A Work Project, presented as part of the requirements for the Award of a Master's Degree in Business Analytics from the Nova School of Business and Economics.

Enhancing Product Categorization with LLMs:

Fine-Tuning Decoder-Only Language Models for Hierarchical E-Commerce Product Classification: A Causal Language Modeling Approach

Kuba Maciej Białczyk (60678)

Group Part co-written by:

Elias Markwardt - 61794

Kuba Bialczyk - 61678

Linus Luedecke - 58411

Rafael Alejandro Moles Marrero - 61896

Work project carried out under the supervision of:

Qiwei Han

22/01/2025

Abstract

This research explored techniques to improve Large Language Models performance for Hierarchical Product Classification (HPC), including optimized fine-tuning, optimal prompting techniques, taxonomy-specific Knowledge Graphs, leveraging Retrieval-Augmented Generation, and implementing LLM-based Entity Matching. Tested on benchmark datasets Icecat and WDC-222, these methods significantly enhanced LLMs' ability to solve HPC tasks across various scenarios. Results achieved a hierarchical F1-score (hF) of 0.921, surpassing traditional DL benchmarks (0.85 hF). While not outperforming proprietary models like GPT, the proposed approaches offer a cost-efficient and effective alternative for businesses, demonstrating strong performance without reliance on expensive LLM solutions.

Keywords

Large Language Models, Hierarchical Classification, E-Commerce, In-Context Learning, Fine Tuning, Prompt Engineering, Knowledge Graphs, Retrieval Augmented Generation, Entity Matching

This work used infrastructure and resources funded by Fundação para a Ciência e a Tecnologia (UID/ECO/00124/2013, UID/ECO/00124/2019 and Social Sciences DataLab, Project 22209), POR Lisboa (LISBOA-01-0145-FEDER-007722 and Social Sciences DataLab, Project 22209) and POR Norte (Social Sciences DataLab, Project 22209).

1 Introduction

The e-commerce landscape in 2024 is rapidly evolving, driven by the rise of new players like Temu and the growing impact of social shopping platforms such as TikTok and Instagram, alongside industry leaders like Amazon and Alibaba. Social shopping on social media platforms is reshaping the consumer shopping experience by seamlessly integrating product discovery with purchase. This integration is powered by the accurate categorization of products. With over 1.98 billion products sold on TikTok alone in 2023 (Statista, 2024), there is a growing demand for efficient and automated solutions capable of accurately categorizing products in this rapidly evolving industry.

Beyond e-commerce, product categorization plays a critical role in logistics and international trade, where precise classifications ensure accurate duties and compliance with regulations. Political developments, such as President-elect Donald Trump's proposed tariff reforms targeting specific product categories (Schwartz, 2024), underscore the significant impact that accurate classification can have on businesses. The World Customs Organization posit that misclassifications result in significant financial losses for global businesses each year, including fines, shipping delays, and increased compliance costs (World Customs Organization, 2023). With over 2 billion unique products in the global marketplace, scalable solutions for categorization are essential, driving the need for focused research and innovation.

Techniques to find scalable solutions have evolved over time, beginning with rule-based systems that relied on manually crafted rules. These proved to be both labor-intensive and brittle, often failing to adapt to new categories or datasets. These methods were succeeded by traditional machine learning approaches, which leveraged feature engineering and linguistic patterns to address hierarchical taxonomies. While these approaches improved scalability, they struggled with capturing semantic relationships and hierarchical dependencies. Advancements such as embedding techniques further enhanced semantic representation and classification accuracy, paving the way for more robust solutions. More recently, deep learning models such as Recurrent Neural Networks (RNN) and Long short-term memory (LSTM) have emerged as powerful tools for product categorization, enabling the direct capture of hierarchical dependencies from data while overcoming the limitations of manual feature engineering. Building

on these foundations, pretrained multimodal models (PML) and text-to-text frameworks have pushed the boundaries of scalability and accuracy.

Despite recent advances, product classification (also referred to as "product categorization" in the paper) in a hierarchical context remains challenging for both traditional and deep-learning techniques. Supervised methods rely heavily on labeled data, which can be costly and time-consuming to obtain at scale. Moreover, typical product taxonomies are highly imbalanced, with certain categories dominating others, leading to biased models that struggle to generalize effectively. The diversity of taxonomies across platforms further complicates the task, as models trained on one system may not easily adapt to another. Additionally, variations in product titles, descriptions, brands and images pose significant hurdles for models attempting to standardize and classify products accurately across global markets. These challenges underscore the need for more adaptive and scalable approaches to hierarchical product categorization.

These challenges motivate the exploration of large language models (LLMs) as a promising solution. Leveraging their ability to understand context and semantics to overcome the limitations of traditional and deep-learning methods in hierarchical product categorization. Successfully integrating LLMs into hierarchical product classification represents an exciting opportunity. Offering significant advancements in research by redefining hierarchical classification methodologies, while providing businesses with a cost-efficient alternative to proprietary models.

We hypothesize that employing modern LLM-enabled techniques has the potential to specialize open-sourced LLMs to the task of hierarchical product classification, to achieve performance comparable to state-of-the-art approaches. Their advanced natural language processing capabilities, especially when integrated with domain-specific knowledge, may provide a more accurate and dynamic solution to existing challenges. To investigate, we will employ various approaches to address following research questions:

RQ-A: *Does fine-tuning decoder-only LLMs with hierarchical context enables them to memorize, understand, and accurately predict hierarchical product categories?*

RQ-B: *How can prompting techniques be effectively utilized to improve performance and scalability in hierarchical product classification?*

RQ-C: *How can taxonomy-specific Knowledge Graphs improve the performance of LLMs in hierarchical product classification?*

RQ-D: *How can Retrieval-Augmented Generation enhance product categorization performance, and which approach is most effective?*

RQ-E: *Can LLM-based entity matching be used for hierarchical product classification, and how does it perform compared to current benchmarks?*

Drawing from these research questions, following further hypotheses are made. First, LLM improve generalization compared to non-LLM approaches, leveraging its pre-trained domain-specific knowledge and ability to adapt to diverse contexts. Second, similar results to supervised methods will be achieved at a fraction of the costs and training data usage.

The paper holds the following structure: *Part 2* discusses the theoretical background and related works of hierarchical product classification and the use of LLMs in this domain. In *Part 3* the relevant datasets and benchmarks are introduced followed by a thorough exploratory data analysis in *Part 4*. *Part 5* explains the adopted methodology and outlines experimental setups, forming the basis for the systematic presentation of results in *Part 6*. In *Part 7* a critical analysis of the results is performed, as well as a discussion on limitations and potential future work. Finally, *Part 8* concludes the study with a summary of findings. The subsequent papers tackle the aforementioned research questions individually. *Paper A* investigates various LLM fine-tuning techniques, followed by the evaluation of various prompting methods in *Paper B*. *Paper C* focuses on the integration of domain-specific KGs, while *Paper D* explores the use of various RAG architectures. Finally, *Paper E* introduces an innovative approach to incorporating entity matching into hierarchical product classification.

2 Background and Related Work

This section explores prior studies and related work on traditional product classification, recent advancements in LLMs and their applications in the domains of hierarchical classification and e-commerce.

2.1 Traditional Product Classification

2.1.1 Taxonomies and Hierarchical Classification

The organization of product catalogs into taxonomies or hierarchical structures is a prevalent practice aimed at facilitating efficient search and navigation for users. Examples include Google's Product Taxonomy and Amazon's Product Category Tree. While multi-class classification treats taxonomy categories as isolated entities, hierarchical classification leverages the structure to capture inter-category relationships (Krishnan and Amarthaluri, 2019).

Despite its advantages, automated classification within hierarchical taxonomies presents significant challenges. Lower levels of the hierarchy often require distinguishing between highly similar items, while higher levels must accommodate diverse parent categories. These challenges are further compounded by imbalanced datasets, evolving taxonomies, and the demand for scalable, adaptable models (Kozareva et al., 2016).

Recent studies have explored various strategies to address these challenges, with two prominent overarching approaches emerging: flat single classifiers and hierarchical multi-level classifiers. Flat single classifiers aim to classify leaf nodes in a single step, treating all categories independently without considering the hierarchical structure. In contrast, hierarchical multi-level classifiers follow the taxonomy's structure, categorizing products step by step (Dumais and Chen, 2000). These approaches can further be divided into global methods, which model the entire hierarchy as a single system, and local methods, which focus on smaller segments of the taxonomy, such as specific nodes or levels (Jiang et al., 2022).

2.1.2 Traditional Supervised Methods

Traditional supervised learning methods have been widely explored for product classification on e-commerce platforms, leveraging various algorithms and feature engineering techniques to improve accuracy and efficiency. Several studies have demonstrated the effectiveness of algorithms such as Naïve Bayes, K-Nearest Neighbor (KNN), Decision Tree, Support Vector Machine (SVM), Random Forest, and Logistic Regression when applied to datasets consisting of product titles and descriptions. For instance, Mathivanan et al. (2019) identified KNN as the

best-performing algorithm due to its high accuracy and computational efficiency. In contrast, Patra et al. (2021) emphasized the robustness of Logistic Regression in classifying products.

Domain-specific enhancements to traditional algorithms have further advanced the field. Shen et al. (2011) at eBay Research Lab introduced a feature engineering framework that incorporated rich domain knowledge and linguistic cues, enabling their SVM-based model to address classification challenges in lower levels of hierarchical taxonomies.

Similarly, Kozareva et al. (2016) from Yahoo! Labs developed an automated mechanism for categorizing products based on their titles within a hierarchical taxonomy of 319 categories spanning six levels. By focusing on linguistic patterns inherent in product titles, this system efficiently assigned categories without relying on extensive metadata. Gupta et al. (2016) extended this line of research by incorporating Word2Vec embeddings to create a distributional semantics representation for product descriptions. These embeddings captured semantic nuances in product titles and descriptions, enabling their two-level ensemble approach to better classify products within the taxonomy. By utilizing classifiers corresponding to paths, nodes, and depths within the hierarchy, their method effectively reduced errors, leveraging both semantic relationships and the hierarchical structure for improved classification accuracy.

2.1.3 Deep Learning-Based Methods

The limitations of traditional supervised machine learning methods, particularly their reliance on manual feature engineering and their inability to capture more complex, contextual, or hierarchical relationships in texts, have driven a shift toward deep learning architectures. These models, such as RNNs and LSTMs have revolutionized product classification by learning hierarchical dependencies and semantic structures directly from data. Deep learning approaches are particularly well-suited for handling the complexities of hierarchical classification, where relationships between parent and child categories are crucial for accurate predictions.

Recurrent Neural Networks (RNNs) are widely used in hierarchical product classification due to their ability to model sequential dependencies in text data. Liu et al. (2016) proposed an RNN-based model that leveraged the sequential structure of product titles and descriptions to predict hierarchical categories, effectively capturing dependencies across taxonomy

levels. Building on this, Huang et al. (2019) introduced the Hierarchical Attention-based Recurrent Neural Network (HARNN), which integrates text content with hierarchical category structures. HARNN uses hierarchical attention to capture relationships between texts and taxonomies while modeling level-specific dependencies in a top-down manner. However, RNNs often face challenges such as vanishing gradients, which can hinder their ability to learn long-term dependencies across deep hierarchies.

To address the limitations of standard RNNs, LSTM networks were introduced as a more robust alternative. LSTMs are specifically designed to retain long-term dependencies, making them highly effective for hierarchical classification. Huang et al. (2021) utilized LSTMs to model hierarchical relationships in e-commerce taxonomies, demonstrating improved accuracy in classifying products into deeply nested categories. Their model leveraged the LSTM's gating mechanisms to preserve critical information about parent-child relationships while filtering irrelevant details. Chen et al. (2021) extended this approach by integrating LSTMs with attention mechanisms to prioritize important features in text data, further enhancing the performance of hierarchical product categorization. These studies highlight the strength of LSTMs in handling the depth and complexity of hierarchical structures in product taxonomies.

By learning directly from data and capturing hierarchical dependencies, deep learning architectures offer a powerful solution for large-scale and complex product classification tasks, paving the way for more sophisticated applications in e-commerce.

2.2 LLMs in Product Classification

2.2.1 Emergence of Pre-trained Language Models

The introduction of Transformer-based models (Vaswani et al., 2023) and subsequent pre-trained language models (PLMs) like BERT and GPT (Devlin et al., 2019; Brown et al., 2020) revolutionized NLP, and dethroned previous state-of-the-art LSTM networks for NLP (Melis et al., 2017). PLMs learn general language representations from unlabeled data and can then be adapted to diverse downstream tasks, often outperforming earlier specialised methods (Radford and Narasimhan, 2018).

Domain- and task-adaptive pretraining further enhance performance (Gururangan et al., 2020).

This approach shifts NLP paradigms toward leveraging PLMs for state-of-the-art results across tasks and domains (Min et al., 2021). Moreover, newly introduced in-context learning techniques (Brown et al., 2020) enable rapid adaptation of PLMs to new tasks via zero-shot and few-shot prompts that specialise model solely based on text instructions, reducing the need for expensive training and alterations of models' architectures (Mosbach et al., 2023).

According to Fan et al. (2023), as LLMs are adopted across various industries and domains, researchers continue to explore their expanding capabilities for tasks like NER, QA, and classification. The authors suggest that due to the rapid pace of research in this field, continual investigation into the capabilities and suitability of LLMs for specialized tasks is required.

2.2.2 Recent Advancements in Generative LLMs

Recent advancements in LLMs have produced architectures such as GPT (Brown et al., 2020; Achiam et al., 2024), Claude (AnthropicAI), Google Gemini (Anil et al., 2024), and LLaMa (Touvron et al., 2023; Grattafiori et al., 2024). These models enable rapid adaptation to downstream tasks without designing new architectures or managing complex training.

Sahoo et al. (2024) review classic prompt engineering methods, including zero-shot and few-shot learning, and modern approaches that enhance reasoning abilities. Logic-based techniques (chain-of-thought, self-consistency, tree-of-thoughts) represent a model's generation as a reasoning process. Other approaches integrate external data or third-party outputs (RAG, ReAct) to reduce LLMs hallucinations and improve factual correctness.

Although prompt engineering and RAG can outperform fine-tuned models in some scenarios, fine-tuning excels in specialized tasks (Shin et al., 2023), especially when domain-specific expertise is crucial (MathavRaj et al., 2024). Moreover, fine-tuned models can be combined with techniques such as RAG and advanced prompt engineering for industry-specific applications (de Luis Balaguer et al., 2024; Rangan and Yin, 2024), allowing to create more sophisticated, specialized solutions enabled by LLM-empowered techniques.

2.2.3 Language Models Implementations for Hierarchical Classification

Hierarchical text classification (HTC) has been broadly explored with encoder-only transformers, resulting in complex, specialized approaches. For instance, HiMatch highlights the importance of integrating semantic label information (Chen et al., 2021). HGCLR leverages a Graphformer network to incorporate label hierarchies directly into encoder training (Wang et al., 2022a). HPT enriches model training and inference with hierarchy constraints and hierarchy injection mechanisms (Wang et al., 2022b). Collectively, these strategies demonstrate that encoder-based HTC models has been widely explored by researchers resulting in highly complex and customised model architectures.

Unlike encoder-based approaches that focus on learning vector representations, Sequence-to-Sequence (seq2seq) tasks (e.g. generation, translation, summarization) involve processing sequences of tokens as both input and output. Recursive nature Encoder-decoder make them suitable for text generation tasks (Yang et al., 2023). Risch et al. (2020) pioneered in exploring the use of seq2seq models for hierarchical classification in the context of patent documents. Their findings demonstrated that treating hierarchical labels as sequences significantly improves performance when combined with transformer-based architectures for predicting label encodings as ordered sequences. Numerous frameworks have been proposed to adapt this strategy for hierarchical text classification, highlighting the issues of leveraging semantic meaning of labels, and constraining the model to generate relevant tokens representing existing labels by introducing vocabulary restriction mechanisms, custom Hierarchical Consistency Rate Metric (HCR) or loss functions penalising irrelevant vocabulary (Yu et al., 2022; Jain et al., 2024; Torba et al., 2024).

Recent research has also looked on decoder-only models and in-context learning strategies for HTC in addition to existing encoder-decoder approaches. Gholamian et al. (2024) showed that decoder-only models achieve state-of-the-art outcomes on benchmark datasets when in-context learning techniques were implemented. For instance, Chen et al. (2024b) introduce framework for mixing few-shot in-context learning hierarchical classification with specialized encoder implemented for context retrieval. In their approach they train a very specific method for retrieving in-context learning examples with the use of custom adapted encoder model,

before passing them into LLM prompt. They use hierarchical context by performing single generation per each level of the hierarchy, by using separate prompts at each level. However, even with these encouraging developments, there remains a lack of research on the use of generative decoder-only LLMs for hierarchical categorization, particularly in the e-commerce industry. This points to a promising avenue for further investigation on optimizing decoder-only models for HTC tasks in certain domains.

2.2.4 Examples of LLM Implementations in E-Commerce & Product Categorization

There have been several applications of LLMs in the e-commerce product categorization contexts. A pre-trained language model called E-BERT was developed to include phrase-level and product-level knowledge for overcoming BERT's restrictions to e-commerce tasks (Zhang et al., 2020). Techniques such as Adaptive Hybrid Masking and Neighbor Product Reconstruction were introduced to the model to make it high-performing in several e-commerce-related tasks including product categorization. Also, researchers produced a completely novel approach; from classification to machine translation for product categorization with an encoder-decoder model (Li et al. (2018); Tan et al. (2020)). The latter has the capability to translate product descriptions into a root-to-leaf path within a product taxonomy and outperforms traditional classification methods in terms of predictive accuracy. In addition, the machine translation model can generate new paths between previously unconnected nodes, transforming the taxonomy tree into a directed acyclic graph. This phenomenon improves user navigation and adjusts to new products, which may eventually transform systems of product categorization in e-commerce.

Decoder-only architectures have also been implemented by researchers for the purpose of e-commerce product categorization. Gholamian et al. (2024) approach hierarchical product classification on the WDC-222 dataset by utilizing decoder-only large language models. They compared various LLMs as well as different techniques including Zero-Shot and Few-Shot learning. Additionally, they evaluate flat and hierarchical approaches. Their hierarchical approach comprises using two models, in which the first predicts the top level class, while the second, retrieves the first classifier's output and uses it as input to predict the low level class. In

their results, hierarchical approaches usually underperformed compared to the flat classification methods. Also, their research was focused on various perturbations to the data, and they prove that these decoder LLMs (LLaMa, GPT) seem to be resistant to various data perturbations, which also highlights a potential of LLMs in HTC contexts. Finetuning and prompt engineering were integrated for e-commerce product classification by Cheng et al. (2024), who suggested a unique two-step framework that uses a fine-tuned LLM to predict the top K hierarchical categories for a given product. These are passed to another LLM prompted to reason about the correct final output. However, despite the recent advancements in the field of decoder-only language models and their application in HTC, the topic in the field of e-commerce product categorization remains relatively unexplored.

3 Data and Benchmarks

This section provides an overview of the two datasets used in our study, along with benchmark results for comparison. Both datasets, WDC-222 Gold Standard and Icecat, are widely recognized benchmark datasets. Most pre-processing and cleaning have already been carried out by the Mannheim Data Science Group of the University of Mannheim, which we will briefly summarize here (Nils Richter and Christian Bizer, n.d.).

3.1 Datasets

3.1.1 WDC-222 Gold Standard

The Web Data Commons (WDC)-222 Gold Standard is a manually curated subset of the WDC Training Dataset for Large-scale Product Matching (WDC-LSPM), specifically selecting products from the Icecat dataset. The matching was done using the Global Trade Item Number (GTIN), addressing vendor-specific inconsistencies in identifier terms, such as ”sku” or ”mpn”. Additional columns were created in the WDC-222 dataset, including product titles, descriptions, and category information from Icecat. After cleansing and removing duplicates, the final dataset consists of 2,984 instances with 222 distinct leaf node categories. Similar to Icecat, it is imbalanced, with 100% prevalence for titles and 77% for descriptions. The hierarchy depth

ranges from 2 to 3, with an average of 2.47, mirroring the structure of Icecat. More information in *Part 4*: EDA. As the WDC-222 Gold Standard is derived from the Common Crawl - the largest publicly available web-crawled dataset - it is inherently more heterogeneous than the cleaner Icecat dataset, making it well-suited for testing purposes (Nils Richter and Christian Bizer, n.d.).

3.1.2 Icecat

Icecat is a global provider of multilingual, standardized product data sheets used by e-commerce platforms, ERP systems, and rating portals. It collaborates with 80,000 e-commerce companies and offers two catalogs: the paid "Full Icecat" with 6.5 million product sheets from 24,000 vendors, and the free "Open Icecat," which contains 1 million product sheets from 340 vendors. For our study, we utilized the preprocessed Open Icecat dataset, focusing on the "Computers & Electronics" category, where duplicates and categories with less than 25 products were removed to ensure sufficient training data. The final Icecat dataset contains 765,473 products across 370 categories, with product attributes including title, description, and brand. While every instance has a title and brand, detailed descriptions are available for only 70% of the products (Nils Richter and Christian Bizer, n.d.).

3.2 Benchmarks

In the inital experiments done by Nils Richter and Christian Bizer (n.d.) the performance of various classification methods was assessed on both WDC-222 and Icecat datasets. The lower performance on the noisier WDC-222 dataset, compared to the cleaner Icecat, highlights the challenges posed by noisy data. The table below summarizes the results, with weighted F1 (wF) and hierarchical F1 (hF) scores serving as the primary evaluation metrics. Additional metrics include weighted precision (wP) and weighted recall (wR).

The Icecat dataset, with its clean and structured nature, consistently achieved high performance across all classification methods. Both flat and hierarchical approaches reached near-perfect weighted and hierarchical F1 scores (0.99).

The WDC-222 dataset, characterized by real-world noise and complexity, exhibited lower

Classification System	Icecat				WDC-222 Gold Standard			
	wP	wR	wF	hF	wP	wR	wF	hF
Initial Results								
Dictionary-based approach	0.79	0.43	0.48	0.55	0.72	0.61	0.62	0.71
Flat traditional classifier	0.98	0.98	0.98	**0.99**	0.80	0.72	0.73	0.79
LCPN traditional classifier	0.98	0.98	0.98	**0.99**	0.82	0.73	0.74	0.81
Flat FastText classifier	0.99	0.98	**0.99**	**0.99**	0.86	0.73	0.76	0.83
LCPN FastText classifier	0.99	0.98	0.98	**0.99**	0.85	0.76	**0.78**	0.84
Flat neural network	0.98	0.98	0.98	0.98	0.84	0.76	**0.78**	0.84
Neural network + LCPN traditional classifier	0.98	0.98	0.98	0.98	0.84	0.75	0.76	0.84
Neural network + LCPN FastText classifier	0.98	0.98	0.98	0.98	0.84	0.78	**0.78**	**0.85**
LLM-Based Results (clean data)								
DeBERTaV3 base Supervised (Flat)	0.98	0.98	0.98	–	0.82	0.71	0.73	–
DeBERTaV3 base Supervised (Hierarchical)	0.97	0.98	0.97	–	0.82	0.69	0.72	–
Llama-2 70b-chat (Flat)	0.50	0.37	0.37	–	0.76	0.51	0.51	–
Llama-2 70b-chat (Hierarchical)	0.65	0.40	0.39	–	0.69	0.42	0.38	–
Llama-2 70b-chat (Few Shot)	0.97	0.96	0.96	–	0.90	0.87	0.86	–
GPT-3.5 ver.: 0613 (Flat)	0.90	0.84	0.84	–	0.92	0.87	0.88	–
GPT-3.5 ver.: 0613 (Hierarchical)	0.88	0.66	0.66	–	0.82	0.65	0.66	–
GPT-3.5 ver.: 0613 (Few Shot)	0.98	0.97	0.97	–	0.94	0.92	0.93	–
GPT-4 ver.: 0613 (Flat)	0.94	0.91	0.91	–	0.95	0.89	0.90	–
GPT-4 ver.: 0613 (Hierarchical)	0.89	0.70	0.70	–	0.85	0.80	0.80	–
GPT-4 ver.: 0613 (Few-shot)	0.99	0.99	**0.99**	–	0.96	0.94	**0.94**	–

Table 1: Overview of experimental results: Initial and LLM-based (Gholamian et al., 2024)

performance compared to Icecat. Despite this, methods employing hierarchical setups, particularly those integrating neural networks or FastText embeddings, achieved notable improvements. The hierarchical FastText classifier combined with neural networks at the root level achieved the highest hierarchical F1 score (0.85). This indicates the importance of combining deep learning techniques with hierarchical structuring for noisy datasets, establishing a comparable benchmark for this dataset.

The results also highlight a significant challenge in generalization. Models trained on Icecat struggled to maintain performance when tested on WDC-222, suggesting limited robustness to unseen, noisy data. This emphasizes the need for training strategies or model improvements that account for real-world noise to improve generalization across datasets.

Building on these initial results, recent research explored the use of Large Language Models (LLMs), such as GPT-4 and Llama 2, for product classification on the same datasets (Gholamian et al., 2024). These models were evaluated under both clean and noisy data conditions, with perturbations such as abbreviations and truncations simulating real-world challenges. GPT-4 achieved state-of-the-art performance on both datasets in clean-data scenarios, while also demonstrating strong robustness to noise compared to traditional methods. Few-shot

prompting, which provides contextual examples, further enhanced the model's performance, especially under noisy conditions. Although hierarchical configurations with LLMs showed limitations due to error propagation, flat classification using LLMs and few-shot prompting consistently delivered the best results. These findings highlight the potential of LLMs, particularly the closed GPT-4, to handle the variability and complexity of real-world e-commerce data, offering a significant advancement over traditional classification methods.

Key Obervations

1. **High Performance on Clean Data**: The Icecat dataset achieved near-perfect scores (0.99) with both flat and hierarchical methods, demonstrating strong performance on clean, structured data.

2. **Challenges with Noisy Data**: The WDC-222 dataset showed lower performance, with hierarchical methods (e.g., FastText + neural networks) reaching 0.85 hF. Models trained on Icecat struggled to generalize to noisy data.

3. **LLMs Show Robustness**: GPT-4, a subscription-based model, achieved state-of-the-art results, excelling in noisy scenarios with few-shot prompting, and outperforms free counterpart LLMs.

4 Exploratory Data Analysis

This exploratory data analysis (EDA) provides a comprehensive comparison of the WDC and Icecat datasets, examining their hierarchical structures, product categories, title lengths, descriptions, and brand distributions to uncover key differences and insights into their respective data compositions.

4.1 Hierarchical Analysis

The hierarchies of the WDC and Icecat datasets share a common foundation in the "Computers & Electronics" category, both featuring the same top-level categories Notably, Icecat includes a wider range of subcategories, particularly under Computer Cables and Software, highlighting

a more extensive selection of digital solutions and peripherals. While both datasets emphasize consumer electronics, Icecat's hierarchy reflects a stronger focus on modern trends such as Smart Wearables and advanced Networking components, indicating a broader market scope. Overall, WDC provides a more streamlined structure, while Icecat showcases a richer variety of product offerings within the same foundational categories.

Dataset	Average Depth	Max Depth	Min Depth
WDC	2.34	3	2
Icecat	2.38	3	2

Table 2: Average Category Depth Statistics for WDC and Icecat Datasets

The average depths across both datasets are comparable, suggesting a similar level of granularity in their hierarchical structures. Additionally, the maximum depths are equal, reflecting consistency in the categorization approach for deeper hierarchies. Overall, both datasets exhibit similar structural characteristics, highlighting a shared methodology in their classification systems.

4.2 Categorical Analysis

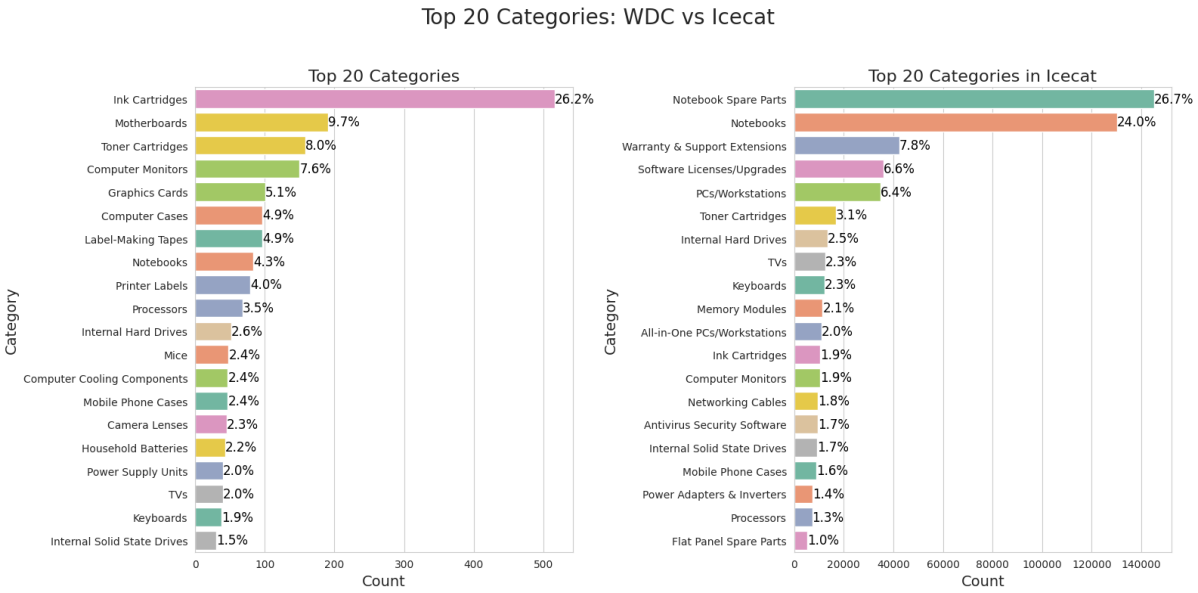


Figure 1: Top 20 Leaf Categories of WDC and Icecat datasets

The comparison between the WDC and Icecat datasets accentuates significant differences in the representation of the products. In WDC, there is higher dependency on consumables, for ex-

ample, ”Ink Cartridges” (516 units, 26.18%), ”Computer Monitors” (150 units, 7.61%), while other categories, for example, ”All-in-One PCs/Workstations” and ”Graphics Cards” represent zero counts - proving lack of diversity. Contrarily, Icecat presents a much wider variety of products, particularly in high-demand categories such as ”Notebooks” (130,294 units, 23.95%) and ”Notebook Spare Parts” (145,020 units, 26.66%).

NaN values at various levels indicate that products are only categorized up to a certain depth. Depth 1 (17 unique categories) and Depth 2 (137 unique categories) contain no NaN-values in the WDC dataset, but it has 924 NaN-values and 103 unique categories in Depth 3. In contrast, the Icecat dataset essentially contains no NaN-values into Depth 1 (17 unique categories) as well as Depth 2 (231 unique categories), but it suffers in Depth 3, where there are 452,976 NaN-values and 161 unique categories. Thus, the two data sources vary with respect to the structure of data, their versatile hierarchy depth across the different categories.

The distribution of Depth 1 categories in WDC and Icecat datasets shows interesting dif-

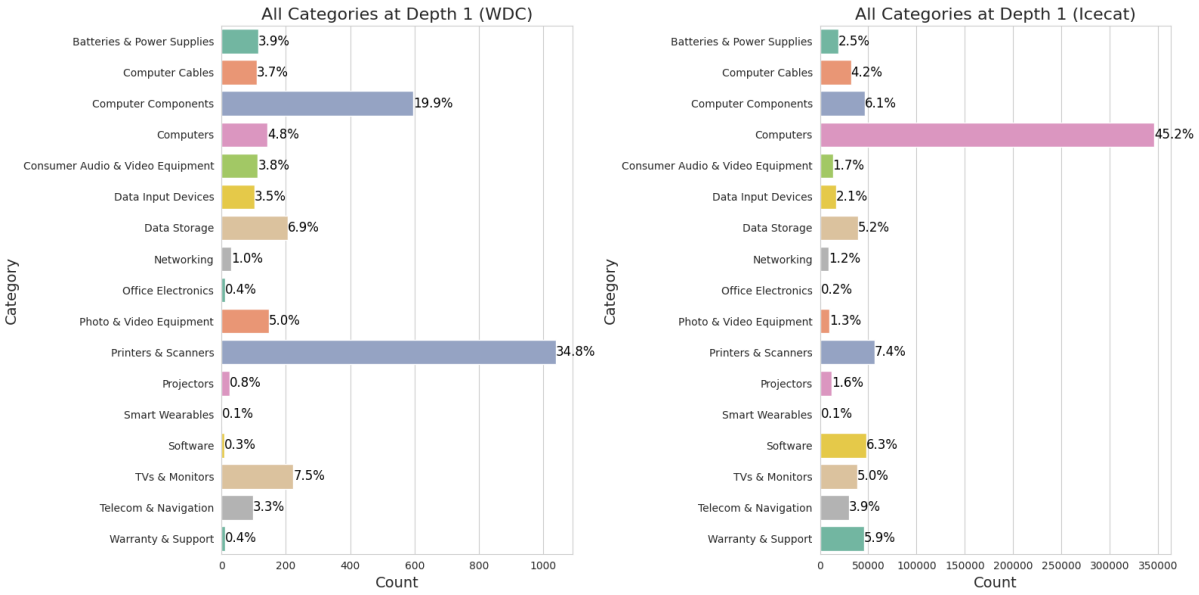


Figure 2: Categories on the first depth in icecat and wdc datasets

ferences regarding the product types they offer (Figure 2). WDC has a very high concentration on ”Printers & Scanners,” with this category representing 34.8% of its products (1,038 units), followed by ”Computer Components” at 19.9% (595 units) and ”TVs & Monitors” at 7.5% (223 units). However, there are some categories, such as ”Smart Wearables” and ”Office Electronics,” which do not have much representation in the dataset. On the contrary, Icecat’s

distribution is clearly dominated by ”Computers,” accounting for 45.2% (346,087 units) of its offerings, leaving ”Printers & Scanners” with only 7.4% (56,404 units). The difference is indicative of how each dataset is skewed towards its unique categories.

Derived from the difference skewness in Depth 1, the Depth 2 distributions of both datasets

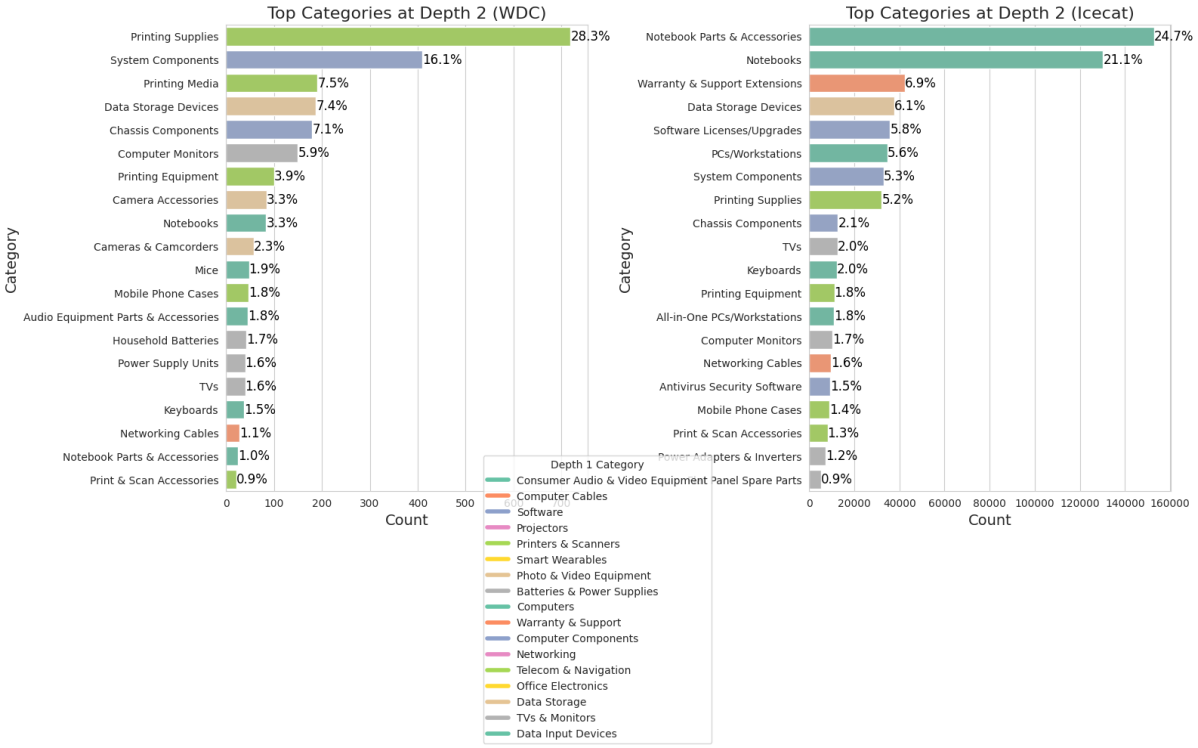


Figure 3: Top Categories on the second depth with the coloring of the first depth

demonstrate quite different structures at depth 2. Of the total categories in WDC, ”Printing Supplies” is the most common, with 28.3% (719), followed by ”System Components” at 16.1% (409) and ”Printing Media” at 7.5% (191). Further categories of in WDC at level included ”Data Storage Devices” and ”Chassis Components”. On the other hand, the prevalent categories at Level 2 in Icecat are ”Notebook Parts & Accessories,” with 24.7% (152,761) and ”Notebooks” with 21.1% (130,294), highlighting the different distributions.

Consequently, also on depth 3 the two datasets contrast greatly. Among all categories in WDC, ”Ink Cartridges” alone comprises 25.0% (516), followed by ”Motherboards” at 9.3% (191), while ”Toner Cartridges” makes up 7.7% (158). The distribution is leaning towards printing and computer supplies. In contrast, Icecat’s largest category at level 3 is ”Notebook Spare Parts” with 46.4% (145,020), followed by ”Toner Cartridges” and ”Internal Hard Drives”. Once again, standing in stark contrast to WDC. Concluding the categorical analysis, WDC data

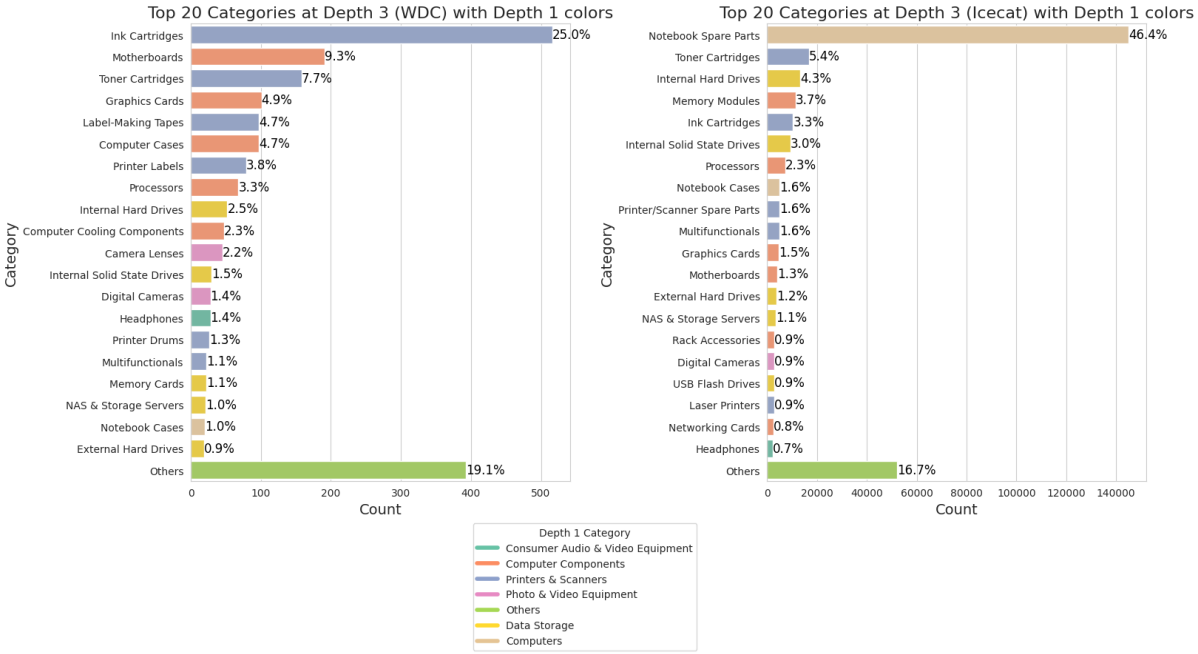


Figure 4: Top Categories on the third depth with the coloring of the first depth.

is skewed toward printing products while Icecat leans more on portable computing accessories, indicating how distinct product focuses can arise in different data sources.

4.3 Product Title Analysis

Distribution of Title Lengths:

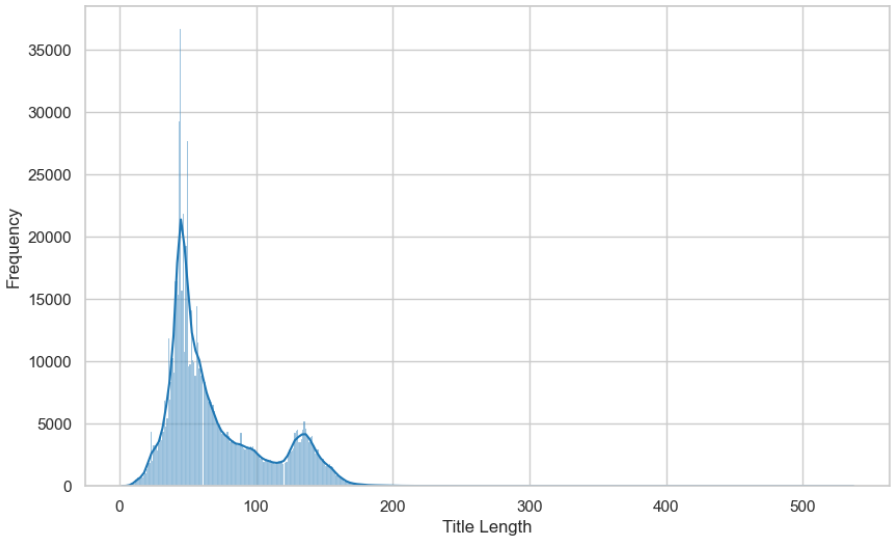


Figure 5: Distribution of Product Title Length Icecat

The analysis of Icecat data reveals that the lengths of product titles are heavily skewed toward shorter titles, with a significant peak observed in the 40-60 character range. With 765,473 data

points, the mean title length is 70.42 characters, with a corresponding standard deviation of 36.26, suggesting some deviation of longer title lengths. The median length is 57 characters, and most titles are found to fall within the quartile range of 45 to 90 characters. Maximum title length interestingly reaches up to 537 characters, a substantial outlier from the average. These findings would suggest that most product titles tend to be short but that there is quite a wide range of lengths, with some few significantly longer outliers.

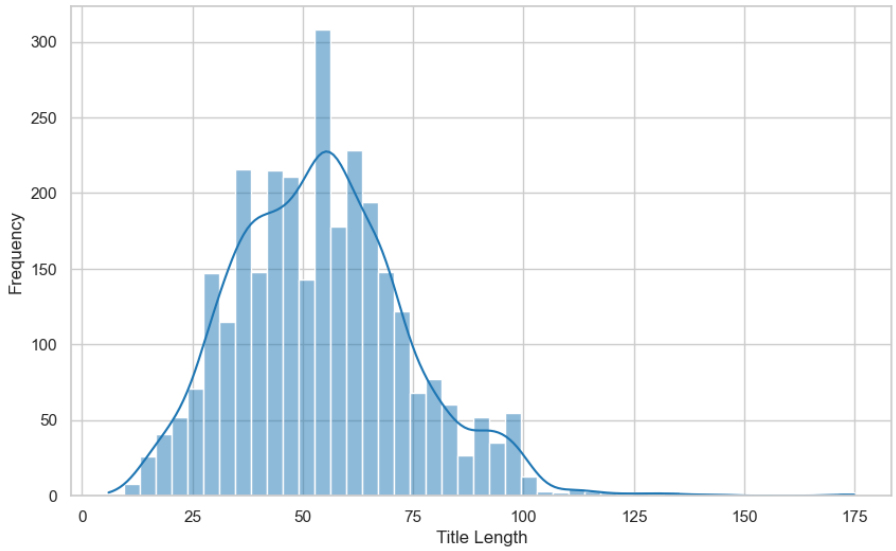


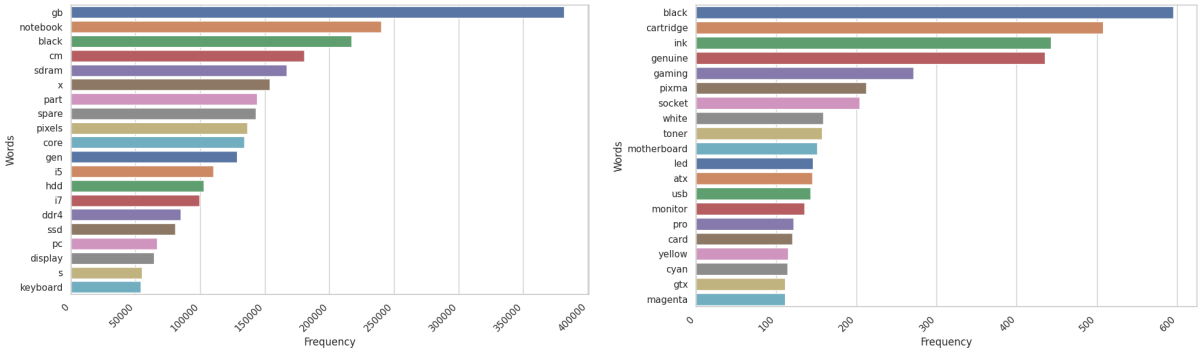
Figure 6: Distribution of Product Title Length WDC

Compared to Icecat, the title lengths of products in the WDC dataset are distributed more symmetrically. There are 2,984 titles. The average title has 54.32 characters with a standard deviation of 19.66, indicating a tighter spread around the mean. The median title length falls at 54 characters, which shows its close affinity to the mean, thus implying that the data is fairly normally distributed, with its mean roughly at the center around 50 to 60 characters. Most title lengths lie in the range between 40 to 66 characters, while the longest title has 175 characters, marking a much smaller variation in the upper range when compared to Icecat data. This further implies that the titles in the WDC dataset are generally shorter and more concise.

Frequent Words

Insights into common phrases or nomenclatures found in product descriptions are the core of this analysis. The study of the frequent words in Icecat product titles. Some of the most used words are ”gb,” ”notebook,” ”black,” and ”cm,” words that clearly point toward technology

relevant terms, specifications, and descriptive words. Terms like ”intel,” ”sdram,” ”hdd,” and ”i7” are related to hardware specifications. Interestingly, when brand names and numbers are eliminated from product titles , the remaining most common words remain related to technical specifications, such as ”ssd”, ”pc”, and ”display”. Domain specific terms and numerical characters are important provide models valuable insights, helping the categorization process.



(a) Icecat (b) WDC

Figure 7: Frequent Words in Product Titles

Results of the analysis of common words in WDC product titles show that they lean heavily on products related to the printing, gaming, and electronics industries. These include common words such as ”black”, ”cartridge”, ”ink” and ”genuine” with a strong dependency on printer products, especially ink and toner cartridges. Other frequent terms such as ”gaming”, ”socket”, ”motherboard” and ”atx” show a focus on computer hardware topics, primarily gaming systems and assembly components. The last terms are more evidence of the visual display technology and connectivity importance, namely ”led”, ”monitor” and ”usb.” Some of the words also contain ”pixma” (from Canon brand) and ”pro”, meaning that there are also consumer-grade and professional items. This spread of terms is in line with the categories included within WDC. There is a heavy skew toward technology, gaming, and office supplies.

4.4 Description Analysis

WDC: 25,87% of the descriptions are empty. The most frequent words include ”summary”, ”black”, and ”product”. Some numbers like ”1” and ”2” also appear repeatedly. In some cases, there are multiple observations with the same gtin but with varying descriptions. The most frequent length is 28 characters, the following histogram shows the distribution of descriptions lengths:

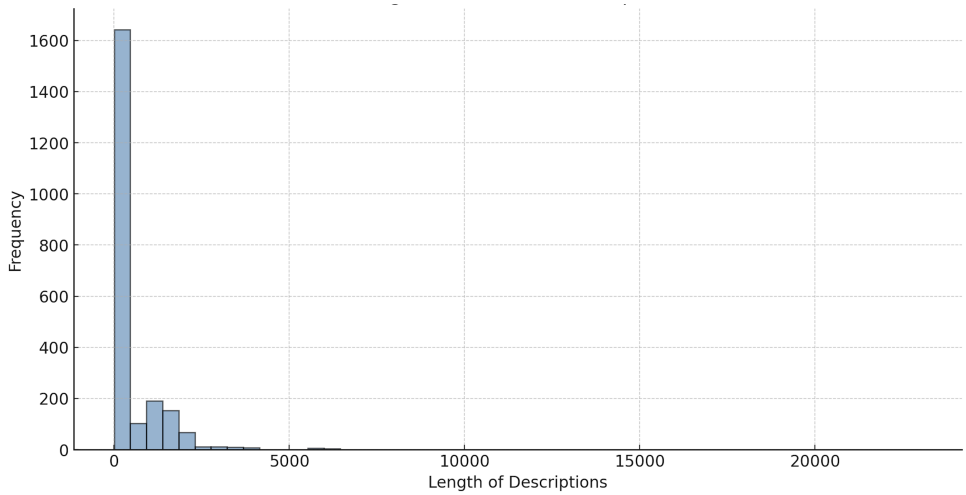


Figure 8: Length Distribution on WDC

The most frequent bigrams in product descriptions highlight technical specifications and features, such as ”sf utp” (likely a cable type), ”intel xeon”, and ”xeon processor”, highlighting the importance of processor-related terms. Other notable examples include ”photo paper” and ”533 MHz,” indicating specific product categories and technical attributes. Descriptive terms like ”summary description” and ”auto generated” suggest how some descriptions are structured. In trigrams, ”intel xeon processor” is the most frequent. Other common trigrams, such as ”cat 5e sf” and ”5e sf utp,” relate to networking cables, while ”processor 06 GHz” and ”GHz 512k cache” highlight processor details. Trigrams like ”premium a4 70x33” focus on specific paper sizes and formats.

Icecat: The description analysis focus on the columns *SummaryDescription. LongSummaryDescription* and *SummaryDescription. ShortSummaryDescription* as there are no missing values, and it contains the most representative descriptions for product categorization.

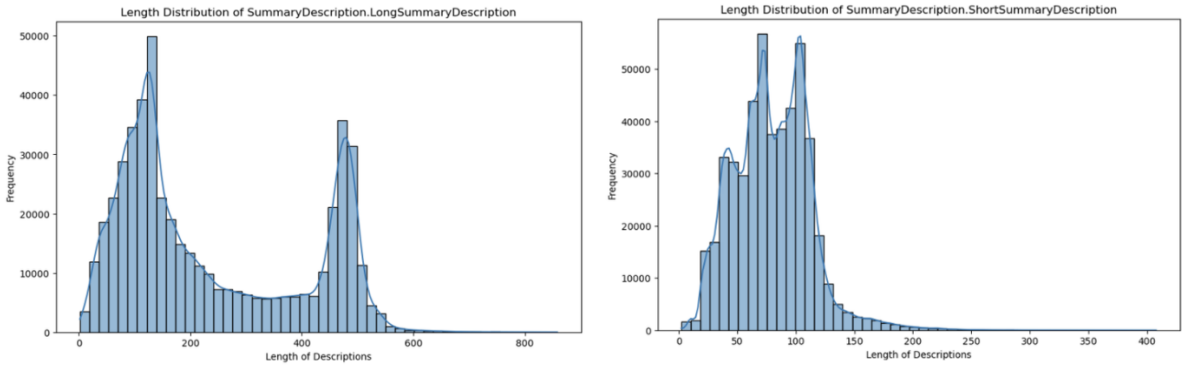


Figure 9: Length Distributions on Icecat Long and Short Descriptions

The length distribution of descriptions shows two distinct patterns for both long and short

summaries. The long summary descriptions exhibit a bimodal distribution with peaks around 150 and 400 characters. The majority of the descriptions cluster in these two ranges, indicating that there are mainly two different products descriptions with varying lengths, possibly depending on the product type or complexity of the description. The short summary descriptions primarily range between 50 and 100 characters, with a much more noticeable bias toward shorter lengths, making them more standardized in format compared to the long summaries.

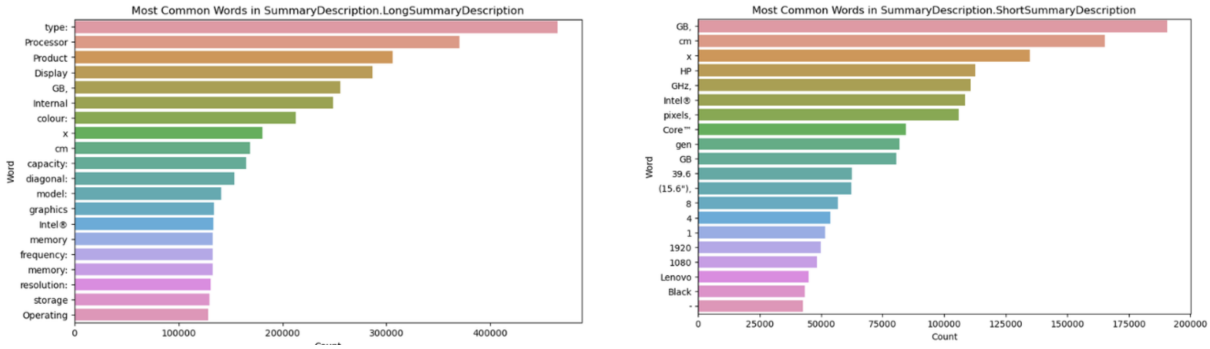


Figure 10: Most common words in Icecat

Most common words: The bar charts illustrates the most common words in both long and short summary descriptions. In the long descriptions, technical specifications such as ”Processor”, ”Product”, ”Display”, ”GB”, and ”Internal” dominate. Similarly, the short descriptions frequently include terms like ”GB”, ”cm”, ”x”, and ”HP”, which are also focused on product characteristics and technical specs.

4.5 Brand Analysis

The WDC dataset lacked a dedicated brand column, so brands were extracted by taking the set of unique brands present in the Icecat data. WDC titles were then searched for instances of this list, extracted and assigned to the observation’s Brand. Within the extracted brands (Fig. 11), the distribution is highly imbalanced, with Canon and Epson dominating Printers & Scanners, while Asus, MSI, and Samsung lead in Computers and Components.

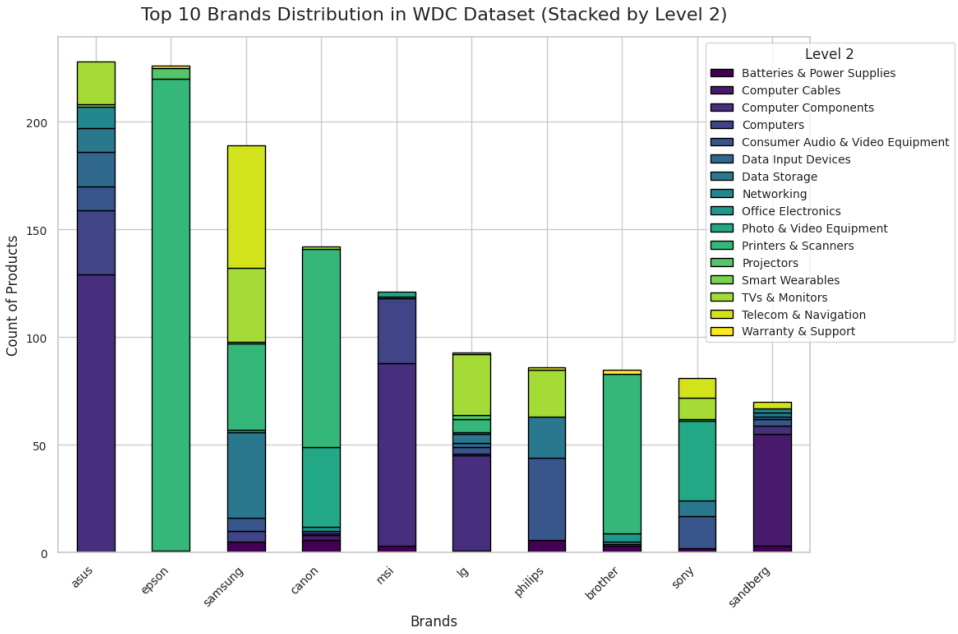


Figure 11: Top 10 Most Frequent Brand in Categories - WDC

Unsurprisingly, with Icecat's leaning towards "Computers", HP overwhelmingly dominates the brand distribution. Lenovo, Acer, and ASUS follow, but with much lower representation. This is highlighted in (Fig. 12) Icecat's brands consist of key market leaders in computing and warranty-related categories, creating a skewed distribution of brands.

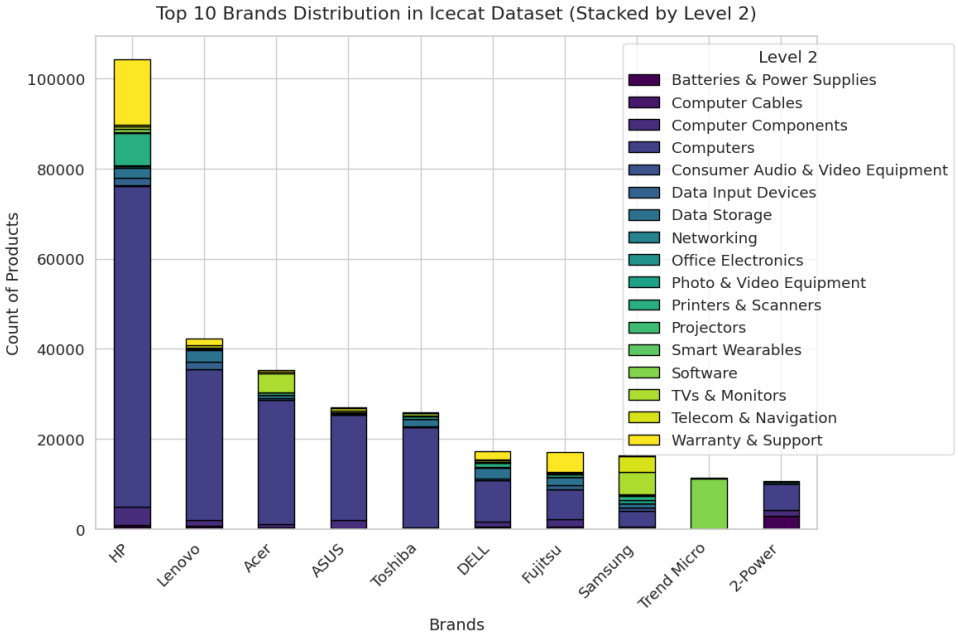


Figure 12: Top 10 Most Frequent Brands in Categories - Icecat

This contrast underscores differing orientations, which are in line with findings from the hierarchical and categorical analysis.

5 Methodology

This paper explores the enhancement of product classification using Large Language Models. To achieve this, five promising approaches are analyzed in detailed deep dives, each focusing on different strategies. All deep dives utilize the same datasets, evaluation metrics, and, in some cases, the same models, which are briefly introduced below. The aim is to conduct a comprehensive comparison, drawing conclusions about which models and approaches perform better in specific scenarios.

5.1 Data Preparation

Different attribute combinations were explored, but for the final models only the title was used. Additionally, the Path and Category columns were utilized for evaluation.

Title	Initial Path	Category Name
ASUS K31CD-IT049T PC 6th gen Intel® Core™ i7 i7-6700 16 GB DDR4-SDRAM 1000 GB HDD Black Tower	Computers & Electronics > Computers > PCs/Workstations	PCs/Workstations
HP 686915-A41 notebook spare part Keyboard	Computers & Electronics > Computers > Notebook Parts & Accessories > Notebook Spare Parts	Notebook Spare Parts
C2G 1m ST/SC Plenum-Rated 9/125 Duplex Single-Mode Fiber Patch Cable - Blue fiber optic cable 39.4" (1 m)	Computers & Electronics > Computer Cables > Fibre Optic Cables	Fibre Optic Cables

Table 3: Three sample rows showcasing the features used in the Icecat dataset.

Further preparation involves exploding the category path to retrieve the actual categories per level. In cases where Level 3 does not exist (i.e., the path has only three entries), the category from Level 2 is propagated to Level 3. This approach is commonly used in hierarchical classification tasks where the depth of categories may vary across different paths (Guo and Zhao, 2021). Lastly, the first-level categories are omitted from the datasets, as they consistently correspond to 'Computers & Electronics' for every observation, making them redundant.

Initial Path	Adjusted Path	Category Level 1	Category Level 2 & Level 3
Computers & Electronics > Computers > PCs/Workstations	Computers > PCs/Workstations > PCs/Workstations	Computers	PCs/Workstations

Table 4: Example row illustrating the Adjusted Path and its derived category levels.

This paper focuses exclusively on the title as input feature, rather than including the description or other attributes, in order to minimize the number of input tokens for the LLM. As highlighted

in Groß et al. (2024), the title contains the most relevant information and is the most important feature for making accurate predictions. Discrepancies in available attributes pose significant challenges in e-commerce Martinek et al. (2024). However, compared to other attributes, product titles generally seem to be consistently available (d'Amato et al., 2018), a pattern supported by the benchmark datasets Icecat and WDC.

Title	Description
acer aspire e1 522 65208g1tmnkk specifications	*(NaN)*
msi gs73vr 7rf 207uk nvidia gtx 1060 6gb gddr5 17 3 uh d ips 4k intel i7 7700hq gaming laptop	gs73vr 7rf stealth pro 4k 207uk 17 3 uh d ips 4k 3840 x 2160 i7 7700hq nvidia gtx 1060 6gb gddr5 256gb ssd 2tb 7200rpm hdd 16gb 2400mhz ddr4 windows 10 home 64 bit 2yr warranty

Table 5: Example Rows from the input WDC-222 dataset, showcasing the inconsistency of Description

5.2 Large Language Models

For the purpose of research of individual LLM enabled techniques (Paper A-E) various LLMs were implemented in experiments. The following provides a non-exhaustive overview of state-of-the-art open-source LLM families referenced in the papers.

5.2.1 Pythia

Pythia is a suite of decoder-only LLMs designed to facilitate analysis across varying training scales (Biderman et al., 2023). It provides eight open-source GPT-NeoX-based models at different sizes, each accompanied by precise training states and hyperparameters. This transparency enables researchers to observe how scaling, optimization, and data affect performance. By offering well-documented internals and standardized transformer-based components, Pythia fosters reproducibility and improves our understanding of language model behavior.

5.2.2 LLaMa

LLaMa (Large Language Model Meta AI) (Touvron et al., 2023) is a family of foundational models (1B–405B parameters) that consistently outperformed competitors upon release. Its latest version, LLaMa 3.2 (November 2024), includes lightweight autoregressive models (1B and 3B parameters) for on-device inference. LLaMa's architecture incorporates RMSNorm

instead of standard normalization, replaces ReLU with SwiGLU for enhanced performance, and uses rotary positional embeddings instead of absolute positional embeddings.

5.2.3 Mistral

Mistral 7B (Jiang et al., 2023a) is designed to rival larger models like LLaMA-2-13B with about half the parameters. Achieving efficiency through re-architected transformer blocks and improved attention patterns, Mistral optimizes training pipelines, parameter initialization, and scaling rules. These innovations allow higher-quality performance with fewer resources, making advanced LLMs more accessible. Its latest version, Mistral-7B-v0.3, is available on the HuggingFace Hub.

5.2.4 Instruct Models

Instruct models are fine-tuned to better follow user instructions, often through supervised tuning or Reinforcement Learning from Human Feedback (RLHF). Unlike base models, which are trained broadly and remain open-ended, instruct variants focus on producing responses aligned with specific prompts and guidelines. This makes them more suited for practical tasks employing prompting and in-context learning like information retrieval, and content generation under defined constraints, ensuring outputs are both accurate and useful. Both LLaMa 3.2 and Mistral-7B-v0.3 exemplify this paradigm by offering two open-source variants: a "base" model and an "instruct" model.

5.3 Topic Deep Dives

5.3.1 Fine Tuning LLMs (Paper A)

To investigate the effectiveness of fine-tuning for hierarchical classification, five pre-trained LLMs of varying sizes were fine-tuned on the same dataset using four distinct fine-tuning strategies. Due to computational constraints and evidence suggesting that pre-trained language models perform better on balanced datasets (Weiss and Provost, 2001), the highly imbalanced Icecat dataset was subsampled. A clustering-based subsampling technique (Yen and Lee, 2009)

was employed to preserve the original variance while reducing the training dataset to approximately 30,000 observations. The resampled dataset was subsequently divided into training, validation, and test subsets. For efficiency, all models were trained exclusively on product titles, as shorter inputs accelerate training time (Vaswani et al., 2023) and titles alone have been shown to provide sufficient context for accurate classification (Dai et al., 2020).

The models selected for this analysis, listed in ascending order of size, include Pythia70m, Pythia410m, LLaMa3.2-1B (base and instruct), LLaMa3.2-3B (base and instruct), and Mistral-7B-v0.3. To assess the impact of hierarchical information during training, all models were initially fine-tuned for Flat Classification, where the objective was to generate a single phrase corresponding to the level 3 category. Subsequently, three hierarchical fine-tuning approaches: Hierarchical Completion, Hierarchical Instruct, and Curriculum Learning were applied, as described in detail in Paper A. In these hierarchical setups, the models were tasked with unrestricted causal language modeling to generate sequences of hierarchical labels. Notably, the relatively small Pythia models underwent full fine-tuning, while larger models were fine-tuned using QLoRA, a parameter-efficient fine-tuning technique that significantly reduces the number of trainable parameters while maintaining strong inference performance.

5.3.2 Different Prompting Techniques (Paper B)

This section explores how prompting techniques can be effectively utilized to enhance performance and scalability in hierarchical product classification, a critical challenge in e-commerce. By exploring how the advanced prompting techniques chain-of-thought (CoT) and prompt chaining can excell in HPC, the study uncovers their potential to navigate complex taxonomies and improve classification across multiple hierarchical levels.

Results from this exploration will provide a comparative analysis of developed techniques, offering a roadmap for useing optimized Techniques to perform HPC with LLMs. This has significant implications for improving e-commerce workflows, enabling rapid deployment and accurate product categorization while maintaining cost-effectiveness. Future directions include combining methods and refining domain-specific LLM fine-tuning to enhance model capabilities further.

5.3.3 Knowledge Graphs (Paper C)

This paper explores the integration of taxonomy-specific Knowledge Graphs (KGs) into LLMs for hierarchical product classification in e-commerce. KGs represent knowledge in the form of structured triplets (entity – relation – entity), enabling symbolic reasoning. When integrated with LLMs, KGs provide structured, factual external knowledge that not only enhances the precision and interpretability of LLM predictions but also contributes to more grounded reasoning. LLMs, in turn, support the expansion and enrichment of KGs through their advanced natural language processing capabilities (Chen et al., 2024a). This synergy seeks to mitigate the challenges of LLM rigidity and hallucinations by leveraging the interpretability and factual consistency of KGs.

In this study, multiple KG integration techniques are evaluated, ranging from representing only the taxonomy as a KG to extending the taxonomy with new entity nodes using Named Entity Recognition (NER) methods. These approaches are tested across various LLMs using the WDC-222 Gold Standard dataset. All experimental configurations follow the Local Classifier per Parent Node (LCPN) approach and are compared against hierarchical and flat baselines without KG augmentation. The hypothesis posits that taxonomy-specific KGs will enhance both the interpretability and precision of LLMs in hierarchical product classification tasks.

5.3.4 Retrieval Augmented Generation (Paper D)

Retrieval-Augmented Generation (RAG) systems enhance Large Language Models (LLMs) by integrating external knowledge about similar products into the classification process, thereby improving accuracy and relevance. By leveraging RAG, the system can retrieve information which serves as a foundation for more informed and contextualized categorization.

The analysis begins with a comprehensive study of configurations that optimize RAG performance. Various retrieval methods are explored, including standard retrieval techniques, graph-based approaches, and an innovative multi-tiered retrieval framework. Each method is assessed and compared in fetching relevant products, which is important for the classification pipeline.

The approach then, examines the use of retrieval similarity as a standalone method for categorization, independent of LLMs, serving as a baseline for comparison with more advanced

RAG systems. These systems include NaiveRAG, which represents a straightforward implementation, AdvancedRAG, which introduces refined retrieval and classification strategies, GraphRAG, which leverages graph-based retrieval for enhanced contextual understanding, and HybridRAG, a combination of techniques including graph and vector retrieval. To build the prompt different prompting techniques such as Zero-shot, Few-shot, CoT, Prompt Expansion and Prompt chaining are taken into consideration. Overall, the study explores how these RAG systems can be effectively applied to both, flat and hierarchical step-by-step categorization processes.

Overall, this study emphasizes the evolution of RAG systems from basic to advanced configurations, highlighting their potential for product classification.

5.3.5 Entity Matching (Paper E)

Entity matching, often referred to as product matching in the e-commerce context, could play a crucial role in hierarchical product classification. At its core, entity matching aims to identify identical or closely related entities across datasets, even when they are described differently. In e-commerce, this translates to finding the same or similar products listed under varying titles, descriptions, or formats by different vendors.

The process consists of two main steps: blocking and matching. Blocking reduces computational complexity by identifying the k most similar products, narrowing the search space to relevant matches. The matching step identifies the most similar product (based on the title in this study) and uses this ”product match” to predict the input’s category based on the matched product’s category. Similar to methodologies from *Paper 4* and RAG, top-k blocking and similarity scores are used as the baseline, excluding LLMs. LLMs are then integrated on top of this baseline to assess whether LLM-based entity matching can improve performance and achieve comparable results to other approaches tested in this paper.

To reiterate: in contrast to the other four approaches, the LLM is not tasked with directly predicting a category. Instead, it identifies the most similar product in the reference dataset, which then implicitly determines the category.

5.4 Evaluation

The different model performances will be analyzed to draw conclusions about model or dataset characteristics, ultimately enabling the formulation of general insights and recommendations.

5.4.1 Performance Metrics

The WDC-222 publication evaluates several machine learning models using a variety of metrics. Specifically, they employ level-specific weighted Precision (wP), weighted Recall (wR), and weighted F1 (wF) scores at each level of hierarchical prediction for both the Icecat and WDC-222 evaluation datasets. When referring to these metrics, by default, level 3 metrics are meant. Additionally, they implement the hierarchical F1 (hF), precision (hP) and recall (hR) scores (Kiritchenko et al., 2006), which accounts for partial hierarchical predictions, that are easily implementable thanks to HiClass python package (Miranda et al., 2023). In the experiments performed in this paper, we apply the same metrics, as they provide a robust estimation of model performance in real-world scenarios characterized by highly imbalanced datasets, which is typically the case for hierarchical classification tasks. Furthermore, the evaluation of wF and hF scores offers insights into the models' ability to "memorize" the hierarchical structure. The hF score is also calculated in flat scenarios.

Additionally, we have developed a metric to assess the coherence of label name representations in the model outputs. The "Phrase Coverage" (or "Label Coherence") metric quantifies the proportion of classifications at each hierarchical level that correctly identify the label name from the subset of available labels for that level. This metric enables the evaluation of the model's ability to memorize label names at each level, thereby enhancing its understanding of the label hierarchy.

5.4.2 Models' Performance Analysis and Comparison

To evaluate the performance of each approach, we employ a comprehensive set of metrics, fully detailed abive. Level-specific metrics reflect how accurately each model identifies correct labels at increasing granularity, while hierarchical metrics measure the severity of misclassifications by considering their placement within the taxonomy. Thus, even when errors occur,

those that remain close to the true class have less impact on the final score. In combination, these metrics provide a balanced evaluation. wF and hF highlight both fine-grained precision and overall taxonomic coherence. Precision and recall offer further insight: models with high precision rarely predict incorrect classes, while those with high recall rarely miss correct ones. By examining all these metrics together, we gain a nuanced understanding of each model's strengths and weaknesses in hierarchical product classification.

5.4.3 Misclassification

A comprehensive analysis of misclassifications is conducted across all the explored approaches, with a focus on high-representation categories such as "Printers & Scanners," "Computer Components," and "TVs & Monitors". Evaluations take place at different hierarchical levels, emphasizing leaf-level (Depth 3) predictions, where overlapping categories like "Ink Cartridges" and "Printer Ink Refills" present significant challenges. Misclassifications are logged and analyzed for trends, highlighting shared challenges across approaches, such as systemic errors in closely related categories. This analysis is crucial for identifying the limitations of classification models and understanding the root causes of errors, enabling targeted improvements in both model performance and taxonomy structure. These insights inform potential taxonomy improvements, including redefining overlapping categories and introducing intermediate subcategories to enhance classification accuracy and reduce ambiguity.

5.4.4 Imbalanced Classes

In this section, the imbalanced class performance across various individual models is analysed. As shown in the EDA section (Categorical Analysis 4.2), the WDC and Icecat datasets are highly imbalanced. The importance of being able to perform in imbalanced scenarios is highly relevant to real-world applications. Product category trees, in particular, are often characterized by such imbalances, making it crucial to develop models that can handle these challenges effectively. For the underrepresented classes, evaluations were performed on a subset of the 100 least-represented categories in the WDC dataset, resulting in 121 observations, with an average of 1.21 observations per class. For the overrepresented classes, the top 5 largest categories were

selected, resulting in 1,213 observations, with an average of 243 observations per class. This further highlights the highly imbalanced nature of the dataset.

5.4.5 Label Coherence Analysis

LLMs, due to their broad training data, often encounter "LLM hallucination," a phenomenon in which the model produces text closely resembling accurate content yet lacking factual correctness (Huang et al., 2024). This issue is particularly relevant for hierarchical classification tasks reliant on LLM outputs. As noted by Risch et al. (2020), treating hierarchical text classification (HTC) as a sequence-to-sequence (seq2seq) task and incorporating the semantic meaning of labels yields better results than generating encoded label signatures. Consequently, it is crucial that LLM-based solutions produce coherent label names both at a flat level and across hierarchical tiers, since even minor deviations, such as subtle wording differences or typos, can severely affect classification accuracy.

Several studies investigating HTC as a seq2seq task have implemented mechanisms to constrain the model's vocabulary, ensuring more coherent label generation (Torba et al., 2024; Yu et al., 2022; Jain et al., 2024). The solutions presented in this paper, however, do not rely on such constraints, instead assuming that LLMs, particularly when guided through in-context learning, can inherently produce coherent labels. To assess label coherence, each model's predictions at all hierarchical levels are compared against established label lists, and the proportion of exact matches is calculated. This coherence metric is critical, as it reflects the model's ability to produce the correct tokens for each label. Any deviation directly diminishes classification performance, as in real-world scenario products without a coherent label are not assigned to any class.

5.4.6 Generalization

Generalization is crucial for product categorization, helping models perform well across diverse datasets and tackle real-world challenges like data variety and multiple languages. The WDC-222 benchmark dataset is already seen as "unstructured" compared to the cleaner Icecat dataset, making it a good indicator for model generalization. However, we wanted to take this even

further, by testing different languages and title structures. This approach addresses WDC-222's English-only limitation and showcases LLMs' ability to handle unseen input effectively.

To extend the generalization analysis, we created a test dataset consisting of 30 inputs sourced from amazon.de. The sample size is small and does not ensure statistical significance, but it provides an initial indication of model performance on real-world data. We selected following categories from the Icecat taxonomy, representing overrepresented as well as underrepresented classes: `Computer Monitors`, `Ink Cartridges`, `Digital Photo Frames`, `Sport Watches`, `Smartwatches`, `Activity Trackers` and `Soundbar Speakers` with their respective Paths.

We manually identified Amazon products matching these categories within the Electronics section. To simulate multilingual and structurally diverse datasets, we retrieved titles in multiple languages, by switching the interface language from German to Turkish, English, and Polish. While multilingual product titles on Amazon may come from two sources: vendor-provided translations or "just" machine-generated translations, this approach allowed us to capture a realistic diversity of input data while maintaining consistency within the Electronics category.

Given the differences in model architectures and training approaches, we expect varying levels of generalization across the tested methods. Specifically, we hypothesize that methods with stronger capabilities for handling multilingual and structurally diverse inputs will demonstrate better performance on this new input set. This experiment will highlight the relative strengths and weaknesses of different methods when faced with variations in title structure and language.

5.4.7 Complexity Analysis

The tested LLM approaches differ in their complexity, training data requirements, inference efficiency, and levels of rigidity and adaptability. This section focuses on a theoretical and conceptual discussion of these factors, highlighting the trade-offs and considerations inherent in each method.

6 Results

The solutions chosen for the comparison in this section represent the most suitable methods based on experimental results from Papers A-E. In the following sections, the best-performing approach derived from the individual papers will be referred to as Fine-tuning, Prompting, KG, RAG and Entity-matching.

Method	Description	Model Used
Fine-tuning	Comprises a Mistral-7B-v0.3 LLM fine-tuned with QLoRA for accurate generation of hierarchical text sequences based on product titles as inputs.	Mistral-7B-v0.3
Prompting	A multi-stage top-3 solution that iteratively filters the taxonomy to identify the best-fitting leaf nodes using LLM-based prompting techniques.	Mistral-7B-v0.3
KG	Combines predictions of three weak LLMs. Differences in predictions trigger a re-prompt where taxonomy-specific KGs are injected. KG-augmented predictions are majority voted; if unresolved, an advanced LLM acts as judge.	Mistral-7B-v0.3-Instruct, Llama 3bn, 8bn, 70bn
RAG	A flat Hybrid RAG that combines Knowledge Graphs and vector retrieval. Cohere Reranker v3 reorders and selects the most relevant documents.	Mistral-7B-v0.3-Instruct
Entity Matching	Combines Sentence Embedding + Cosine Similarity as top-k blocking, with Google Flan T5 Large serving as classifier on top using the LLM Select strategy.	Google Flan T5 Large

Table 6: Comparison of Final and Best Performing Models

6.1 Performance Metrics

Method	F1 Level 1	F1 Level 2	wF	hF	wR	wP	hR	hP
Fine-tuning	0.94	0.86	0.80	0.87	0.77	0.88	0.87	0.87
Prompting	0.81	0.74	0.69	0.76	0.65	0.79	0.75	0.77
KG	0.88	0.81	0.76	0.83	0.74	0.83	0.83	0.83
RAG	0.95	0.90	0.86	0.92	0.86	0.88	0.92	0.91
Entity Matching	0.92	0.84	0.77	0.84	0.73	0.88	0.84	0.84

Table 7: Performance Metrics for Different Methods.

The results demonstrate a clear hierarchy in performance among the evaluated approaches. The RAG-based technique emerges as the strongest performer, achieving an wF of 0.860 and a hF of 0.915, thereby indicating superior accuracy at the finest level of product classification as well as exceptional consistency across the entire category structure. Meanwhile, the fine-tuning-based model attains the second-best results (wF = 0.804, hF = 0.874), suggesting that direct

parameter adaptation with domain-specific data remains a reliable strategy to capture complex labeling schemes.

In contrast, the derived prompting framework yields a more modest outcomes (wF = 0.685, hF = 0.7634), illustrating that relying solely on LLM prompts is less effective in handling the nuanced hierarchy of product categories. Intermediate performances emerge from approaches that leverage knowledge graphs or entity matching techniques (e.g., KG: wF = 0.759, hF = 0.827; Entity Matching: wF = 0.771, hF = 0.839), indicating that while such strategies enhance the model's understanding of product categories, they do not fully achieve the performance levels observed with fine-tuning or RAG.

The relationship between the hierarchical and fine-grained metrics also provides meaningful insights. The higher hF scores, relative to wF, underscore that many "mispredictions" are actual "near-misses", often placing items within closely related categories that share the same parent node. This insight suggests that while some misclassifications occur, they often aren't severe in a hierarchical sense, reflecting that the model's understanding of category groupings is reasonably coherent even when it makes mistakes at the finest granularity.

Examining precision and recall patterns further refines these insights. The RAG model shows balanced precision and recall at Level 3 (0.882 and 0.861) and maintains this balance hierarchically (0.912 and 0.918), indicating comprehensive and accurate predictions. By contrast, finetuning favors precision (0.881) over recall (0.773) at Level 3, yet its hierarchical scores (hPrecision = 0.874, hRecall = 0.874) show that though the model's Level-3 recall lags behind its precision due to strict final-category matching, its balanced hierarchical scores indicate it often places instances on the correct branch of the taxonomy, reflecting partial but meaningful correctness. Prompting, on the other hand, struggles more with recall at Level 3 (0.6501), but the hierarchical perspective (hRecall = 0.753) shows its errors still lie within neighboring classes. The KG approach, while achieving moderate, maintains fairly balanced hierarchical precision (0.829) and recall (0.826), indicating it rarely veers far from the correct path. Similarly, Entity Matching achieves a respectable balanced hierarchical recall (0.838) and precision (0.840), reflecting that even when not entirely accurate at the deepest level, predictions remain close to the correct categories. These patterns highlight the value of hierarchical metrics and

confirm that significant portion of misclassifications remain near the correct category node.

6.2 Label Coherence

Model	Level 1	Level 2	Level 3
Fine-tuning	100.00	97.59	95.24
Prompting	96.82	96.82	96.82
KG	99.97	98.83	97.89
RAG	99.87	99.66	99.10
Entity Matching	100.00	100.00	100.00

Table 8: Label Coherence (%) of Models.

The comparison of label coherence across different techniques reveals consistently high performance at all hierarchical levels. Entity Matching achieves perfect coherence because it does not rely on label generation by the model; instead, it selects from existing labels based on LLM-driven entity matching. Other methods also maintain a high degree of coherence, even at the most granular classification levels, ranging from 95.24% (Fine-Tuning) to 99.10% (RAG). The strong coherence of Prompting (96.82%), KG (97.89%), and RAG (99.10%) at level 3 can be attributed to the presence of the target labels directly in the prompt, facilitating accurate label assignments. Notably, the Fine-Tuning model's 95.24% coherence, despite the prompt consisting solely of the product titles, demonstrates its ability to generate appropriate label names without additional cues.

It is important to note that any shortfall from full coherence directly reduces overall performance. For example, Fine-Tuning's 4.76 percentage-point gap from perfect coherence translates into a corresponding reduction in its performance metrics, reflecting the model's diminished capacity to generate entirely correct predictions, ultimately leaving critical predictions unlabeled—a costly inefficiency in any business-driven environment.

6.3 Misclassification

The misclassification analysis (Appendix - 1.1.5) reveals consistent challenges across all approaches, particularly for high-representation categories like "Printers & Scanners," which accounts for the most misclassifications in every method (e.g., 172 for Fine-Tuning, 260 for

Prompting, 220 for Knowledge Graph, and 320 for Entity Matching). Categories with overlapping characteristics, such as ”Computer Components” and ”TVs & Monitors,” also exhibit significant misclassification rates, indicating difficulties in distinguishing between closely related product types. Notably, approximately half of all misclassified data points were predicted incorrectly by at least two approaches, emphasizing the shared limitations and challenges across different methods.

Entity Matching, while effective for smaller categories, produces the highest misclassification rate in ”Printers & Scanners,” reflecting limitations in handling nuanced distinctions. Prompting and Fine-Tuning distribute errors more broadly but still struggle in high-frequency categories. In contrast, Knowledge Graph and RAG approaches show more evenly distributed misclassification patterns but still fail to handle ambiguous cases effectively.

These results highlight the impact of dataset imbalances and the inherent ambiguity in certain categories. This suggests that combining the strengths of these approaches or addressing class imbalances could further improve performance in hierarchical product classification.

Misclassification analysis at the leaf level (Depth 3) reveals significant challenges in distinguishing closely related categories within hierarchical product classification. For instance, *ink cartridges* were frequently misclassified as *printer ink refills* (110 instances), *ink sticks* (43), *inkjet printers* (41), and *photo printers* (30). These misclassifications suggest that overlapping titles and insufficient specificity in category definitions complicate accurate classification. Similarly, *notebooks* were often confused with *pcs/workstations* (59), *graphics cards* (35), and *tablets* (32), highlighting ambiguity in distinguishing between portable computing devices and related components. Other notable misclassifications include *mobile headsets* being assigned to *headphones* (38) and *internal hard drives* to *external hard drives* (30), which reflects overlap in product features that may cause confusion for both machine learning models and human annotators.

The approach-specific trends further underscore these challenges. Fine-tuning, while generally accurate, struggled with distinctions such as *mobile phone cases* misclassified as *peripheral device cases* (4.1% of errors), while prompting frequently misclassified *notebooks* as *pcs/workstations* (4.7%) or *tablets* (2.5%). Knowledge graphs improved structural consistency

but showed difficulties with overlapping hardware categories, such as *power supply units* misclassified as *power supplies* (3.2%). RAG demonstrated strong contextual understanding but often confused *mobile headsets* with *headphones* (3.3%). Entity matching faced challenges with rigid mappings, leading to frequent misclassifications of categories like *ink cartridges* that are part of overrepresented categories.

These findings suggest that certain adjustments to the taxonomy could improve classification accuracy. Categories with high misclassification rates, such as *ink cartridges* and *printer ink refills*, could be merged or redefined with clearer boundaries to reduce overlap. Similarly, categories like *notebooks* and *pcs/workstations* might benefit from additional intermediate subcategories that emphasize distinguishing features. For example, adding a subcategory that separates consumer-grade notebooks from professional workstations could aid in resolving ambiguity. Additionally, leveraging hierarchical relationships more effectively, such as requiring more granular context (e.g., intended use or technical specifications), could improve classification at the leaf level. These adjustments would not only improve LLM performance but also align the taxonomy with how humans intuitively group products, thus addressing inherent classification ambiguities.

6.4 Imbalanced Class Performance

6.4.1 Underrepresented Classes

Model	wP	wR	wF	hF	Level 1	Level 2	Level 3
Fine-tuning	0.92	0.47	0.46	0.68	0.85	0.62	0.47
Prompting	0.86	0.32	0.31	0.50	0.69	0.43	0.32
KG	0.91	0.38	0.39	0.56	0.69	0.51	0.38
RAG	**0.93**	0.39	0.40	0.59	0.75	0.50	0.39
Entity Matching	0.92	**0.70**	**0.70**	**0.81**	**0.88**	**0.80**	**0.70**

Table 9: Performance of methods in underrepresented categories.

Table 9 presents the evaluation metrics for a subset of results focused on underrepresented categories, where Entity Matching consistently outperformed the other methods. With an hF score of 81% and 70% for wR, wF, and level 3 accuracy, Entity Matching significantly outperformed finetuning, which achieved only 49% accuracy for leaf node categories. Other methods performed even worse, with scores below 40%.This difference may be attributed to the fact that

entity matching was the only method where the model was not explicitly prompted to categorize products. Instead, it was tasked with finding an entity match within a subset of similar titles, meaning that if the entity-match belongs to the underrepresented category, the prediction will inherently be correct. In contrast, methods relying on prompts might exhibit hesitancy in assigning products to underrepresented categories due to their niche and specific nature, which could make them appear less likely to the model.

6.4.2 Overrepresented Classes

Model	wP	wR	wF	hF	Level 1	Level 2	Level 3
Fine-tuning	**1.00**	0.93	0.96	0.97	**0.99**	0.96	0.93
Prompting	0.99	0.84	0.90	0.92	0.92	0.91	0.84
KG	0.99	0.96	0.97	0.98	**0.99**	0.98	0.96
RAG	**1.00**	**0.98**	**0.99**	**0.99**	**0.99**	**0.99**	**0.98**
Entity Matching	0.98	0.74	0.84	0.87	0.97	0.88	0.74

Table 10: Performance of methods in overrepresented categories.

In contrast to the performance observed in underrepresented categories, Table 10 highlights a notable difference in overrepresented categories, where entity matching performs significantly worse than the other models. While the fine-tuning, RAG, and KG-based methods consistently achieve metrics in the high 90s, and prompting scores in the high 80s to low 90s, entity matching lags behind with a wF score of 84 and an hF score of 87, demonstrating a clear underperformance. In overrepresented categories, where a larger variety of products with similar titles exist, the model may struggle to differentiate between subtle variations or incorrectly match to a common but incorrect entity. In contrast, methods like fine-tuning, RAG, and KG-based approaches leverage broader contextual understanding and hierarchical information, enabling them to generalize more effectively and accurately classify products within these categories.

6.5 Generalization

Fine Tuning performs strongly on WDC-222, achieving an remarkable *hF* of 0.87. On Avg Amazon, however, its *hF* dips to 0.81, showing a 7% decline. While still a reliable performer overall, this drop reveals its slight struggle with the more varied Amazon datasets. Prompting, on the other hand, delivers moderate results on WDC-222, with an *hF* of 0.76. Interestingly, it

Method	WDC-222		Avg. Amzn (Δ)		Amzn DE		Amzn EN		Amzn PL		Amzn TR	
	wF	hF	wF	hF	wF	hF	wF	hF	wF	hF	wF	hF
Fine-tuning	0.80	0.87	0.76	0.81 (-7%)	0.81	0.84	**0.81**	**0.86**	0.72	0.75	**0.76**	**0.77**
Prompting	0.69	0.76	**0.73**	0.81 (+7%)	0.78	0.87	0.70	0.80	0.72	0.81	0.63	0.77
KG	0.76	0.83	**0.79**	**0.82** (-%1)	**0.89**	**0.89**	0.73	0.81	**0.78**	**0.84**	0.74	0.74
RAG	**0.86**	**0.92**	0.75	0.78 (-15%)	0.81	0.84	**0.81**	0.84	0.71	0.75	0.67	0.68
Entity Matching	0.77	0.84	0.56	0.68 (-19%)	0.61	0.73	0.70	0.77	0.49	0.63	0.44	0.58

Table 11: Performance metrics across WDC-222 and Amazon datasets. The best performance per dataset is bold, while the overall best hF model performance is greyed out.

improves on Avg Amazon, where the *hF* rises to 0.81, marking a 7% increase. This highlights Prompt's strong generalization capabilities, as it performs better on the Amazon datasets than on WDC-222. Knowledge Graph stands out for its consistency. It achieves an *hF* of 0.83 on WDC-222 and maintains nearly the same level on Avg Amazon, where the *hF* is 0.82. With only a minor 1% decrease, this method proves its robustness and ability to adapt well across different datasets. RAG emerges as the best-performing method on WDC-222, achieving the highest *hF* of 0.92. However, this success does not carry over to the Avg Amazon datasets, where the *hF* drops sharply to 0.78, a significant 15% decline. While RAG excels on WDC-222, it struggles to generalize across the more diverse Amazon datasets. Entity Matching performs well on WDC-222, recording an *hF* of 0.84. In contrast, it experiences the most substantial drop on Avg Amazon, falling to an *hF* of 0.68, steep 19% decrease. This suggests that Entity Matching has difficulty adapting to the diversity present in the Amazon datasets.

In summary, KG and Prompting demonstrate the best generalization to the Amazon datasets, with minimal drops or even improvements in performance compared to WDC-222. In contrast, RAG and Entity Matching face the largest declines, highlighting challenges in handling the variability of Amazon data. Fine Tuning remains strong but sees a moderate drop. Additionally, DE and EN consistently outperform PL and TR across all methods, likely due to the LLMs' reliance on English training data, such as Mistral and the fact that the quality of the German titles seem to be better. Within those two categories KG and Prompting achieve their best results for DE, Fine Tuning, EM, and RAG perform highest for EN due to the fact that it employs an english focused re-ranker.

During the analysis, some model predictions seemed like "misclassifications." On closer inspection, they often corrected our initial labels. For instance, a product labeled as "Ink

Cartridges" was accurately refined to "Printer Ink Refill," highlighting both the models' ability to capture fine distinctions and the limitations of manual labeling, which led us to re-label parts of the data.

7 Discussion

7.1 Analysis of Key Findings

7.1.1 Strengths and Weaknesses of Approaches

Fine-tuning demonstrates high precision by leveraging sampling techniques and K-means clustering to balance overrepresented classes. This approach ensures a diverse set of centroids, enabling the model to capitalize on these features during training. As a result, it balances hR and hP, providing reliability and depth in its predictions. However, at level 3, the wR is moderate (0.74), indicating some overfitting on more common classes. Despite being trained on specific data, the model exhibits moderately strong generalization abilities, performing reasonably well on the Amazon dataset indicating successful fine-tuning on the task of hierarchical classification in the computers and electronics segment.

The multi-stage top-k implementation, explored as part of different prompting techniques, effectively captures the hierarchical structure without requiring pretraining or additional context. This is evidenced by the difference between hf and level 3 wF scores, highlighting the presence of numerous near misses as the method incrementally refines predictions at each hierarchical level. This technique significantly outperforms both flat and hierarchical baselines, including the highly parameterized LLaMA 70B, demonstrating its ability to enhance the performance of lower-parameterized models. However, its reliance on the pretrained knowledge of the LLM limits its overall performance, as reflected in the lowest hF scores among the approaches. Despite this, the technique exhibits strong generalization on unseen Amazon data by avoiding overfitting, making it highly relevant and applicable to diverse, unseen datasets across multiple languages.

The Knowledge Graph (KG) approach with KG-reasoning and voting mechanisms, provides

contextual knowledge in a hierarchical manner, enabling balanced predictions with minimal drop-offs across hierarchical levels. However, through error propagation and too general contextual information on node attributes, the approach demonstrates lower precision at level 3 (0.834). Additionally, its moderate recall is limited by its reliance on existing relationships reducing its ability to handle edge cases. The KG approach generalizes strongly across the Amazon dataset in various languages by focusing on important entities rather than the entire data, converting raw document information into a structured graph. This reduces overfitting and enhances its applicability to unseen data.

RAG leverages task-specific retrieval to ground the LLM, enhancing its predictions and achieving very strong performance across alln hierarchical levels. The combination of retrieval and generation ensures a strong balance between recall and precision at level 3, though its precision is slightly lower compared to finetuning due to occasional irrelevant context being introduced. RAG's generalization, however, is only moderate; the reliance on retrieving specific titles for predictions leads to overfitting, limiting its effectiveness on unseen data across various languages.

Entity matching achieves high precision at level 3 by minimizing false positives, particularly excelling in handling underrepresented classes. The prompt to match entities rather than categorize a title performs better in niche underrepresented categories. Its average recall is dependent on the quality of retrieval, which impacts performance on unseen data. Similar to RAG, entity matching exhibits moderate generalization but struggles with scenarios outside the training data, as the retrieval-based approach does not strongly enhance adaptability to new contexts and languages.

7.1.2 Complexity Analysis

Fine-tuning relies heavily on a subset of 30,000 observations sampled using clustering techniques from the entire dataset, while focusing on finetuning the model to the training data. While the training process is computationally intensive and time-consuming, the resulting model is highly efficient during inference, with fast response times, minimal resource usage, and category-only outputs, limiting token costs. This efficiency makes it suitable for domain-

specific applications. However, the rigidity of this approach is a drawback, as adapting to a new taxonomy or changes in the category structure requires retraining the model, which is both time and resource-intensive.

The multi-stage top-k prompting approach, in contrast, does not require any training data, making it a plug-and-play solution with strong generalization capabilities. By iteratively refining predictions at each level of the hierarchy, this technique leverages the taxonomy structure effectively to enhance classification accuracy without the need for pretraining or additional fine-tuning. It is the least complex of all approaches, requiring a few LLM inferences per observation when employing a local classifier per hierarchical level. Outputs are concise, consisting solely of the predicted category, which ensures token efficiency and eliminates the need for post-processing. Its low input token requirements further enhance its simplicity and reduces costs. Moreover, the setup for adapting to a new taxonomy or updating the hierarchy is instant, requiring no additional training or restructuring.

The Knowledge Graph (KG) approach has minimal training data requirements, needing only a few observations per category to build the graph and load attributes. It shares similarities with prompting in this regard but diverges significantly in complexity. The KG approach involves an inference process that, while offering greater interpretability through reasoning, is less efficient. This inefficiency is amplified by high input token counts, the use of Local Classifier Per Node (LCPN), and the implementation of a voting mechanism, resulting in at least nine LLM inferences per observation. While it provides significant value in scenarios requiring interpretability, its application to tasks such as product categorization may depend on specific use cases and requirements. However, adapting to a new taxonomy is straightforward, as it only requires creating a new KG from the category structure before inference.

Retrieval-Augmented Generation (RAG) is the most reliant on strong training data to ensure effective performance. Its inference outputs are limited to category predictions, but the process is complex and resource-intensive. High input token counts stem from the inclusion of 10 examples per call. While only one LLM inference is made per observation, this process is coupled with proprietary reranking mechanisms. The KG creation, vector database construction, and reranking process add significant computational overhead, resulting in high inference

costs. Adapting to a new taxonomy or making updates to it is similarly complex, requiring re-vectorization of the database, rebuilding of the KG, and reranking before inference.

Entity Matching shares similarities with RAG in its reliance on robust training data but offers a simpler and more streamlined approach. It outputs only the matched title, ensuring 100% label coherence, with average input token counts and a straightforward inference process. The method utilizes a basic vector database and requires just one LLM inference per observation, making it computationally efficient. Adapting to a new taxonomy is also relatively simple, involving only vectorization before proceeding to inference.

In the dynamic world of taxonomies, prompting emerges as the most adaptable approach, followed by the KG and Entity Matching, while finetuning and RAG offer higher accuracy at the cost of increased complexity, reduced generalizability, and greater rigidity.

7.2 Implications

7.2.1 Theoretical Implications

Our results confirm that LLMs can be rapidly adapted to hierarchical classification tasks with satisfactory accuracy. Methods like Prompting and RAG can be easily tested, while fine-tuning enables models to integrate hierarchical knowledge and reuse it during prediction. This supports our hypothesis that generative LLMs, combined with methods fostering effective generation (KG, RAG, Entity Matching), are well-suited for hierarchical classification tasks, often improving label correctness. Moreover, our solutions establish new benchmarks in the E-Commerce and Hierarchical Text Classification domains using WDC-222 with weak LLMs. Although methods suggested mostly do not surpass performance of strong LLM-based (RAG-based) techniques proposed by Gholamian et al. (2024). Implementing these approaches with powerful LLMs, such as GPT-4, has the potential to achieve similar or even higher performance levels.

The conclusions stem from examining five well-structured methodologies involving LLMs. Future improvements may arise from merging these approaches. We outline three promising directions:

Advanced Fine-Tuned Models Implementation: These models inherently maintain proper

hierarchical outputs from minimal inputs, even without in-context learning. Pairing them with prompting and in-context techniques (e.g., KG, RAG) may yield even better results. Since fine-tuned models often lose in-context learning capabilities, in-context-focused, specialized fine-tuning methods like the two-step ProMoT framework (Wang et al., 2024b) for maintaining generalization abilities of fine-tuned models. In case of combining fine-tuned model with prompting, Dual Model Approach by Cheng et al. (2024) may lead to improved performance thanks to using fine-tuned model's output as hint for a prompt-based LLM technique.

Class-dependent method selection: Some techniques tend to perform better on certain types of classes. For instance, Entity Matching methods excel at handling underrepresented classes. A custom mechanism could retrieve few-shot examples (based on retrieval methods discussed in Part D) from vectorstores, determine if underrepresented classes dominate these examples, and then apply Entity Matching when needed. Otherwise, other highly efficient methods would apply.

Enhanced Knowledge Graph Retrieval: The Knowledge Graph (Part C) approach can be improved by combining it with retrieval techniques suggested in (Part D). Based on categories returned by the retrieval step, knowledge graph filtering mechanism can be developed (or inherited from the GraphRAG framework described in Paper D), in order to more efficiently provide partial Sub-Graphs as context for the in-context learning approaches, hence, potentially improving the classification performance while reducing prompt size.

7.2.2 Practical Implications

We offer businesses a cost-effective solution for categorizing products across diverse datasets while achieving competitive, state-of-the-art results. Although closed-source LLMs like GPT-4 may deliver better performance, our research demonstrates a viable and affordable alternative through open-source models such as Llama, Mistral, and Google Flan T5. To illustrate the cost difference, let's consider categorizing 100,000 products into a specific taxonomy. Using GPT-4 (8k), each call would process approximately 100 tokens (50 input, 50 output), with an average cost of $0.0045 per call. This amounts to a total cost of $450 for 100,000 API calls. In comparison, our approach leverages open-source models, where the only cost involved is

the computational infrastructure, such as a Colab Pro subscription or equivalent IDE services, roughly priced between $10 and $50/month. This significant cost disparity highlights the practicality of open-source solutions. Businesses with budget constraints can achieve robust classification results at a fraction of the cost, making these alternatives particularly suitable for small to medium-sized enterprises. While closed LLMs may excel in scenarios requiring maximum accuracy, open-source models offer scalable and cost-efficient performance.

Further, we can provide tailored recommendations for businesses on when to use specific LLM methods, depending on their unique requirements and constraints.

If time is a constraint or attaining training data is costly, we recommend using multi-stage top-k prompting. This approach offers a quick and flexible solution that does not require significant computational resources or training time. This makes it ideal for businesses that need rapid deployment or are working with frequently changing taxonomies.

If the company works with a large dataset of a specific taxonomy, we propose fine-tuning a model tailored to that taxonomy. Fine-tuning achieves high accuracy by adapting the model to domain-specific data, making it the best choice for businesses operating in specialized industries with well-defined category structures, such as fashion or automotive.

If the company deals with many diverse datasets, we suggest using the Knowledge Graph (KG) framework. KG demonstrates excellent generalization capabilities, making it suitable for businesses managing product catalogs with varying structures or those operating across different markets and languages.

If access to a domain-specific unstructured knowledge source is available, we recommend exploring RAG (Retrieval-Augmented Generation). RAG is particularly effective for scenarios involving incomplete or noisy datasets (like WDC-222), as it uses retrieval techniques to enhance accuracy. This is valuable for businesses handling unstructured product descriptions or legacy data systems.

By aligning model selection with specific business needs, companies can maximize the efficiency and accuracy of their product categorization workflows while minimizing costs and resource requirements. Our analysis demonstrates that open-source models, when implemented effectively, can be a cost-effective alternative for various e-commerce and data management

challenges.

7.3 Limitations

This study has several limitations that affect its generalizability and scalability. The focus on the ”Computers & Electronics” domain and the use of only one taxonomy, narrows the applicability of the results to other product categories. The work was only generalized on products in the same domain and it was conducted on only 30 manually selected instances. Furthermore, since WDC was sourced from Icecat, the overlap between the WDC and Icecat datasets creates challenges for evaluating methods, as shared information may artificially boost the performance of retrieval-based approaches like RAG, reducing the reliability of comparisons. Dataset imbalances and overlapping subcategories further complicate classification, leading to reduced accuracy across all methods.

Weak learners were used and tested in different frameworks, including Ollama and Huggingface, which may have introduced inconsistencies in performance due to differences in model capabilities. Computational constraints also posed significant challenges. Fine-tuning requires retraining when taxonomies change, making it resource-intensive and impractical for dynamic or large-scale systems. Knowledge Integration methods like RAG and Knowledge Graph are computationally expensive, with high token costs and long inference times, limiting their scalability. While prompting-based techniques are more flexible, they struggle to handle the complexity of deep hierarchies without additional training or fine-tuning.

The hierarchical structure of the datasets, combined with shared information between training and testing sets, raises questions about the robustness of the results. Additionally, the reliance on product titles alone, while practical, may overlook valuable information in attributes like their descriptions. This highlights the need for further research on integrating richer data sources to improve classification accuracy and adaptability.

Improving accuracy is critically important to ensure real-world applicability, as low accuracy can result in significant misclassification, particularly in larger datasets. However, even with improved accuracy, scalability remains a challenge due to issues with label coherence. Failing to assign labels to certain data points necessitates manual labeling, which is not a scalable

solution.

7.4 Future Work

Future work should address the limitations identified in this study while exploring new directions for advancing hierarchical classification with large language models. Integration of the different explored methodologies that combine the strengths of existing approaches, such as fine-tuning, prompting, and knowledge graph-based reasoning can further enhance product categorization.

The current study's focus on computers and electronics domain and uses product titles for categorization. Therefore, future research should test the proposed methods on broader and more diverse domains such as fashion, healthcare, or industrial equipment. There is also potential to explore different taxonomies within the same domain. Furthermore, including richer data like descriptions, specifications, and customer reviews, as this can improve accuracy, especially for underrepresented and rare cases. Incorporating multimodal data, such as images or videos, can significantly enhance classification, not only in areas where text alone falls short but also when used in combination with textual data to improve performance.

Another promising direction is developing class dependent method selection mechanisms. Techniques like entity matching work well for underrepresented classes, while others are better suited for common categories. An adaptive pipeline that chooses methods based on class characteristics could greatly improve performance. A relevant improvement can be done regards, improving label coherence by exploring dynamic vocabulary restriction techniques, which limit the label space contextually during prediction, enhancing precision. Developing more robust training objectives or loss functions for fine-tuning could address the challenge of ambiguous or inconsistent labeling, a key factor for improving scalability.

Cost-effective approaches, such as multi-stage top-k prompting, can be refined for complex hierarchies. Fine-tuning methods could be enhanced for incremental updates, enabling models to adapt to changing taxonomies without full retraining. Future research may focus on reducing token costs and computational demands through strategies like pruning retrieval examples. The interaction between generative and discriminative approaches in hierarchical classification, rep-

resent an other area of research, generative LLMs could complement traditional models through techniques like knowledge distillation or dual-model frameworks. Overall, these advancements aim to create scalable, adaptable, and cost-efficient systems for diverse taxonomies, languages, and data sources, addressing real-world challenges effectively.

8 Conclusion

This study investigated how various techniques can enhance hierarchical product classification using LLMs. Our findings demonstrate that LLMs have significant potential to improve HPC, leveraging their advanced natural language processing capabilities to address challenges such as imbalanced taxonomies, and diverse datasets. While the performance is model-dependent and comes with certain limitations, such as moderate recall in entity matching or computational overhead in RAG, the approaches explored already offer practical implications for real-world applications.

Fine-tuning with hierarchical information proved effective for domain-specific classifications, achieving a strong balance between recall and precision, albeit with some limitations at deeper hierarchical levels. Multi-stage prompting emerged as a scalable and cost-effective solution, excelling in adaptability and generalization without requiring pretraining or large datasets. However, its performance remains inferior to advanced methods like fine-tuning or RAG when high precision is essential. Knowledge Graphs added value through structured contextual reasoning, balancing predictions and generalizing well to unseen data, although precision at deeper levels was sometimes affected by error propagation. RAG excelled in performing HPC on WDC but encountered limitations in generalization due to reliance on specific examples. Finally, entity matching excelled in precision for underrepresented classes, making it an ideal choice for stable and monolingual taxonomies, though its generalization was limited by retrieval quality.

The results confirm that the different explored techniques improve HPC using LLMs. The developed frameworks provide viable alternatives to traditional supervised methods, achieving comparable results with reduced training data requirements. RAG and Fine Tuning managed

to beat the supervised benchmark of *0.85 hF* set by Nils Richter and Christian Bizer (n.d.). By strategically employing the developed methods, businesses can implement adaptable and cost-efficient classification systems tailored to their needs. Future advancements, such as integrating these methodologies and further developing them hold promise for continued improvements in HPC with LLMs. These insights pave the way for scalable and robust solutions to meet the demands of an increasingly dynamic and competitive marketplace.

Paper A: Fine-Tuning Decoder-Only Language Models for Hierarchical E-Commerce Product Classification: A Causal Language Modeling Approach

A - 1 Introduction

Adapting general pre-trained language models (LLMs) to domain-specific or specialized tasks is essential for unlocking their full potential. Unlike techniques such as prompt engineering or retrieval-augmented generation (RAG), fine-tuning directly updates the Large Language Models' weights, aligning the model to a given application. While traditionally computationally expensive, recent advances, such as Parameter-Efficient Fine-Tuning (PEFT) methods like LoRA (Low-Rank Adaptation), have significantly reduced the cost and time of fine-tuning while preserving performance. This makes fine-tuning more accessible for tasks requiring deep specialization, such as hierarchical product classification (HPC).

Decoder-only LLMs, with their autoregressive causal language modeling (CausalLM) nature, are particularly promising for hierarchical classification tasks. Their step-by-step text generation aligns well with hierarchical structures, where intermediate levels can serve as implicit cues for the final prediction. This recursive process allows models to "self-generate hints" for earlier levels before making more granular predictions, effectively mimicking a step-by-step reasoning approach.

This study investigates whether fine-tuning decoder-only LLMs with hierarchical context enables models to memorize, understand, and accurately predict hierarchical classes. The research is guided by the following questions: (1) Does hierarchical context during fine-tuning improve the model's ability to predict correct hierarchical paths?; (2) Does hierarchical context help the model build an understanding of the hierarchy and memorize label names?; (3) Is causal modeling a suitable approach for fine-tuning decoder-only LLMs for hierarchical text classification in the e-commerce context?

This paper evaluates various fine-tuning strategies on decoder-only LLMs of different sizes,

focusing on hierarchical e-commerce product classification. By comparing hierarchical and flat fine-tuning approaches, the study provides insights into the advantages of contextualized hierarchical fine-tuning, particularly in improving label coherence and prediction accuracy across multiple levels.

A - 2 Background and related work

A - 2.1 LLMs and Hierarchical Classification

The task of hierarchical classification (HTC) has been widely explored with the use of language models. Research about implementing various text encoding strategies with the use of novel and complex has been saturated over the last decade. Beyond previously mentioned encoder-based (Chen et al., 2021; Wang et al., 2022a,b) . This saturation in the field, pivoted researchers' focus into different paradigm of hierarchical classification — namely, as a sequence-to-sequence language modeling problem, leading to multiple findings employing encoder-decoder methods (Risch et al., 2020; Torba et al., 2024; Jain et al., 2024).

A - 2.1.1 Decoder-Only LLM approaches

While encoder-decoder approaches established a strong foundation for hierarchical text classification as a seq2seq task, they often required architectural modifications or complex mechanisms. The rise of large language models (LLMs), such as the GPT family, introduced a new paradigm. Decoder-only LLMs can achieve strong generalization through in-context learning alone, without altering the model's architecture. For example, Chen et al. (2024b) demonstrated that fine-tuning retrieval-augmented techniques with decoder-only LLMs enables state-of-the-art results for few-shot hierarchical classification. Continued pre-training or fine-tuning further enhances these models' applicability.

Decoder-only models have also been tested without external augmentation. Chen et al. (2024c) applied GPT-3.5 and GPT-4 for biomedical hierarchical text classification using prompting techniques alone. Although they achieved competitive results, these models did not surpass simpler, task-specific classifiers. Similarly, Gholamian et al. (2024) explored hierarchical prod-

uct classification in e-commerce using the WDC-222 dataset. Their study, focusing on model robustness under dataset distortions, proposed three in-context learning techniques for LLaMa-2 (70B), GPT-3.5, and GPT-4, including one with RAG integration. Their findings set a relevant benchmark for this paper.

In the context of e-commerce, Ma et al. (2023) introduced EcomGPT-CT, a BLOOM-7B model fine-tuned on e-commerce data, including product descriptions and reviews. They demonstrated the benefits of in-domain continued pre-training for e-commerce tasks, including classification with in-context learning. However, their work did not explore task-specific fine-tuning or evaluate performance on hierarchical product classification datasets.

Although LLMs can be easily adapted to complex task in various domains, prompting techniques might not be sufficient to solve the task efficiently, hence, more advanced model adaptation should be considered. The research landscape is very limited in the context of continued pre-training of decoder-only LLMs for hierarchical classification task, especially in the context of e-commerce.

A - 2.2 Fine-Tuning LLMs

LLMs are trained on extensive general-purpose datasets, equipping them with advanced language comprehension and reasoning abilities. State-of-the-art models like GPT-4, Claude-3, and Gemini can solve complex tasks through natural language prompts alone (Yang et al., 2023). However, tasks requiring specialized domain knowledge or complex context often exceed the capabilities of in-context learning. Fine-Tuning involves further training of a PLM on small and specific dataset to boost its performance on specific task (Parthasarathy et al., 2024). Privacy concerns and computational constraints further lead enterprises to favor weaker LLMs over advanced ones (MathavRaj et al., 2024). In such cases, Fine-Tuning on domain-specific data significantly improves performance. Empirical studies, such as those in phishing detection, show that fine-tuning both strong and weak LLMs outperforms prompt engineering, particularly for specialized tasks (Trad and Chehab, 2024).

In Large Language Models, fine-tuning typically follows two main approaches: Supervised Fine-Tuning, which uses labeled input-output pairs, and Unsupervised Fine-Tuning, which

adapts the model to domain-specific text without labels. Another distinction involves fine-tuning on smaller, task-specific datasets versus continued pre-training adapting modelk to domain-specific corpora (Parthasarathy et al., 2024). For example, enhancing performance on an e-commerce classification task may involve continued pre-training on e-commerce data, then fine-tuning for the specific classification objective.

A - 2.2.1 Fine-Tuning Process

The fine-tuning process by Parthasarathy et al. (2024) comprises several key steps. First, data preparation involves cleaning, splitting, and generating input-output pairs. Choice of PLM, training setup, hyperparameters, objective and loss function follows. Fine-tuning is then performed, through Full Fine-Tuning or Parameter-Efficient Fine-Tuning methods. Finally, the model is evaluated using carefully chosen metrics to assess its performance.

Zhang et al. (2024a), in their review of LLM Instruction Tuning Pipeline, outline the format for an instruction prompt used in the fine-tuning process, consisting of 3 key components: Instruction, Input, and Desired Output. They emphasize the importance of selecting high-quality examples for fine-tuning and employing balanced sampling strategies, as demonstrated in datasets like the OpenAssistant Conversations dataset (Köpf et al., 2023). Within fine-tuning, multiple advanced input techniques are explored, including Hard-Prompt and Soft-Prompt tuning. Soft prompts, which leverage continuous feature vectors, have been shown to outperform both traditional fine-tuning and hard prompts, particularly in tasks like clinical concept extraction. However, soft prompts suffer from limited interpretability and lack transferability across different models (Wen et al., 2023). Conversely, hard prompts, which consist of interpretable words, can be optimized using gradient-based methods for both text-to-image and text-to-text applications (Wen et al., 2023).

In the context of HTC, the paradigm of curriculum learning—which assumes a multi-step, sequential training process—has shown significant promise (Bengio et al., 2009). Since hierarchical classes can be conceptualized as a series of classification steps, each level of classification can be trained independently, with deeper levels of training building on the outcomes of preceding levels. The underlying assumption is that knowledge acquired during lower-level

classification is reused in subsequent levels, resulting in a more robust training process at the final classification stages. Curriculum learning has been successfully applied to hierarchical text classification tasks, such as medication recommendation systems (Sun et al., 2023).

A - 2.2.2 Parameter-Efficient Fine-Tuning (PEFT)

Immense computational and resource demands of LLM fine-tuning necessitate more efficient approaches to model adaptation (Zhou et al., 2024; Hu et al., 2021; Dettmers et al., 2023). Efficient fine-tuning of LLMs often leverages both parameter-efficient techniques and quantization to reduce memory, computational costs, and training times. Parameter-Efficient Fine-Tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA) (Hu et al., 2021), introduce a small set of trainable parameters that integrate into the model's layers, significantly cutting down the resources needed compared to full-model fine-tuning. LoRA involves hyperparameters like the rank parameter (r), which determines the size of the low-rank adaptation matrices; LoRA alpha, a scaling factor that governs the contribution of the LoRA parameters relative to the base model parameters; and LoRA dropout, which can help regularize the adaptation matrices by randomly dropping some components during training. Furthermore, LoRA allows targeting specific modules within the model's architecture (e.g., attention layers or feed-forward layers) for adaptation, enabling fine-grained control over which parts of the network receive additional learnable parameters.

Building upon this, QLoRA (Dettmers et al., 2023) combines LoRA with quantization to achieve even greater efficiency by representing model parameters with fewer bits—commonly 4-bit precision—instead of the standard 16- or 32-bit floating-point format. Quantization entails lowering the numerical precision of weights and sometimes activations, thereby reducing their storage requirements and accelerating computations on compatible hardware Zhou et al. (2024). Although decreasing precision can introduce some approximation errors, carefully designed quantization strategies can preserve model quality. By integrating these approaches—parameter-efficient adaptation through LoRA and lower-precision parameter representation via QLoRA - researchers and practitioners can fine-tune very large models on limited hardware resources without sacrificing performance.

A - 3 Methodology

A - 3.1 Training Data

Icecat dataset was used as training data for fine-tuning. Due to limitation in availability of computational resources, it was neccessary to resample the Icecat training dataset to a smaller subset. Additionally, evidence shows that training multi-class classifiers is often not the best with using natural distribution of classes (Weiss and Provost, 2001), hence, more balanced dataset can lead to better modeling results.

Due to computational limitations, the dataset was resampled to a smaller, more balanced subset. Training multi-class classifiers on natural class distributions often leads to suboptimal performance (Weiss and Provost, 2001), and balancing the data improves results. A variant of cluster-based under-sampling (Yen and Lee, 2009; SUN et al., 2009) was applied, as it preserves the variance and structure of the original feature space while ensuring class balance. Unlike random under-sampling, which risks discarding informative samples and ignoring dataset's variance, cluster-based methods retain representative examples evenly spread across each class.

In implementation, product titles and descriptions were converted into vector embeddings using the sentence-transformer model *all-miniLM-L6-v2*, chosen for its efficiency and semantic accuracy. The 482,433 training records were processed in batches of 128, completing embedding generation in approximately 16 minutes. For Level 3 classes, a maximum threshold of 100 samples was imposed; classes with fewer examples remained untouched. K-Means clustering (k=100, random state=42) was applied to the embeddings within each class, and data points nearest to cluster centroids were retained. This reduced the dataset to 28,769 samples, with each Level 3 class containing up to 100 representative and diverse examples. The clustering step, excluding embedding generation, took just over 2 minutes.

The resampled dataset was split into training (80%), validation (10%), and test (10%) sets using stratified sampling to preserve class distributions. To reduce computational costs, inputs were limited to product titles, as they alone provide sufficient context for classification (Dai et al., 2020) also mitigating the quadratic scaling of attention mechanisms complexity with input sequence length (Vaswani et al., 2023).

A - 3.2 Models Selection

This experiment aimed to evaluate hierarchical e-commerce product classification using light weight large language models suitable for a single computational unit. Five model sizes were selected to represent diverse architectures and training backgrounds (see Appendix - 2.1). While Pythia models (Pythia-70m and Pythia-410m) provide accessible checkpoints, their compatibility with parameter-efficient fine-tuning frameworks was limited. To address this and include industry-relevant benchmarks, we added LLaMa 3.2 (1B and 3B) and Mistral-7B-v0.3 models, evaluating both base and instruct variants. This selection highlights the performance of smaller Pythia models while demonstrating the potential of widely adopted open-source LLMs like LLaMa and Mistral for specialized classification tasks.

A - 3.3 Fine-Tuning Variants

Each model is trained with all fine-tuning variants, with performance evaluated using relevant metrics. The fine-tuning prompts remain consistent for all methods, varying only in the inclusion of variant-specific special tokens, such as the EOS and Padding tokens. For each scenario, additional special tokens are added to the tokenizer, with corresponding embeddings reshaped. The special tokens include: `[SEP]` (separator for input and output), `[L1]`, `[L2]`, and `[L3]` (start of each level's category name), and `[LVL_SEP]` (separating tokens for class levels). Detailed prompt templates and examples for each method are provided in Appendix - 2.2.

A - 3.3.1 Flat Fine-Tuning

Flat Fine-Tuning involves training the model to predict the third-level category directly from the input title without considering intermediate hierarchical levels. It is performed exclusively using base models; therefore, the text following the [SEP] special token represents only a single level 3 phrase, followed by an [EOS] token.

Hierarchical Fine-Tuning extends the flat approach by incorporating the hierarchical structure of the classification task. This method is also applied solely to base models and is conducted over three epochs. The prompt template used in hierarchical fine-tuning includes special tokens that delineate each level of the hierarchy: [L1] for Level 1, [L2] for Level 2, and [L3] for

Level 3, separated by [LVL_SEP] tokens. This structured prompting allows the model to generate category names at each hierarchical level sequentially, thereby leveraging the hierarchical relationships inherent in the data. The rationale behind this approach is to enhance the model's ability to understand and utilize the hierarchical dependencies between different classification levels, leading to more accurate and contextually appropriate categorizations.

Instruct Hierarchical Fine-Tuning method uses Instruct models from LLaMa and Mistral, based models from Pythia. It spans three epochs and employs a more elaborate prompt template that includes explicit instructions for the classification task. The instruction guides the model to classify a given product title into a hierarchical category by specifying the format and structure of the expected output, including the use of special tokens [SEP], [L1], [L2], [L3], and [LVL_SEP]. The inclusion of instructional text aims to leverage the strengths of instruct models in following detailed directives, thereby improving the precision and reliability of hierarchical classifications.

Curriculum Learning Fine-Tuning adopts a multi-step, sequential training process to progressively train the model on each level of the hierarchical classification. This approach is exclusively applied to base models and involves a total of nine epochs, divided into three epochs for each classification level. Initially, the model is fine-tuned to predict the Level 1 category, followed by Level 2, and finally Level 3. At each stage, the prompt template is adjusted to include the relevant level tokens and class separators, enabling the model to build upon the knowledge acquired in previous stages. For example, during Level 2 training, the model not only learns to predict the Level 2 category but also retains the Level 1 classification by calculating the loss for both levels simultaneously. This method ensures that the model reinforces its understanding of lower-level classifications while acquiring expertise in higher-level distinctions, resulting in a more robust and coherent hierarchical classification capability.

A - 3.4 Training Objective and Training Steup

A - 3.4.1 Causal Language Modeling (CLM)

To train the models for hierarchical classification, we employed a causal language modeling (CLM) approach as the foundational objective. In this paradigm, the model is tasked with pre-

dicting the next token in a sequence based on all previously seen tokens, thereby learning an autoregressive representation of the data (Radford et al., 2019). This autoregressive property is particularly advantageous for hierarchical sequences: as the model predicts each successive token, it relies on previously generated tokens—such as those indicating higher-level categories—mirroring the step-by-step inference process used to navigate a taxonomy of product categories. Each classification level is generated conditionally, enabling the model to produce a coherent hierarchical class structure that is sensitive to previously predicted levels.

To optimize the model parameters, we employ a cross-entropy loss function, which measures the divergence between the model's predicted probability distribution over possible next tokens and the actual target token distribution. This loss function encourages the model to assign higher probabilities to correct class labels at each hierarchical step and penalizes it for assigning probability mass to incorrect or irrelevant tokens. Consequently, the model iteratively refines its understanding of both product titles and their associated hierarchical labels, enhancing its ability to predict the correct sequence of categories over time.

A - 3.4.2 Input Data Processing for Training

For each task, an appropriate prompt template is selected, and training examples are created by populating the prompt template with relevant variables from the dataset. Subsequently, each training example is tokenized using the tokenizer corresponding to the model being trained, thereby generating the input_ids and attention masks for the training and validation datasets. Additionally, for each input, token labels are created using the following logic: all tokens before and including the [SEP] token are labeled as -100, while the remaining tokens retain their original labels. This approach ensures that the loss function is calculated exclusively for the generated tokens. Padding tokens are then added to unify the length of input sequences for model training; for Parameter Efficient Fine-Tuning, right padding and left truncation are performed. The maximum input sequence length for all models was set to 256 tokens.

A - 3.4.3 Training Setup & Infrastructure

For Pythia Models, due to their relatively small size, the selected Pythia models are trained using Full Fine-Tuning with the default HuggingFace Trainer. LLaMa and Mistral Models: For these models, we utilize resources provided by the authors of the Unsloth Library for Python (Daniel Han and team, 2023), which facilitates Parameter Efficient Fine-Tuning of Quantized LLMs (QLORA) without performance loss, potentially doubling training speed. Unsloth provides quantized versions of all LLaMa and Mistral models (both base and instruct models), as well as the UnslothTrainer class, which replaces the Trainer class from Hugging Face.

Training and inference were conducted using Google Colab Pro+, a cloud service hosting Python environments tailored for data science purposes. The Pro+ subscription provides access to GPUs such as T4 (16GB VRAM), L4 (22GB VRAM), and A100 (40GB VRAM), which are essential for training large language models.

Training arguments varied slightly depending on the model and GPU used in the experimental setup. For models trained with LoRA, the LoRA parameters also differ based on model size (LLaMa 1B, LLaMa 3B, and Mistral 7B). To maintain relatively low training time, for larger models, the r parameter was decreased alongside the lora_alpha parameter to maintain a consistent ratio across models. Model Paths (from HuggingFace Hub), Detailed Training Arguments and Detailed LoRA Parameters are specified in Appendix Appendix - 2.3.

A - 4 Results

The training process including 5 different pre-trained large language models representing different sizes and architectures, combined with 4 different fine-tuning techniques oriented on causal language modeling for hierarchical classification resulted in fine-tuning and evaluating 20 specialized large language models. The predictions for test data and WDC-222 were performed for each model with greedy search strategy that unlike Sampling-based or Beam search methods, selects the highest probability token at each step. Table 12 provides performance metrics on both WDC-222 and isolated test dataset from resampled subset of original Icecat data (with balanced class distribution). It also contains the label coherence metrics at each level of

classification (not applicable for level 1 and level 2 of Flat Classification Fine-Tuning), and additionally details on training and evaluation such as type of GPU used and time of computation.

Pre-Trained Model	Fine-Tuning Type	Icecat Test Data Metrics				WDC Evaluation Metrics				WDC Label Coherence			Training and Inference Time			
		wR	wP	wF	hF	wR	wP	wF	hF	Level 1	Level 2	Level 3	Training GPU	Training Time	Inference GPU	Inference Time
Pythia 70m	Flat Completion	0.849	0.873	0.853	0.885	0.272	0.621	0.324	0.400	x	x	50.90%	T4	35 min 35s	T4	1min 14s
	Hierarchical Completion	0.844	0.858	0.844	0.893	0.312	**0.703**	**0.366**	**0.559**	100.00%	88.10%	79.32%	T4	34min 37s	T4	1min 21s
	Hierarchical Instruct	0.801	0.838	0.805	0.863	0.262	0.665	0.307	0.544	99.97%	90.11%	**80.66%**	T4	26min 11s	T4	1min 32s
	Curriculum Learning	0.818	0.849	0.821	0.867	**0.464**	0.471	0.206	0.468	94.47%	76.41%	56.40%	T4	37min 30s	T4	1min 30s
Pythia 410m	Flat Completion	0.890	0.909	0.891	0.922	0.458	0.761	0.510	0.587	x	x	**91.22%**	T4	3h 33min	T4	6min 57s
	Hierarchical Completion	0.844	0.858	0.844	0.884	0.312	0.703	0.366	0.517	100.00%	91.02%	80.06%	T4	2h 22min	T4	7min 17s
	Hierarchical Instruct	0.899	0.909	0.899	0.934	**0.521**	**0.803**	**0.580**	**0.694**	99.97%	91.35%	83.48%	T4	3h 11min	T4	6min 47s
	Curriculum Learning	0.868	0.894	0.870	0.911	0.447	0.737	0.496	0.627	93.73%	86.66%	77.68%	L4	3h 55min	T4	9min 40s
LLaMa 3.2 1B	Flat Completion	0.846	0.891	0.856	0.895	**0.598**	**0.837**	**0.652**	0.696	x	x	**79.22%**	A100	54min 43s	A100	38s
	Hierarchical Completion	0.813	0.881	0.829	0.855	0.543	0.808	0.594	**0.746**	98.22%	83.58%	75.13%	A100	54 min 34s	L4	2min 43s
	Hierarchical Instruct	0.615	0.798	0.663	0.740	0.404	0.735	0.475	0.681	96.98%	86.46%	65.68%	A100	41min 57s	T4	5min 26s
	Curriculum Learning	0.249	0.516	0.306	0.417	0.076	0.401	0.112	0.184	8.55%	8.51%	10.05%	A100	49min 15s	L4	3min 11s
LLaMa 3.2 3B	Flat Completion	0.864	0.906	0.873	0.907	0.671	**0.877**	0.724	0.783	x	x	84.92%	A100	1h 36min	L4	5min 24s
	Hierarchical Completion	0.858	0.895	0.865	0.908	**0.705**	0.870	**0.747**	**0.837**	100.00%	91.42%	**87.06%**	A100	1h 36min	L4	7min 38s
	Hierarchical Instruct	0.508	0.843	0.599	0.682	0.311	0.806	0.397	0.614	96.98%	86.46%	65.68%	A100	1h 39min	L4	8min 31s
	Curriculum Learning	0.797	0.860	0.809	0.855	0.655	0.857	0.717	0.822	99.40%	92.06%	83.31%	A100	1h 58min	L4	9min 5s
Mistral 7B v0.3	Flat Completion	0.900	0.916	0.903	0.935	0.721	**0.882**	0.766	0.825	x	x	88.10%	A100	2h 55min	L4	11min 31s
	Hierarchical Completion	0.892	0.906	0.894	0.934	**0.774**	0.880	**0.804**	**0.874**	100.00%	97.59%	95.24%	A100	2h 50min	L4	18min 11s
	Hierarchical Instruct	0.872	0.896	0.876	0.921	0.758	**0.882**	0.798	0.867	100.00%	94.57%	**100.00%**	A100	2h 40min	A100	3min 48s
	Curriculum Learning	0.888	0.900	0.889	0.931	0.770	0.879	0.801	0.864	100.00%	97.39%	93.54%	A100	4h 9min	A100	4min 17s

Table 12: Research Results for Pre-Trained Models with Different Fine-Tuning Variants

A - 4.1 The Best Performing Model

Across the tested models, Mistral 7B v0.3 consistently achieved top results in all hierarchical fine-tuning strategies, surpassing smaller models in both weighted (wR, wP, wF) and hierarchical (hF) metrics. Among its hierarchical variants, Curriculum Learning was conceptually promising but did not outperform others, with hF=0.86 and wF=0.80. Still, Mistral 7B notably benefited from Curriculum Learning, outperforming its Flat variant (Flat: hF=0.83, wF=0.77; Curriculum: hF=0.86, wF=0.80). Its hierarchical methods also maintained strong label coverage, with Hierarchical Instruct reaching nearly 100% label coherence at level 3. This suggests larger models respond well to more complex, instruction-based regimes, boosting coherence and hierarchical metrics. Moreover, although Mistral 7B's Flat variant excelled on Icecat Test data, all its Hierarchical Variants surpassed the Flat approach on WDC data, implying no overfitting and good generalization to noisier, more diverse datasets.

Mistral 7B perfrormed best in its Hierarchical Completion variant (with just title as input and hierarchical sequence of labels as output). Its wF and hF were 0.80 and 0.874. Though the model's wR (0.77) lags behind its precision (0.88) due to strict final-category matching Notably, these improvements did not undermine overall accuracy as its high hF score, indicates it

often places instances on the correct branch of the taxonomy, reflecting partial but meaningful correctness.

Experiment on Mistral 7B demonstrates that LLMs can be adapted to HPC task, leveraging hierarchical context of labels for improving accuracy and coherence of predictions. Meaning that such model benefits from and gains comprehension of the label hierarchy when predicting.

Against external benchmarks, Mistral 7B v0.3 set a higher standard for deep-learning models and weak LLMs. It outperformed the WDC-222 baseline model (group section 3.2), raising wPrecision by 0.040pp, wF1 by 0.024pp, and hF1 by 0.025pp, while maintaining similar wR. Although it did not surpass few-shot-based solutions of Gholamian et al. (2024), it is worth noticing that it still achieved relatively close results while haeving no additional context (few-shot), and being a significantly smaller model.

A - 4.2 Effect of Hierarchical Training Context

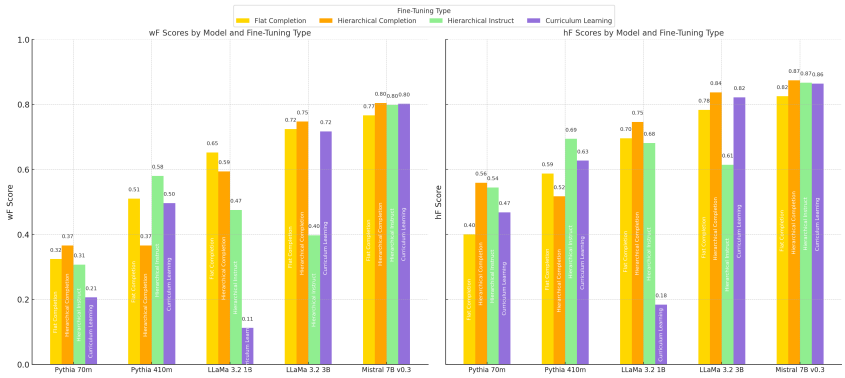


Figure 13: Bar charts of wF and hF scores per model & fine-tuning strategy

As shown on Figure 13, almost all hierarchical fine-tuning methods (Completion, Instruct, and Curriculum) improved metrics compared to the Flat Completion baseline, but gains varied. For Pythia 70m, hierarchical context improved wF1 and reduced severe errors, though only Hierarchical Completion (wF=0.37) beat Flat (wF=0.32) at level 3. For Pythia 410m, only Hierarchical Instruct boosted performance at level 3 (wF: from 0.51 to 0.58; hF: from 0.59 to 0.69), while other hierarchical methods were weaker than Flat. Notably, Pythia 410m was the only model that greatly benefited from the Hierarchical Instruct approach.

LLaMa-3.2-1B, only slightly larger than Pythia 410m, did not gain much from hierarchy. Its best strategy was Flat Completion (wF=0.65) over Hierarchical Completion (wF=0.59). Hier-

archical Instruct raised hF (0.75 vs. Flat's 0.70), implying better classification at higher levels, while Curriculum Fine-Tuning caused inconsistent and poor outputs (only 10% coverage at level 3).

LLaMa-3.2-3B did well with Flat (wF=0.72; hF=0.70) and performed even better with Hierarchical Completion (wF1=0.75; hF1=0.84). Curriculum Learning gave slightly lower wF1 (0.72) but higher hF1 (0.82), indicating fewer distant errors, though it introduced irregular formatting of generated separation tokens. Hierarchical Instruct remained ineffective for both LLaMa models, suggesting their sensitivity to training prompts length & complexity.

Hierarchical methods often showed a bigger gap between wF and hF, meaning that their errors were usually closer to the correct labels, indirectly indicating positive influence of hierarchical predictions. In contrast, Flat variants tended toward more distant misclassifications. Despite its complexity, Curriculum Learning rarely surpassed simpler hierarchical methods. Although it aided Mistral 7B, it did not outperform Hierarchical Completion or Instruct, and it showed no consistent advantages elsewhere, suggesting that multi-step fine-tuning should not be considered a default application for HPC with LLMs.

A - 4.3 Class Labels Coherence

Throughout fine-tuning, models had to recall and generate labels without constraints. For smaller models like Pythia 70m, adding hierarchical cues to the prompt greatly improved label coherence, boosting correctness by about 30 percentage points over the Flat variant. In contrast, for medium-sized models (Pythia 410m and LLaMa 3.2 1B), Flat classification unexpectedly preserved or even enhanced coherence. Although LLaMa 3.2 1B's Flat approach achieved the highest wF, Pythia 410m's Hierarchical Instruct achieved the best overall performance, showing that hierarchical prompts may have versatile implications on different LLMs.

For LLaMa 3.2 3B, hierarchical context improved coherence only in the Hierarchical Completion variant (87%); other hierarchical methods lagged behind the Flat baseline (85%), hinting that increased prompt complexity can sometimes reduce label coherence. Mistral 7B showed the strongest gains in coherence from hierarchical training. All hierarchical methods, especially Hierarchical Instruct, consistently improved label coherence, reaching up to 100% at

the deepest level, suggesting that fine-tuning on hierarchical prompts increases both accuracy and label consistency of larger models, even without strict vocabulary constraining rules.

A - 4.4 Computation time

For smaller models, Full Fine-Tuning was efficient, with training times of 35 minutes for Pythia 70m and 2 hours 22 minutes to 3 hours 33 minutes for Pythia 410m. Inference on a T4 GPU took approximately 1.5 minutes and 7 minutes, respectively. Larger models used Parameter-Efficient Fine-Tuning (PEFT) on A100 GPUs. LLaMa 3.2 1B trained in under 1 hour, while LLaMa 3.2 3B required 1.5 to 2 hours. Inference times varied: on L4, LLaMa 3.2 1B took 3 minutes, and LLaMa 3.2 3B ranged between 5 minutes 24 seconds and 9 minutes 5 seconds. Mistral 7B trained in 2 hours 40 minutes to 2 hours 55 minutes, with Curriculum Learning extending to 4 hours 9 minutes. For inference, the Hierarchical Completion on L4 took 18 minutes, while Hierarchical Instruct and Curriculum Learning on A100 GPUs took 4 minutes.

A - 5 Conclusions, Limitations & Further Research

The results of the study confirm that incorporating hierarchical context during fine-tuning can significantly improve models' ability to predict correct hierarchical paths. Majority of tested models benefited from hierarchical fine-tuning, especially from hierarchical completion method which usually produced improved accuracy while keeping the label coherence on high level. Mistral 7B model, fine-tuned with hierarchical methods consistently outperformed those with flat classification proving that tuning LLMs to make prediction step-by-step may yield better performance. Hierarchical fine-tuning facilitated a deeper understanding of hierarchical structures and enhanced the memorization of label names across different levels, decreasing severity of mispredictions, when compared to flat methods. Larger models, such as Mistral 7B, achieved near-perfect label coherence at the deepest hierarchy levels, demonstrating their capability to grasp and retain complex hierarchical relationships. Conversely, medium-sized models like LLaMa 3.2-1B occasionally performed better with flat fine-tuning, suggesting that excessive prompt complexity may hinder label recall of medium-sized models. Interestingly,

most complex method (Curriculum Learning) did not prove to surpass performance of simpler hierarchical implementations and often distorted the results. The study also validates that causal modeling-based fine-tuning is an effective approach for rapid adaptation of decoder-only LLMs for HTC. Most models adapted through causal language modeling showed strong performance and resilience on messy test data, indicating that hierarchical fine-tuning LLMs can be successful for HPC, implemented without architectural changes or complex training strategies.

Based on the results of the study, in the business context, suggested method of fine-tuning LLMs for HPC is applicable rapid development of product categorization system with relatively high accuracy without any architectural changes for the pre-trained model. However, it can be problematic with growing scale of categories and products to categorize due to computational demands of inference and re-training with Large Language Models. When dealing with large-scale HPC, the proposed methodology should be implemented with more curated training data, more efficient and batched inference methods, and smaller LLMs when computational resources are limited.

Limitations of this study are related to restricted training data to the Computers & Electronic category, limiting the ability to generalize findings to other product hierarchies. Additionally models were evaluated only on WDC-222 dataset of the same origin as Icecat data used for training, which may raise concerns about generalisation ability. Lastly, Inference with complex models was time-consuming, taking approximately three minutes for a 2894 samples. This highlights a need for optimization to make it more scalable for business application.

Recommendations for future research include wider range of product categories and increased diversity of evaluation datasets. Potential improvement also lies in combining fine-tuned models with different LLM-based method like prompting or RAG. For the models proposed, different generation strategies like beam-search remain unexplored. Moreover, investigating more sophisticated fine-tuning strategies, inference mechanisms, and training optimizations may lead to more business-ready outcomes. Finally, similar procedures can be tested on state-of-the-art LLMs like GPT-4o, Google Gemini etc.

References

J. Achiam, S. Adler, S. Agarwal, et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

C. Alagoz. Performance improvement in multi-class classification via automated hierarchy generation and exploitation through extended lcpn schemes, 2023. URL https://arxiv.org/abs/2310.20641.

R. Anil, S. Borgeaud, J.-B. Alayrac, et al. Gemini: A family of highly capable multimodal models, 2024. URL https://arxiv.org/abs/2312.11805.

AnthropicAI. The claude 3 model family: Opus, sonnet, haiku. URL https://api.semanticscholar.org/CorpusID:268232499.

Arcus, July 2024. URL https://www.arcus.co/blog/rag-at-planet-scale.

J. Baek, A. F. Aji, and A. Saffari. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering, 2023. URL https://arxiv.org/abs/2306.04136.

N. Barlaug and J. A. Gulla. Neural networks for entity matching: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15:1 – 37, 2020. URL https://api.semanticscholar.org/CorpusID:233486427.

E. Bayram, A. Garcia-Duran, and R. West. Node attribute completion in knowledge graphs with multi-relational propagation, 2020. URL https://arxiv.org/abs/2011.05301.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

P. Bhavsar. Mastering rag: How to select an embedding model, 2024. URL https://www.galileo.ai/blog/mastering-rag-how-to-select-an-embedding-modelimpact-of-embedd

S. Biderman, H. Schoelkopf, Q. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, A. Skowron, L. Sutawika, and O. van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023.

S. S. Birunda and R. K. Devi. A review on word embedding techniques for text classification. In *Innovative Data Communication Technologies and Application*, 2021. URL https://api.semanticscholar.org/CorpusID:234104086.

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, 2020. URL https://arxiv.org/abs/2005.14165.

A. Cattaneo, D. Justus, H. Mellor, D. Orr, J. Maloberti, Z. Liu, T. Farnsworth, A. Fitzgibbon, B. Banaszewski, and C. Luschi. Bess: Balanced entity sampling and sharing for large-scale knowledge graph completion, 2022. URL https://arxiv.org/abs/2211.12281.

A. Cevahir and K. Murakami. Large-scale multi-class and hierarchical product categorization for an e-commerce giant. In *International Conference on Computational Linguistics*, 2016. URL https://api.semanticscholar.org/CorpusID:18840587.

Bibliography

H. Chen, Q. Ma, Z. Lin, and J. Yan. Hierarchy-aware label semantics matching network for hierarchical text classification. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4370–4379, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.337. URL https://aclanthology.org/2021.acl-long.337.

H. Chen, X. Shen, Q. Lv, J. Wang, X. Ni, and J. Ye. Sac-kg: Exploiting large language models as skilled automatic constructors for domain knowledge graphs, 2024a. URL https://arxiv.org/abs/2410.02811.

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, 2024b. URL https://arxiv.org/abs/2406.17534.

S. Chen, Y. Li, S. Lu, H. Van, H. J. W. L. Aerts, G. K. Savova, and D. S. Bitterman. Evaluating the chatgpt family of models for biomedical reasoning and classification. *Journal of the American Medical Informatics Association*, 31(4):940–948, Jan. 2024c. ISSN 1527-974X. doi: 10.1093/jamia/ocad256. URL http://dx.doi.org/10.1093/jamia/ocad256.

Z. Cheng, W. Zhang, C.-C. Chou, Y.-Y. Jau, A. Pathak, P. Gao, and U. Batur. E-commerce product categorization with LLM-based dual-expert classification paradigm. In S. Kumar, V. Balachandran, C. Y. Park, W. Shi, S. A. Hayati, Y. Tsvetkov, N. Smith, H. Hajishirzi, D. Kang, and D. Jurgens, editors, *Proceedings of the 1st Workshop on Customizable NLP: Progress and Challenges in Customizing NLP for a Domain, Application, Group, or Individual (CustomNLP4U)*, pages 294–304, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.customnlp4u-1.22. URL https://aclanthology.org/2024.customnlp4u-1.22.

O. Cherednichenko, O. Ivashchenko, and M. Vovk. Towards pipeline construction for product matching task. *41 st International Conference on Organizational Science Development*, 2022. URL https://api.semanticscholar.org/CorpusID:249940666.

N. Choudhary and C. K. Reddy. Complex logical reasoning over knowledge graphs using large language models, 2024. URL https://arxiv.org/abs/2305.01157.

J. Dai, T. Wang, and S. Wang. A deep forest method for classifying e-commerce products by using title information. In *2020 International Conference on Computing, Networking and Communications (ICNC)*, page 1–5. IEEE, Feb. 2020. doi: 10.1109/icnc47757.2020.9049751. URL http://dx.doi.org/10.1109/ICNC47757.2020.9049751.

M. H. Daniel Han and U. team. Unsloth, 2023. URL http://github.com/unslothai/unsloth.

M. A. de Luis Balaguer, V. Benara, R. L. de Freitas Cunha, R. de M. Estevao Filho, T. Hendry, D. Holstein, J. Marsman, N. Mecklenburg, S. Malvar, L. Nunes, R. Padilha, M. Sharp, B. L. B. Silva, S. Sharma, V. Aski, and R. Chandra. Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *ArXiv*, abs/2401.08406, 2024. URL https://api.semanticscholar.org/CorpusID:267027552.

T. Deshpande, N. Kowtal, and R. Joshi. Chain-of-translation prompting (cotr): A novel prompting technique for low resource languages, 2024. URL https://arxiv.org/abs/2409.04512.

T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. URL https://arxiv.org/abs/2305.14314.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Bibliography

Y. Dong, H. Zhang, C. Li, S. Guo, V. C. M. Leung, and X. Hu. Fine-tuning and deploying large language models over edges: Issues and approaches, 2024. URL https://arxiv.org/abs/2408.10691.

M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou. The faiss library, 2024. URL https://arxiv.org/abs/2401.08281.

S. T. Dumais and H. Chen. Hierarchical classification of web content, 2000. URL https://api.semanticscholar.org/CorpusID:1194024.

C. d'Amato, P. Ristoski, P. Petrovski, P. Mika, and H. Paulheim. A machine learning approach for product matching and categorization. *Semant. Web*, 9(5):707–728, Jan. 2018. ISSN 1570-0844. doi: 10.3233/SW-180300. URL https://doi.org/10.3233/SW-180300.

O. Fagbohun, R. M. Harrison, and A. Dereventsov. An empirical categorization of prompting techniques for large language models: A practitioner's guide, 2024. URL https://arxiv.org/abs/2402.14837.

L. Fan, L. Li, Z. Ma, S. Lee, H. Yu, and L. Hemphill. A bibliometric review of large language models research from 2017 to 2023, 2023.

I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969. URL https://api.semanticscholar.org/CorpusID:17349112.

J. Foxcroft, T. Chen, K. Padmanabhan, B. Keng, and L. Antonie. Product matching lessons and recommendations from a real world application. *Proceedings of the Canadian Conference on Artificial Intelligence*, 2021. URL https://api.semanticscholar.org/CorpusID:237927201.

Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang. Retrieval-augmented generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

S. Gaur, A. Dagar, A. Punia, and P. Kumar. A brief study of prompting techniques for reasoning tasks. In I. Woungang, S. K. Dhurandher, and Y. J. Singh, editors, *Proceedings of the NIELIT's International Conference on Communication, Electronics and Digital Technology*, pages 147–159, Singapore, 2024. Springer Nature Singapore. ISBN 978-981-97-3601-0.

S. Gholamian, G. Romani, B. Rudnikowicz, and S. Skylaki. Llm-based robust product classification in commerce and compliance, 2024. URL https://arxiv.org/abs/2408.05874.

A. Grattafiori, A. Dubey, A. Jauhri, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

L. Groß, R. Walter, N. Zoppi, and A. R. Justus. Rebuild product classification. Masters thesis, Nova School of Business and Economics, Lisbon, Portugal, 2024.

S. Guo and H. Zhao. Hierarchical classification with multi-path selection based on granular computing. *Artif. Intell. Rev.*, 54(3):2067–2089, Mar. 2021. ISSN 0269-2821. doi: 10.1007/s10462-020-09899-2. URL https://doi.org/10.1007/s10462-020-09899-2.

V. Gupta, H. Karnick, A. Bansal, and P. Jhala. Product classification in e-commerce using distributional semantics, 2016. URL https://arxiv.org/abs/1606.06083.

S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL https://aclanthology.org/2020.acl-main.740.

Bibliography

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, Nov. 2024. ISSN 1558-2868. doi: 10.1145/3703155. URL http://dx.doi.org/10.1145/3703155.

R. Huang, C. Wei, B. Wang, J. Yang, X. Xu, S. Wu, and S. Huang. Well performance prediction based on long short-term memory (lstm) neural network. *Journal of Petroleum Science and Engineering*, 208:109686, 10 2021. doi: 10.1016/j.petrol.2021.109686.

W. Huang, E. Chen, Q. Liu, Y. Chen, Z. Huang, Y. Liu, Z. Zhao, D. Zhang, and S. Wang. Hierarchical multi-label text classification: An attention-based recurrent network approach, 2019. URL https://api.semanticscholar.org/CorpusID:207758666.

V. Jain, M. Rungta, Y. Zhuang, Y. Yu, Z. Wang, M. Gao, J. Skolnick, and C. Zhang. Higen: Hierarchy-aware sequence generation for hierarchical text classification, 2024. URL https://arxiv.org/abs/2402.01696.

Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, Mar. 2023. ISSN 1557-7341. doi: 10.1145/3571730. URL http://dx.doi.org/10.1145/3571730.

A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023a.

T. Jiang, D. Wang, L. Sun, Z. Chen, F. Zhuang, and Q. Yang. Exploiting global and local hierarchies for hierarchical text classification, 2022. URL https://arxiv.org/abs/2205.02613.

X. Jiang, C. Xu, Y. Shen, X. Sun, L. Tang, S. Wang, Z. Chen, Y. Wang, and J. Guo. On the evolution of knowledge graphs: A survey and perspective, 2023b. URL https://arxiv.org/abs/2310.04835.

M. Khatri. Cosine similarity function for the temporal dynamic web data. 2012. URL https://api.semanticscholar.org/CorpusID:14356140.

S. Kiritchenko, S. Matwin, R. Nock, and A. F. Famili. *Learning and Evaluation in the Presence of Class Hierarchies: Application to Text Categorization*, page 395–406. Springer Berlin Heidelberg, 2006. ISBN 9783540248408. doi: 10.1007/11766247_34. URL http://dx.doi.org/10.1007/11766247_34.

T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners, 2023. URL https://arxiv.org/abs/2205.11916.

H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data Knowl. Eng.*, 69: 197–210, 2010. URL https://api.semanticscholar.org/CorpusID:16535573.

Z. Kozareva, Q. Li, K. Zhai, and W. Guo. Recognizing salient entities in shopping queries, 01 2016.

A. Krishnan and A. Amarthaluri. Large scale product categorization using structured and unstructured attributes, 2019. URL https://arxiv.org/abs/1903.04254.

Bibliography

A. Kumar, A. Pandey, R. Gadia, and M. Mishra. Building knowledge graph using pre-trained language model for learning entity-aware relationships. *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 310–315, 2020. URL https://api.semanticscholar.org/CorpusID:226266162.

S. Kumar. Mastering rag: A deep dive into embeddings, Aug 2024. URL https://medium.com/@shravankoninti/mastering-rag-a-deep-dive-into-embeddings-b78782aa1259.

A. Köpf, Y. Kilcher, D. von Rütte, S. Anagnostidis, Z.-R. Tam, K. Stevens, A. Barhoum, N. M. Duc, O. Stanley, R. Nagyfi, S. ES, S. Suri, D. Glushkov, A. Dantuluri, A. Maguire, C. Schuhmann, H. Nguyen, and A. Mattick. Openassistant conversations – democratizing large language model alignment, 2023. URL https://arxiv.org/abs/2304.07327.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL https://arxiv.org/abs/2005.11401.

J. Li, Y. Zhao, Y. Li, G. Li, and Z. Jin. Acecoder: An effective prompting technique specialized in code generation. *ACM Trans. Softw. Eng. Methodol.*, 33(8), Nov. 2024a. ISSN 1049-331X. doi: 10.1145/3675395. URL https://doi.org/10.1145/3675395.

M. Y. Li, S. Kok, and L. Tan. Don't classify, translate: Multi-level e-commerce product categorization via machine translation. *ArXiv*, abs/1812.05774, 2018. URL https://api.semanticscholar.org/CorpusID:55702283.

Y. Li, J. Li, Y. Suhara, J. Wang, W. Hirota, and W. C. Tan. Deep entity matching. *Journal of Data and Information Quality (JDIQ)*, 13:1 – 17, 2021. URL https://api.semanticscholar.org/CorpusID:231731102.

Y. Li, S. Ma, X. Wang, S. Huang, C. Jiang, H.-T. Zheng, P. Xie, F. Huang, and Y. Jiang. Ecomgpt: Instruction-tuning large language models with chain-of-task tasks for e-commerce. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18582–18590, Mar. 2024b. doi: 10.1609/aaai.v38i17.29820. URL https://ojs.aaai.org/index.php/AAAI/article/view/29820.

P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning, 2016. URL https://arxiv.org/abs/1605.05101.

W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang. K-bert: Enabling language representation with knowledge graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34: 2901–2908, 04 2020. doi: 10.1609/aaai.v34i03.5681.

S. Longpre, L. Hou, T. Vu, A. Webson, H. W. Chung, Y. Tay, D. Zhou, Q. V. Le, B. Zoph, J. Wei, and A. Roberts. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:256415991.

S. Ma, S. Huang, S. Huang, X. Wang, Y. Li, H.-T. Zheng, P. Xie, F. Huang, and Y. Jiang. Ecomgpt-ct: Continual pre-training of e-commerce large language models with semi-structured data, 2023. URL https://arxiv.org/abs/2312.15696.

A. Martinek, S. Łukasik, and A. H. Gandomi. Text-based product matching – semi-supervised clustering approach, 2024. URL https://arxiv.org/abs/2402.10091.

Bibliography

J. MathavRaj, V. Kushala, H. Warrier, and Y. Gupta. Fine tuning llm for enterprise: Practical guidelines and recommendations. *ArXiv*, abs/2404.10779, 2024. URL https://api.semanticscholar.org/CorpusID:269188062.

N. Mathivanan, N. Ghani, and R. Mohd Janor. Analysis of k-means clustering algorithm: A case study using large scale e-commerce products, 11 2019.

G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models, 2017. URL https://arxiv.org/abs/1707.05589.

B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heinz, and D. Roth. Recent advances in natural language processing via large pre-trained language models: A survey, 2021. URL https://arxiv.org/abs/2111.01243.

S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao. Large language models: A survey, 2024. URL https://arxiv.org/abs/2402.06196.

F. M. Miranda, N. Köhnecke, and B. Y. Renard. Hiclass: a python library for local hierarchical classification compatible with scikit-learn. *Journal of Machine Learning Research*, 24(29):1–17, 2023. URL http://jmlr.org/papers/v24/21-1518.html.

M. Mosbach, T. Pimentel, S. Ravfogel, D. Klakow, and Y. Elazar. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12284–12314, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.779. URL https://aclanthology.org/2023.findings-acl.779.

S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 19–34, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450347037. doi: 10.1145/3183713.3196926. URL https://doi.org/10.1145/3183713.3196926.

N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022. doi: 10.48550/ARXIV.2210.07316. URL https://arxiv.org/abs/2210.07316.

Nils Richter and Christian Bizer. Wdc-222 gold standard for hierarchical product categorization, n.d. URL https://data.dws.informatik.uni-mannheim.de/largescaleproductcorpus/categorization/. Accessed: 2024-11-11.

Open Icecat. Open icecat catalog. URL https://icecat.de/. Accessed: 2024-11-11.

S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599, July 2024. ISSN 2326-3865. doi: 10.1109/tkde.2024.3352100. URL http://dx.doi.org/10.1109/TKDE.2024.3352100.

V. B. Parthasarathy, A. Zafar, A. Khan, and A. Shahid. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities, 2024.

A. Patra, V. Vivek, B. R. Shambhavi, K. S. Sindhu, and S. A. Balaji. Product classification in e-commerce sites, 2021. URL https://api.semanticscholar.org/CorpusID:235082768.

Bibliography

D. Paulsen, Y. Govind, and A. Doan. Sparkly: A simple yet surprisingly strong tf/idf blocker for entity matching. *Proc. VLDB Endow.*, 16:1507–1519, 2023. URL https://api.semanticscholar.org/CorpusID:258190793.

R. Peeters and C. Bizer. Using chatgpt for entity matching, 2023. URL https://arxiv.org/abs/2305.03423.

R. Peeters, A. Steiner, and C. Bizer. Entity matching using large language models, 2024. URL https://arxiv.org/abs/2310.11244.

B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, and S. Tang. Graph retrieval-augmented generation: A survey, 2024. URL https://arxiv.org/abs/2408.08921.

F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. Language models as knowledge bases?, 2019. URL https://arxiv.org/abs/1909.01066.

P. K. Pushp and M. M. Srivastava. Train once, test anywhere: Zero-shot learning for text classification. *ArXiv*, abs/1712.05972, 2017. URL https://api.semanticscholar.org/CorpusID:7244042.

A. Radford and K. Narasimhan. Improving language understanding by generative pre-training, 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019. URL https://openai.com/research/language-models-are-unsupervised-multitask-learners.

K. V. Rajan and E. Lambert. Entity matching for digital world: A modern approach using artificial intelligence and machine learning. *Global Journal of Computer Science and Technology*, 23(D1):35–44, Apr. 2023. doi: 10.34257/GJCSTDVOL23IS1PG35. URL https://computerresearch.org/index.php/computer/article/view/102285.

K. K. Rangan and Y. Yin. A fine-tuning enhanced rag system with quantized influence measure as ai judge. *Scientific Reports*, 14, 2024. URL https://api.semanticscholar.org/CorpusID:268033066.

J. Risch, S. Garda, and R. Krestel. Hierarchical document classification as a sequence generation task. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, JCDL '20, page 147–155, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375856. doi: 10.1145/3383583.3398538. URL https://doi.org/10.1145/3383583.3398538.

P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. S. Mondal, and A. Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *ArXiv*, abs/2402.07927, 2024. URL https://api.semanticscholar.org/CorpusID:267636769.

B. Sarmah, B. Hall, R. Rao, S. Patel, S. Pasquali, and D. Mehta. Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction, 2024. URL https://arxiv.org/abs/2408.04948.

S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, P. S. Dulepet, S. Vidyadhara, D. Ki, S. Agrawal, C. Pham, G. Kroiz, F. Li, H. Tao, A. Srivastava, H. D. Costa, S. Gupta, M. L. Rogers, I. Goncearenco, G. Sarli, I. Galynker, D. Peskoff, M. Carpuat, J. White, S. Anadkat, A. Hoyle, and P. Resnik. The prompt report: A systematic survey of prompting techniques, 2024. URL https://arxiv.org/abs/2406.06608.

Bibliography

B. Schwartz. Ceos want trump to change course on tariffs. he isn't budging. *The Wall Street Journal*, 2024. URL https://www.wsj.com/economy/trade/trump-tariff-plan-business-lobbying-8f02ccea. Accessed: 2024-12-15.

D. Shen, J.-D. Ruvini, M. Somaiya, and N. Sundaresan. Item categorization in the e-commerce domain, 10 2011.

J. Shin, C. Tang, T. Mohati, M. Nayebi, S. Wang, and H. Hemmati. Prompt engineering or fine tuning: An empirical assessment of large language models in automated software engineering tasks, 2023. URL https://arxiv.org/abs/2310.10508.

Statista. Gross merchandise value (gmv) of tiktok shops worldwide as of 2024, by product category, 2024. URL https://www.statista.com/statistics/1461508/gross-merchandise-value-tiktok-shops-worldwide Accessed: 2024-12-15.

A. Steiner, R. Peeters, and C. Bizer. Fine-tuning large language models for entity matching, 2024. URL https://arxiv.org/abs/2409.08185.

M. Sun, J. Niu, X. Yang, Y. Gu, and W. Zhang. Cehmr: Curriculum learning enhanced hierarchical multi-label classification for medication recommendation. *Artificial Intelligence in Medicine*, 143:102613, Sept. 2023. ISSN 0933-3657. doi: 10.1016/j.artmed.2023.102613. URL http://dx.doi.org/10.1016/j.artmed.2023.102613.

Y. SUN, A. K. C. WONG, and M. S. KAMEL. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04): 687–719, June 2009. ISSN 1793-6381. doi: 10.1142/s0218001409007326. URL http://dx.doi.org/10.1142/S0218001409007326.

Tamanna. Langchain vs. llamaindex: A comprehensive comparison for retrieval-augmented generation (rag), Oct 2024. URL https://medium.com/@tam.tamanna18/langchain-vs-llamaindex-a-comprehensive-comparison-for-r

L. Tan, M. Y. Li, and S. Kok. E-commerce product categorization via machine translation. *ACM Trans. Manage. Inf. Syst.*, 11(3), July 2020. ISSN 2158-656X. doi: 10.1145/3382189. URL https://doi.org/10.1145/3382189.

F. Torba, C. Gravier, C. Laclau, A. Kammoun, and J. Subercaze. A Study on Hierarchical Text Classification as a Seq2seq Task. In *46th European Conference on Information Retrieval (ECIR 2024)*, GLASGOW, United Kingdom, Mar. 2024. URL https://hal.science/hal-04423348.

H. Touvron, L. Martin, K. Stone, et al. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

J. Tracz, P. Wójcik, K. Jasinska-Kobus, R. Belluzzo, R. Mroczkowski, and I. Gawlik. Bert-based similarity learning for product matching. In *ECOMNLP*, 2020. URL https://api.semanticscholar.org/CorpusID:227231218.

F. Trad and A. Chehab. Prompt engineering or fine-tuning? a case study on phishing detection with large language models. *Mach. Learn. Knowl. Extr.*, 6:367–384, 2024. URL https://api.semanticscholar.org/CorpusID:267548910.

D. Trautmann. Large language model prompt chaining for long legal document classification. *ArXiv*, abs/2308.04138, 2023. URL https://api.semanticscholar.org/CorpusID:260704370.

Bibliography

A. Valenzuela. Chain-of-thought prompting: Step-by-step reasoning with llms, Jul 2024. URL https://www.datacamp.com/tutorial/chain-of-thought-prompting.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

A. Vilcek, S. Mottaghinejad, M. U. S. SHI, K. Gupte, W. Labs, Steven.Yukai, and Shi. Transformer-based deep siamese network for at-scale product matching and one-shot hierarchy classification. 2021. URL https://api.semanticscholar.org/CorpusID:237139778.

T. Wang, X. Chen, H. Lin, X. Chen, X. Han, H. Wang, Z. Zeng, and L. Sun. Match, compare, or select? an investigation of large language models for entity matching, 2024a. URL https://arxiv.org/abs/2405.16884.

X. Wang, P. Kapanipathi, R. Musa, M. Yu, K. Talamadupula, I. Abdelaziz, M. Chang, A. Fokoue, B. Makni, N. Mattei, and M. Witbrock. Improving natural language inference using external knowledge in the science questions domain, 09 2018.

Y. Wang, S. Si, D. Li, M. Lukasik, F. Yu, C.-J. Hsieh, I. S. Dhillon, and S. Kumar. Two-stage llm fine-tuning with less specialization and more generalization, 2024b. URL https://arxiv.org/abs/2211.00635.

Z. Wang, P. Wang, L. Huang, X. Sun, and H. Wang. Incorporating hierarchy into text encoder: a contrastive learning approach for hierarchical text classification. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7109–7119, Dublin, Ireland, May 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.491. URL https://aclanthology.org/2022.acl-long.491.

Z. Wang, P. Wang, T. Liu, B. Lin, Y. Cao, Z. Sui, and H. Wang. Hpt: Hierarchy-aware prompt tuning for hierarchical text classification, 2022b. URL https://arxiv.org/abs/2204.13413.

Z. Wang, Y. Pang, and Y. Lin. Large language models are zero-shot text classifiers. *ArXiv*, abs/2312.01044, 2023. URL https://api.semanticscholar.org/CorpusID:265609304.

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, 2023. URL https://arxiv.org/abs/2201.11903.

G. Weiss and F. Provost. The effect of class distribution on classifier learning: An empirical study. *Tech Rep*, 09 2001.

Y. Wen, N. Jain, J. Kirchenbauer, M. Goldblum, J. Geiping, and T. Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *ArXiv*, abs/2302.03668, 2023. URL https://api.semanticscholar.org/CorpusID:256627601.

Y. Wen, Z. Wang, and J. Sun. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models, 2024. URL https://arxiv.org/abs/2308.09729.

World Customs Organization. Annual report 2022-2023, 2023. Accessed: 2024-11-13.

Z. Xu, M. J. Cruz, M. Guevara, T. Wang, M. Deshpande, X. Wang, and Z. Li. Retrieval-augmented generation with knowledge graphs for customer service question answering. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2024, page 2905–2909. ACM, July 2024. doi: 10.1145/3626772.3661370. URL http://dx.doi.org/10.1145/3626772.3661370.

Bibliography

J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, B. Yin, and X. Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond, 2023. URL https://arxiv.org/abs/2304.13712.

S.-J. Yen and Y.-S. Lee. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, 36(3):5718–5727, Apr. 2009. ISSN 0957-4174. doi: 10.1016/j.eswa.2008.06.108. URL http://dx.doi.org/10.1016/j.eswa.2008.06.108.

C. Yu, Y. Shen, Y. Mao, and L. Cai. Constrained sequence-to-tree generation for hierarchical text classification, 2022. URL https://arxiv.org/abs/2204.00811.

D. Zhang, Z. Yuan, Y. Liu, F. Zhuang, and H. Xiong. E-bert: A phrase and product knowledge enhanced language model for e-commerce. 2020. URL https://api.semanticscholar.org/CorpusID:221516807.

S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu, and G. Wang. Instruction tuning for large language models: A survey, 2024a. URL https://arxiv.org/abs/2308.10792.

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, 2024b. URL https://arxiv.org/abs/2403.00165.

H. Zhao, F. Yang, B. Shen, H. Lakkaraju, and M. Du. Towards uncovering how large language model works: An explainability perspective, 2024. URL https://arxiv.org/abs/2402.10688.

Z. Zhou, X. Ning, K. Hong, T. Fu, J. Xu, S. Li, Y. Lou, L. Wang, Z. Yuan, X. Li, S. Yan, G. Dai, X.-P. Zhang, Y. Dong, and Y. Wang. A survey on efficient inference for large language models, 2024. URL https://arxiv.org/abs/2404.14294.

Y. Zhu, J.-C. Gu, C. Sikora, H. Ko, Y. Liu, C.-C. Lin, L. Shu, L. Luo, L. Meng, B. Liu, and J. Chen. Accelerating inference of retrieval-augmented generation via sparse context selection, 2024. URL https://arxiv.org/abs/2405.16178.

Appendix - 1 Group Work

Appendix - 1.1 Exploratory Data Analysis

Appendix - 1.1.1 Hierarchical Analysis

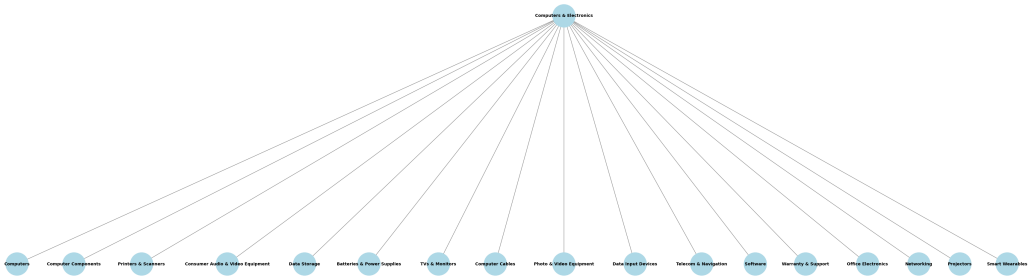


Figure 25: First and Second Levels of Product Categories in WDC and Icecat

Appendix - 1.1.2 Title Analysis

Table 29: Descriptive Statistics Summary for Icecat and WDC

Statistic	Icecat Value	WDC Value
Count	765,473	2,984
Mean	70.42	54.32
Standard Deviation	36.26	19.66
Minimum	2.00	6.00
25th Percentile	45.00	40.00
Median (50th Percentile)	57.00	54.00
75th Percentile	90.00	66.00
Maximum	537.00	175.00

Appendix - 1.1.3 Description Analysis

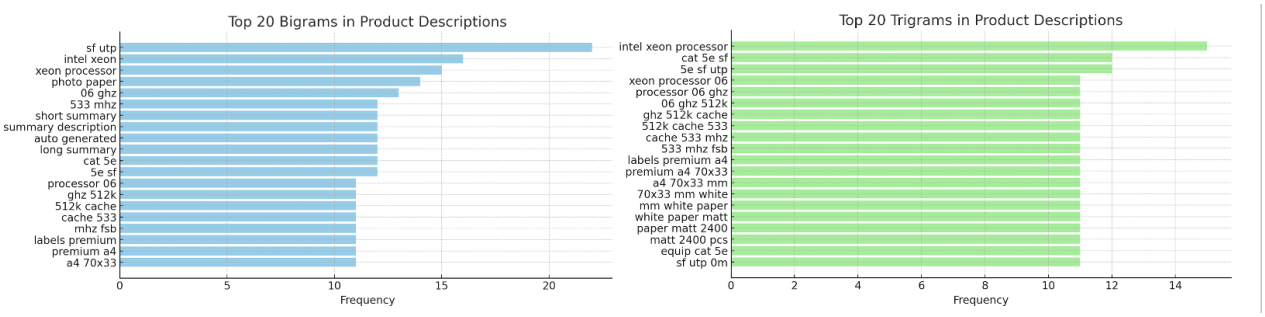


Figure 26: Top 20 Biagrams and Triagrams in Product Descriptions - WDC

Appendix - 1.1.4 Brand Analysis

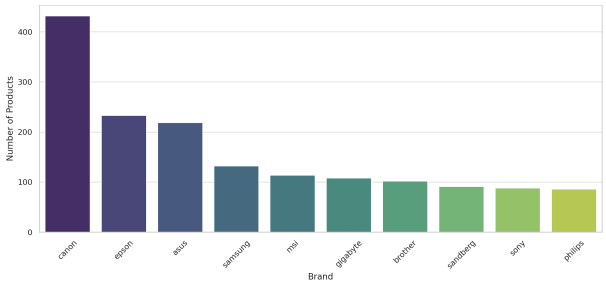


Figure 27: Top 10 Most Frequent Brands - WDC

Appendix - 1.1.5 Mislassifications Analysis

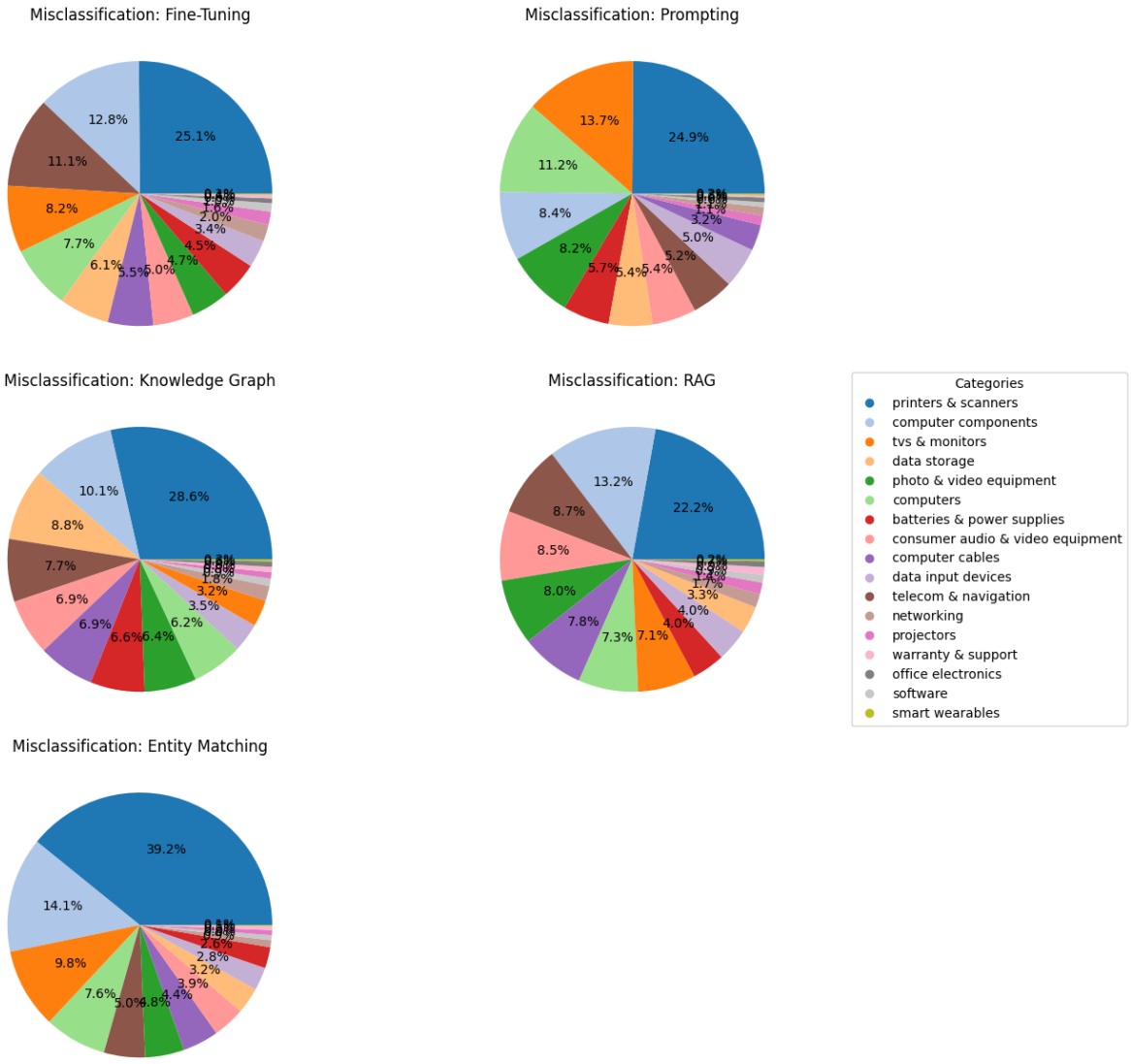


Figure 28: Normalized Misclassification

Appendix - 2 Fine-Tuning Appendix

Appendix - 2.1 Fine-Tuning Models

Table 30: Overview of Language Models

Model Name	Training Data	Number of Parameters	Company/Organization
Pythia-70m	The Pile (diverse English text corpus)	∼70 Million	EleutherAI
Pythia-410m	The Pile (diverse English text corpus)	∼410 Million	EleutherAI
LLaMa 3.2 1B (base & instruct)	Curated diverse multilingual web data	∼1 Billion	Meta (LLaMa)
LLaMa 3.2 3B (base & instruct)	Curated diverse multilingual web data	∼3 Billion	Meta (LLaMa)
Mistral-7B-v0.3 (base & instruct)	Curated multi-domain open web text	∼7 Billion	Mistral AI

Appendix - 2.2 Fine-tuning Prompts

Flat Fine-Tuning	
Models	`Pythia70m, Pythia410m, LLaMa3.2-1B, LLaMa3.2-3b, Mistral-7b-v0.3`
Prompt Template	`{bos_token}{input_title}{sep_token}{Level_3_class}{eos_token}`
Example (LLaMa)	`<|begin_of_text|> Fujitsu ETERNUS LT40 2S SAS tape auto loader/library 144000 GB 2U Black [SEP] Tape Auto Loaders & Libraries <|endoftext|>`
Hierarchical Fine-Tuning	
Models	`Pythia70m, Pythia410m, LLaMa3.2-1B, LLaMa3.2-3b, Mistral-7b-v0.3`
Prompt Template	`{bos_token}{input_title}{sep_token} [L1] {Level_1_class}{class_sep} [L2] {Level_2_class}{class_sep} [L3] {Level_3_class}{eos_token}`
Example (LLaMa)	`<|begin_of_text|> Fujitsu ETERNUS LT40 2S SAS tape auto loader/library 144000 GB 2U Black [SEP] [L1] Data Storage [SEP_LVL] [L2] Data Storage Devices [SEP_LVL] [L3] Tape Auto Loaders & Libraries <|endoftext|>`
Hierarchical Instruct Fine-Tuning	
Models	`Pythia70m, Pythia410m, LLaMa3.2-1B-Instruct, LLaMa3.2-3b, Mistral-7b-v0.3-Instruct`
Prompt Template	`{bos_token} As a classification system for e-commerce products from the computers & electronics product group, your task is to classify a given product title into a hierarchical category. Each category has three levels: Level 1: The broadest category. Level 2: A subcategory of Level 1. Level 3: A subcategory of Level 2. Use the following format for your classifications: [SEP] [L1] class name level 1 [SEP_LVL] [L2] class name level 2 [SEP_LVL] [L3] class name level 3. Please classify the following product title: {input_title} Classification: {sep_token}`

Example (LLaMa)	<|begin_of_text|> As a classification system for e-commerce products from the computers & electronics product group, your task is to classify a given product title into a hierarchical category. Each category has three levels: Level 1: The broadest category. Level 2: A subcategory of Level 1. Level 3: A subcategory of Level 2. Use the following format for your classifications: [SEP] [L1] class name level 1 [SEP_LVL] [L2] class name level 2 [SEP_LVL] [L3] class name level 3. Please classify the following product title: Fujitsu ETERNUS LT40 2S SAS tape auto loader/library 144000 GB 2U Black Classification: [SEP]
Curriculum Learning	
Models	Pythia70m, Pythia410m, LLaMa3.2-1B, LLaMa3.2-3b, Mistral-7b-v0.3
Prompt Template (Level 1)	{BOS_TOKEN}{input_title}{SEP_TOKEN}{LEVEL_TOKENS[1]}{LEVEL_1}{EOS_TOKEN}
Example (LLaMa)	<|begin_of_text|> Fujitsu ETERNUS LT40 2S SAS tape auto loader/library 144000 GB Black [SEP] [L1] Data Storage <|endoftext|>
Prompt Template (Level 2)	{BOS_TOKEN}{input_title}{SEP_TOKEN}{LEVEL_TOKENS[1]}{LEVEL_1}{CLASS_SEP}{LEVEL_TOKENS[2]}{LEVEL_2}{EOS_TOKEN}
Example (LLaMa)	<|begin_of_text|> Fujitsu ETERNUS LT40 2S SAS tape auto loader/library 144000 GB Black [SEP] [L1] Data Storage [SEP_LVL] [L2] Data Storage Devices <|endoftext|>
Prompt Template (Level 3)	{BOS_TOKEN}{input_title}{SEP_TOKEN}{LEVEL_TOKENS[1]}{LEVEL_1}{CLASS_SEP}{LEVEL_TOKENS[2]}{LEVEL_2}{CLASS_SEP}{LEVEL_TOKENS[3]}{LEVEL_3}{EOS_TOKEN}
Example (LLaMa)	<|begin_of_text|> Fujitsu ETERNUS LT40 2S SAS tape auto loader/library 144000 GB Black [SEP] [L1] Data Storage [SEP_LVL] [L2] Data Storage Devices [SEP_LVL] [L3] Tape Auto Loaders & Libraries <|endoftext|>

Appendix - 2.3 Fine-Tuning Training Setup

Models	Trainer Class	Training Device	Inference Device	Training Parameters	LoRa Parameters
Pythia 90m & Pythia 410m	HF Trainer	T4	T4	num_train_epochs=3, per_device_train_batch_size=8, per_device_eval_batch_size=8, gradient_accumulation_steps=2, warmup_steps=50, weight_decay=0.01, logging_steps=100, eval_steps=500, save_steps=500, eval_strategy='steps', save_strategy='steps', logging_strategy='steps', fp16=True, bf16=False, dataloader_num_workers=4, metric_for_best_model='eval_loss'	Not applicable

Continued on next page

Appendix

Models	**Trainer Class**	**Training Device**	**Inference Device**	**Training Parameters**	**LoRa Parameters**
LLaMa 1B, LLaMa 3B	UnslothTrainer	A100	T4	per_device_train_batch_size=64, per_device_eval_batch_size=64, eval_accumulation_steps=4, gradient_accumulation_steps=4, warmup_steps=5, num_train_epochs=3, learning_rate=5e-5, embedding_learning_rate=5e-5, fp16=not is_bfloat16_supported(), bf16=is_bfloat16_supported(), fp16_full_eval=not is_bfloat16_supported(), bf16_full_eval=is_bfloat16_supported(), optim="adamw_8bit", weight_decay=0.01, lr_scheduler_type="linear", seed=2137, logging_steps=20, eval_steps=20, save_steps=20, output_dir="outputs", report_to="none", logging_dir='logs', eval_strategy='steps', save_strategy='steps', logging_strategy='steps', load_best_model_at_end=True, metric_for_best_model='eval_loss'	r=64, alpha=32, dropout=0, use_gradient_checkpointing='unsloth', bias='none', random_state=2137, target_modules=["q_proj", "k_proj","v_proj", "o_proj","gate_proj", "up_proj","down_proj"]
Mistral 7B	UnslothTrainer	A100	L4	per_device_train_batch_size=64, per_device_eval_batch_size=64, eval_accumulation_steps=4, gradient_accumulation_steps=4, warmup_steps=5, num_train_epochs=3, learning_rate=5e-5, embedding_learning_rate=5e-5, fp16=not is_bfloat16_supported(), bf16=is_bfloat16_supported(), fp16_full_eval=not is_bfloat16_supported(), bf16_full_eval=is_bfloat16_supported(), optim="adamw_8bit", weight_decay=0.01, lr_scheduler_type="linear", seed=2137, logging_steps=20, eval_steps=20, save_steps=20, output_dir="outputs", report_to="none", logging_dir='logs', eval_strategy='steps', save_strategy='steps', logging_strategy='steps', load_best_model_at_end=True, metric_for_best_model='eval_loss'	r=16, alpha=8, dropout=0, use_gradient_checkpointing='unsloth', bias='none', random_state=2137, target_modules=["q_proj", "k_proj","v_proj","o_proj", "gate_proj", "up_proj","down_proj"]