**			**DBpedia**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Discriminative Models						
bert-base-cased	0	0	0	0.9772	0.9750	0.9746
bert-large-cased	0.0139	0.0167	0.0152	0.9866	0.9818	0.9810
roberta-base	0.3625	0.3333	0.2718	0.9756	0.9705	0.9697
roberta-large	0.4906	0.4333	0.3783	0.9842	0.9795	0.9788
Generative Models (GPT-4o)						
flat	1.0	0.9667	0.9815	0.9648	0.9777	0.9696
naïve hierarchical	0.9630	0.8833	0.8905	0.5204	0.4444	0.4513
single-path	0.9762	0.9667	0.9653	0.9481	0.9667	0.9548
path-traversal	0.9722	0.9500	0.9564	0.9611	0.9667	0.9592
Fine-tuned Generative Models (GPT-3.5)						
respective dataset						
flat	$0.9584_{.008}$	$0.9278_{.010}$	$0.9150_{.010}$	$0.9197_{.008}$	$0.9296_{.006}$	$0.9164_{.002}$
single-path	$0.9576_{.010}$	$0.9111_{.010}$	$0.9055_{.010}$	$0.8964_{.013}$	$0.8926_{.006}$	$0.8858_{.009}$
path-traversal	$0.8853_{.060}$	$0.8667_{.050}$	$0.8537_{.065}$	$0.8686_{.003}$	$0.8629_{.032}$	$0.8546_{.023}$
both datasets						
flat	$0.9504_{.012}$	$0.9111_{.019}$	$0.9003_{.018}$	$0.9392_{.019}$	$0.9481_{.017}$	$0.9396_{.018}$
single-path	$0.9537_{.008}$	$0.9222_{.010}$	$0.9094_{.010}$	$0.9227_{.014}$	$0.9445_{.019}$	$0.9295_{.019}$
path-traversal	$0.9456_{.006}$	$0.9167_{.000}$	$0.9060_{.004}$	$0.9140_{.020}$	$0.9407_{.013}$	$0.9234_{.017}$

TABLE II: Comparison with related work. Results marked with * were obtained by [13]

Model	**Accuracy**
flat	0.978
single-path	0.967
path-traversal	0.967
Gated CNN [26]	0.950
SBERT + SVM [27]	0.987
char-CNN* [28]	0.983
BERT* [10]	0.993

particularly problematic before the availability of structured outputs. When fine-tuned for hierarchical classification or zero-shot, generative models have greatly benefited from the use of structured outputs, as these ensure that the classification is aligned with the underlying taxonomy.

The path-traversal method resulted in a marginal improvement over the single-path technique. While both approaches performed similarly overall, the path-traversal method corrected misclassified instances more effectively by considering all classification options and weighing them carefully before making a final prediction. This method of selecting a final prediction based on a holistic evaluation of multiple possible classes proved beneficial in scenarios where other methods struggled.

Flat classification methods occasionally skipped classifying specific samples, which led to what would have been significant performance decreases. This issue was mitigated only when the models were prompted to generate the input alongside the classification and explanation. Hierarchical classification methods were less prone to this issue, as their structure inherently reduced the likelihood of skipping classifications.

The quality of the taxonomy used for classification is another critical factor. Many misclassifications stem from ambiguities within the taxonomy. A clear, well-structured taxonomy is essential for accurate predictions, even for advanced models.

Finally, discriminative models performed well on general classification tasks such as DBpedia, where sufficient data was available for fine-tuning. However, these models struggled on specialized datasets like ComFaSyn, where data was more limited. In such cases, generative models proved more effective, particularly those fine-tuned with structured outputs. This suggests that generative models may be preferable in scenarios with limited data availability due to their ability to generate more contextually appropriate predictions with less reliance on large amounts of labeled data.

B. *Efficiency trade-off*

While hierarchical classification methods offer several advantages, they also introduce efficiency trade-offs. A naïve implementation of hierarchical classification is often inefficient because it requires the model to be queried at every level of the hierarchy. This increases the number of model queries, making the process computationally expensive. To address this, we introduced modified single-pass prompts, which mitigate the inefficiency by querying the model only once compared to n queries in the naïve hierarchical method. The single single-pass prompts approach maintains many advantages of hierarchical classification, such as interpretability, without incurring the overhead of multiple queries.

However, despite the efficiency improvements, hierarchical classification methods still result in longer token lengths for both input and output (Figure 2). This is because the hierarchical structure requires the model to process additional layers of information for each input, leading to increased token counts. The higher token count directly translates into greater computational cost, as most LLMs calculate processing costs based on the number of tokens. This makes hierarchical methods potentially more expensive than flat classification, mainly when operating at scale.

Another fundamental trade-off is the cost of building a robust taxonomy, which requires time and domain expertise. We argue that this investment is worthwhile in critical fields like healthcare, finance, and law. A detailed taxonomy offers a clear view of the classification structure, helps identify gaps, and enables more informed and traceable classifications, ultimately enhancing trust and transparency.

Ultimately, while hierarchical methods bring about additional costs regarding token usage and taxonomy construction, these costs are justified in domains where accuracy, interpretability, and accountability are paramount. In such cases, the ability to offer more detailed insights into the classification process and reduce ambiguities far outweighs the added computational and developmental effort.

VIII. Conclusion

We have explored the potential of leveraging LLMs for HTC, introducing a novel approach that capitalizes on the structure of taxonomies. We implemented three algorithmic variations—hierarchical, single-path, and path-traversal—to demonstrate the effectiveness of breaking down complex classification tasks into smaller, more manageable subtasks that align with the hierarchical structure of class categories.

Hierarchical classification methods offer improved interpretability and more detailed insights into misclassifications but come with efficiency trade-offs, including longer token lengths and higher computational costs. Modified single-pass prompts help mitigate inefficiencies, though building a robust taxonomy remains resource-intensive. However, in critical domains like healthcare and finance, investing in a well-structured taxonomy is worthwhile for ensuring accurate, transparent, and trustworthy classifications.

IX. Limitations and Future Work

Our approach relies on OpenAI's GPT because it produces structured outputs, which is critical for our methodology. Other models, such as LLaMA3, were explored but ultimately abandoned due to inconsistent output formatting, limiting the flexibility of our model choices. This restriction to a single model could limit generalizability and applicability to other model architectures.

Regarding scalability, while our method does not require a large sample size for training, it is constrained by the relatively small subset of the DBpedia dataset we used. This small sample size allowed us to focus on precision. Still, our approach's scalability, particularly in inference throughput, remains limited, making it less feasible for large-scale applications.

Additionally, the ComFaSyn dataset introduces potential biases. The training examples, derived from literature, were selected by a single expert, which may introduce bias in data selection. Moreover, the synthetic test dataset generated by the language model could carry biases inherent in the model itself, potentially skewing the evaluation results. Future work should address these limitations by expanding the dataset, introducing more rigorous bias mitigation techniques, and incorporating a dynamic parameter to determine the number of paths to explore at each layer.

References

[1] H. Chen, Y. Zhao, Z. Chen, M. Wang, L. Li, M. Zhang, and M. Zhang, "Retrieval-style In-Context Learning for Few-shot Hierarchical Text Classification," Jun. 2024, arXiv:2406.17534 [cs]. [Online]. Available: http://arxiv.org/abs/2406.17534

[2] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He, "A Survey on Text Classification: From Traditional to Deep Learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 13, no. 2, pp. 31:1–31:41, Apr. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3495162

[3] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018. [Online]. Available: https://doi.org/10.1002/widm.1253

[4] P. Teja Nallamothu and M. Shais Khan, "Machine Learning for SPAM Detection," *Asian Journal of Advances in Research*, vol. 6, no. 1, pp. 167–179, Mar. 2023, number: 1. [Online]. Available: http://eprint.subtopublish.com/id/eprint/3333/

[5] A. B. Dieng, F. J. R. Ruiz, and D. M. Blei, "Topic Modeling in Embedding Spaces," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 439–453, Jul. 2020. [Online]. Available: https://doi.org/10.1162/tacl_a_00325

[6] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Predicting the Type and Target of Offensive Posts in Social Media," Apr. 2019, arXiv:1902.09666 [cs]. [Online]. Available: http://arxiv.org/abs/1902.09666

[7] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1, pp. 31–72, Jan. 2011. [Online]. Available: https://doi.org/10.1007/s10618-010-0175-9

[8] S. Dumais and H. Chen, "Hierarchical classification of Web content," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '00. New York, NY, USA: Association for Computing Machinery, Jul. 2000, pp. 256–263. [Online]. Available: https://dl.acm.org/doi/10.1145/345508.345593

[9] Aixin Sun and Ee-Peng Lim, "Hierarchical text classification and evaluation," in *Proceedings 2001 IEEE International Conference on Data Mining*. San Jose, CA, USA: IEEE Comput. Soc, 2001, pp. 521–528. [Online]. Available: http://ieeexplore.ieee.org/document/989560/

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[11] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[12] D. Stammbach and E. Ash, "DocSCAN: Unsupervised Text Classification via Learning from Neighbors," Oct. 2022, arXiv:2105.04024 [cs]. [Online]. Available: http://arxiv.org/abs/2105.04024

[13] Y. Meng, Y. Zhang, J. Huang, C. Xiong, H. Ji, C. Zhang, and J. Han, "Text classification using label names only: A language model self-training approach," 2020. [Online]. Available: https://arxiv.org/abs/2010.07245

[14] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish,

A. Radford, I. Sutskever, and D. Amodei, “Language Models are Few-Shot Learners,” Jul. 2020, arXiv:2005.14165 [cs]. [Online]. Available: http://arxiv.org/abs/2005.14165

[15] X. Sun, X. Li, J. Li, F. Wu, S. Guo, T. Zhang, and G. Wang, “Text Classification via Large Language Models,” Oct. 2023, arXiv:2305.08377 [cs]. [Online]. Available: http://arxiv.org/abs/2305.08377

[16] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,” Jan. 2023, arXiv:2201.11903 [cs]. [Online]. Available: http://arxiv.org/abs/2201.11903

[17] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, “Tree of Thoughts: Deliberate Problem Solving with Large Language Models,” May 2023, arXiv:2305.10601 [cs]. [Online]. Available: http://arxiv.org/abs/2305.10601

[18] C. Longo, M. Mongiovi, L. Bulla, and G. Tuccari, “HTC-GEN: A Generative LLM-Based Approach to Handle Data Scarcity in Hierarchical Text Classification:,” in *Proceedings of the 13th International Conference on Data Science, Technology and Applications*. Dijon, France: SCITEPRESS - Science and Technology Publications, 2024, pp. 129–138. [Online]. Available: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0012790700003756

[19] Y. Zhang, R. Yang, X. Xu, R. Li, J. Xiao, J. Shen, and J. Han, “TELEClass: Taxonomy Enrichment and LLM-Enhanced Hierarchical Text Classification with Minimal Supervision,” Jun. 2024, arXiv:2403.00165 [cs]. [Online]. Available: http://arxiv.org/abs/2403.00165

[20] J. Shen, W. Qiu, Y. Meng, J. Shang, X. Ren, and J. Han, “TaxoClass: Hierarchical Multi-Label Text Classification Using Only Class Names,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, pp. 4239–4249. [Online]. Available: https://aclanthology.org/2021.naacl-main.335

[21] Y. Huang, J. Song, Z. Wang, S. Zhao, H. Chen, F. Juefei-Xu, and L. Ma, “Look Before You Leap: An Exploratory Study of Uncertainty Measurement for Large Language Models,” Oct. 2023, arXiv:2307.10236 [cs]. [Online]. Available: http://arxiv.org/abs/2307.10236

[22] M. Xiong, Z. Hu, X. Lu, Y. Li, J. Fu, J. He, and B. Hooi, “Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs,” Mar. 2024, arXiv:2306.13063 [cs]. [Online]. Available: http://arxiv.org/abs/2306.13063

[23] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, “Deep learning–based text classification: A comprehensive review,” *ACM Comput. Surv.*, vol. 54, no. 3, apr 2021. [Online]. Available: https://doi.org/10.1145/3439726

[24] J. Weinberger, “Common Factors in Psychotherapy,” in *Comprehensive Handbook of Psychotherapy Integration*, G. Stricker and J. R. Gold, Eds. Boston, MA: Springer US, 1993, pp. 43–56. [Online]. Available: https://doi.org/10.1007/978-1-4757-9782-4_4

[25] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, “DBpedia: A Nucleus for a Web of Open Data,” in *The Semantic Web*, K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 722–735.

[26] J. Sun, R. Jin, X. Ma, J.-y. Park, K.-a. Sohn, and T.-s. Chung, “Gated convolutional neural networks for text classification,” in *Advances in Computer Science and Ubiquitous Computing*, J. J. Park, S. J. Fong, Y. Pan, and Y. Sung, Eds. Singapore: Springer Singapore, 2021, pp. 309–316.

[27] D. Stammbach and E. Ash, “Docscan: Unsupervised text classification via learning from neighbors,” 2022. [Online]. Available: https://arxiv.org/abs/2105.04024

[28] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf

[29] S. Rosenzweig, “Some implicit common factors in diverse methods of psychotherapy,” *American journal of orthopsychiatry*, vol. 6, no. 3, pp. 412–415, 1936.

[30] J. D. Frank, “Therapeutic factors in psychotherapy,” *American journal of psychotherapy*, vol. 25, no. 3, pp. 350–361, 1971.

[31] T. B. Karasu, “The specificity versus nonspecificity dilemma: toward identifying therapeutic change agents,” *The American journal of psychiatry*, vol. 143, no. 6, pp. 687–695, 1986.

[32] J. Weinberger, “Common factors aren’t so common: The common factors dilemma,” *Clinical psychology (New York, N.Y.)*, vol. 2, no. 1, pp. 45–69, 1995.

[33] K. Grawe, “Grundriss einer allgemeinen psychotherapie,” *Psychotherapeut*, vol. 40, no. 3, pp. 130–145, 1995.

[34] L. M. Grencavage and J. C. Norcross, “Where are the commonalities among the therapeutic common factors?” *Professional psychology, research and practice*, vol. 21, no. 5, pp. 372–378, 1990.

[35] W. Tschacher, U. M. Junghan, and M. Pfammatter, “Towards a taxonomy of common factors in psychotherapy-results of an expert survey,” *Clinical psychology and psychotherapy*, vol. 21, no. 1, pp. 82–96, 2014.

APPENDIX

A. Taxonomy of ComFaSyn

Common factors are postulated underlying mechanisms of symptom change in patients that can be found in all forms of psychotherapy [29]–[33]. Our taxonomy serves as a knowledge representation to encapsulate a collection of positive and negative common factors impacting treatment outcomes and to delineate the relationships between these factors. Therefore, our taxonomy encompasses (1) classes, subclasses, and instances of common factors as categories of entities; (2) descriptions of classes, subclasses, and entities; and (3) examples representing typical statements if the factor is present. Numerous common factor models have emerged in psychotherapy research over time and typically encompass specific therapeutic change processes (e.g., corrective experiencing), therapist characteristics (e.g., positive regard), treatment structure (e.g., provision of an explanatory scheme), therapeutic relationship characteristics (e.g., goal consensus), and client factors (e.g., positive outcome expectations) [34], [35]. We have based our taxonomy on these models. However, this paper focused on factors unrelated to the therapist because no therapist training data was available, and the paper emphasized methodological aspects. We included critical external factors impacting the therapy process, such as stressful life events or low platform usability, to test their aptness for later inclusion into patient risk monitoring functionality. In general, we differentiated between factors with a positive versus a negative impact on symptom development. Our taxonomy contains ten classes and 36 instances (12 negative, 24 positive instances). For example, the factor ”Self-efficacy” is positively represented in the patient as ’High self-efficacy expectation’ and ’Mastery experience’, and negatively represented as ’Self-depreciation’. The taxonomy includes a definition and sample sentences for each instance to be included in the prompt.

B. Taxonomy of DBpedia

DBpedia [25] comprises structured data that is extracted from Wikipedia. Figure 4 shows an excerpt of the taxonomy. DBpedia consists of a train ($n = 240942$), validation ($n = 36003$), and test dataset ($n = 60794$). We sample 10 instances from each first layer class from the validation set to evaluate the model performance and the corresponding classes from the

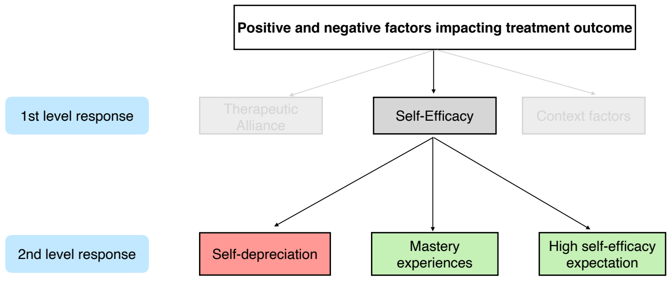


Fig. 3: ComFaSyn Taxonomy Excerpt

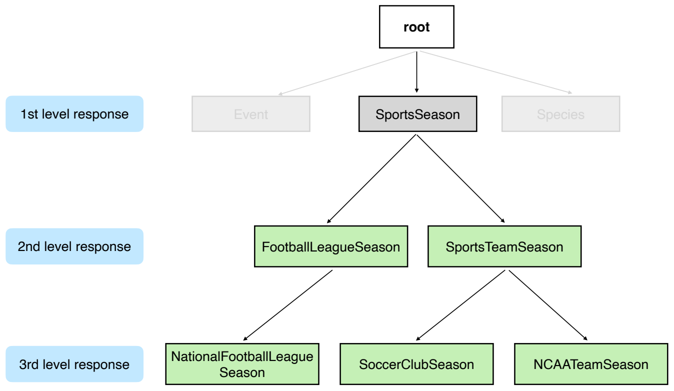


Fig. 4: DBpedia Taxonomy Excerpt

test set to create the examples. More information about the taxonomy of DBpedia is available at the DBpedia webpage[1].

C. Hardware Configuration

Finetuning experiments for the discriminative models were conducted using a cluster equipped with an AMD EPYC 7742 64-Core Processor CPU, featuring 128 cores and a base clock speed of 2250 MHz. The GPU was an NVIDIA A100-SXM4-80GB with 80GB of GDDR6 VRAM and CUDA version 12.2. System memory was comprised of 2TB RAM. Storage was managed through four AI400X and two AI400X2 all NVMe SSD storage servers. The computational environment was standardized on Rocky Linux 8.9 (Green Obsidian) with Huggingface transformer 4.39.3 and PyTorch 2.1.2+cu121 for machine learning experiments.

D. Classification prompts

This appendix presents three prompt templates used for different classification tasks. The first template in Listing 2 facilitates flat classification, where the input is directly assigned to one of the predefined classes without considering any hierarchical structure. The second template in Listing 3 describes the single-path classification method, which involves a step-by-step process, classifying inputs through multiple layers of a taxonomy. Each input is progressively refined, starting from a general class and moving down to more specific subclasses. The third template in Listing 4 is designed for path-traversal classification, where the model predicts the top three

[1]https://www.dbpedia.org/resources/ontology/

Listing 1: JSON schema for the hierarchical classification

```
{
  "name": "hierarchical_classification",
  "strict": true,
  "schema": {
    "type": "object",
    "properties": {
      "layer_classifications_per_input": {
        "type": "array",
        "items": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "explanation": {
                "type": "string",
                "description": "Explanation for
                    the classification."
              },
              "output": {
                "type": "string",
                "description": "The classification
                     result at that layer."
              }
            },
            "required": [
              "explanation",
              "output"
            ],
            "additionalProperties": false
          }
        }
      },
      "final_answer": {
        "type": "array",
        "items": {
          "type": "string",
          "description": "The final result."
        }
      }
    },
    "required": [
      "layer_classifications_per_input",
      "final_answer"
    ],
    "additionalProperties": false
  }
}
```

distinct classes at each layer, repeating the process until a final classification is made. These templates leverage the taxonomy and emphasize structured, step-by-step reasoning to enhance classification accuracy.

E. Response templates

Listing 1 defines the JSON schema used to structure the output of hierarchical classification responses. The schema organizes classification results across multiple layers for each input, where each layer contains an explanation and the corresponding classification result.

```
Classify the input into one of the following classes: NationalFootballLeagueSeason, GolfTournament, SoccerLeague, OlympicEvent,
NCAATeamSeason, CultivatedVariety, Lighthouse, AutomobileEngine, Election, MusicGenre, Museum, OfficeHolder, TennisPlayer, Play,
WomensTennisAssociationTournament, SupremeCourtOfTheUnitedStatesCase, Grape, President, MountainPass, Magazine, Bird,
Racecourse, Economist, RaceHorse, ArtistDiscography, RailwayLine, GrandPrix, Dam, HockeyTeam, Mollusca, Stadium, Amphibian,
Planet, Insect, AmericanFootballPlayer, PublicTransitSystem, Monarch, Single, Musical, SoccerClubSeason, RailwayStation,
MixedMartialArtsEvent, Journalist and BiologicalDatabase.

Input to classify: {{ input }}
```

Listing 2: The prompt template for flat classification

```
Classify the inputs based on the given class hierarchy below.

Class hierarchy:
Each subclass belongs to exactly one superclass. The subclasses for each superclass are shown below to give you a reference for
the class hierarchy:
{{ class_hierarchy }}

Hierarchical classification process:
To classify each input into the final layer class based on this hierarchy, you'll evaluate layer by layer: You'll start by
predicting the first layer class of the input. Once you've predicted the first layer classes, you'll move on to the next layer
and predict the subclass of the superclass. The process is repeated until the final layer is reached. The final answer is just
the last layer class. Repeat this approach for each input instance.

Examples:
Think step by step and print the thinking process. ALWAYS follow this process for any input and provide the final layer output.
For instance, see the following example of a valid path for a hierarchical classification:
Example 1:
first layer output: Agent, second layer output: Person, third layer output: OfficeHolder
Example 2:
Agent: Person: Journalist
first layer output: Agent, second layer output: Person, third layer output: Journalist
Example 3:
first layer output: Device, second layer output: Engine, third layer output: AutomobileEngine

Input to classify: {{ input }}
```

Listing 3: The prompt template for single-path classification

```
Classify the input based on the given class hierarchy you were provided with.

Class hierarchy:
Each subclass belongs to exactly one superclass. The subclasses for each superclass are shown below to give you a reference for
the class hierarchy:
{{ class_hierarchy }}

Hierarchical classification process:
To classify the input into the final layer class based on this hierarchy, you'll apply the following approach to evaluating
layer by layer: Predict the top three classes at each layer. Start by predicting the three most suitable DISTINCT first layer
classes. Once you've predicted the first layer classes, you'll move on to the next layer and predict the top three subclasses of
the superclasses. The process is repeated until the final layer is reached. The final predictions consist of just the last layer
class ranked at the top.

Example
Think step by step and output the thinking process. ALWAYS follow this process for any input and provide the final layer output.
For instance, see the following example of a valid multi-expert hierarchical classification:
Layer 1:
Highest ranked output: Agent, second highest ranked output: Place, third highest ranked output: Event
Layer 2:
Highest ranked output: Organisation, second highest ranked output: RouteOfTransportation, third highest ranked output: Station
Layer 3:
Highest ranked output: PublicTransitSystem, second highest ranked output: RailwayLine, third highest ranked output:
RailwayStation
Final answer:
PublicTransitSystem
Input to classify: {{ input }}
```

Listing 4: The prompt template for path-traversal classification