Prompt Tuned Embedding Classification for Industry Sector Allocation

Valentin Leonhard Buchner[1,2*] **Lele Cao**[1*] **Jan-Christoph Kalo**[2,3] **Vilhelm von Ehrenheim**[1]
[1]Motherbrain, EQT Group, Stockholm, Sweden
[2]Vrije Universiteit Amsterdam [3]University of Amsterdam
{valentin.buchner,lele.cao,vilhelm.vonehrenheim}@eqtpartners.com j.c.kalo@uva.nl

Abstract

We introduce Prompt Tuned Embedding Classification (PTEC) for classifying companies within an investment firm's proprietary industry taxonomy, supporting their thematic investment strategy. PTEC assigns companies to the sectors they primarily operate in, conceptualizing this process as a multi-label text classification task. Prompt Tuning, usually deployed as a text-to-text (T2T) classification approach, ensures low computational cost while maintaining high task performance. However, T2T classification has limitations on multi-label tasks due to the generation of non-existing labels, permutation invariance of the label sequence, and a lack of confidence scores. PTEC addresses these limitations by utilizing a classification head in place of the Large Language Models (LLMs) language head. PTEC surpasses both baselines and human performance while lowering computational demands. This indicates the continuing need to adapt state-of-the-art methods to domain-specific tasks, even in the era of LLMs with strong generalization abilities.

1 Introduction

Investors leveraging thematic investment strategies concentrate their efforts on specific industry sectors, such as "Circular Economy." This strategy involves compiling a comprehensive list of companies within these sectors by analyzing unstructured natural language data on platforms such as Pitchbook (2024) and Crunchbase (2024). For instance, investors might utilize the description and associated keywords of a company like "Vinted" to identify the industries it operates in. In this context, machine learning can be instrumental by framing this as a multi-label text classification challenge: given a natural language description of a company X, the goal is to categorize it into one or more industries from a predefined industry sector taxonomy $T = \{Y_1, Y_2, \ldots, Y_n\}$.

While there exist various machine learning solutions for multi-label text classification, this industrial application encompasses some challenges:

- **Scarce annotations:** The annotation process, carried out by investment professionals familiar with a firm's taxonomy, results in only a limited number of labeled examples. Given that an industry taxonomy may include up to 300 industries, there are only few annotations per industry.
- **Imbalanced annotations:** Annotations are primarily focused on investment opportunities relevant to the annotator's industry of interest, leading to a long-tail distribution.
- **Large and heterogeneous inference dataset:** The necessity to infer industries for over $10M$ companies, coupled with the likelihood of the inference data being out-of-distribution compared to the annotated dataset in terms of language use and descriptiveness.
- **Dynamic taxonomy and training data:** Frequent updates in industry taxonomy, company information, and annotations necessitate ongoing re-training and inference processes.

Traditional text classification approaches demand large amounts of annotated training data and often struggle to generalize effectively to novel data (Srivastava et al., 2023). Large Language Models (LLMs) exhibit superior generalization capabilities to unseen data and can be fine-tuned on smaller annotated datasets (Raffel et al., 2020). However, fine-tuning LLMs may lead to the undesirable phenomenon of "catastrophic forgetting" of pretraining knowledge (Chen et al., 2020), and is computationally demanding. These challenges can be mitigated through Parameter-Efficient Fine-Tuning (PEFT, Ding et al., 2023) techniques such as Prompt Tuning (PT). PT minimizes the number

[*]Corresponding authors. The source code is publicly available at https://github.com/EQTPartners/PTEC.

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track), pages 108–118

of parameters that need fine-tuning by focusing on a *soft prompt* appended to the tokenized and embedded input text, thus reducing computational costs and preserving the pretrained knowledge of the LLM, as the main body of the LLM's parameters remains unaltered (Tam et al., 2022; Tu et al., 2022; Lester et al., 2021). Hence, PT emerges as a viable solution for computational efficiency and knowledge retention in LLM applications.

This research evaluates the scalability, efficiency, and performance of PT in a real-world industry classification scenario, benchmarked against common baseline methods. However, PT as a text-to-text (T2T) classification approach encounters limitations on multi-label tasks as discussed in Subsection 2.4. We enhance PT by (1) integrating constrained decoding using Trie Search (Yang et al., 2023; De Cao et al., 2020) and (2) replacing the language model head with a specialized classification head. Our key contributions include:

- The adaptation of the Trie Search decoding method (Yang et al., 2023), preventing repetitive prediction of the same label, akin to the approach in (Chen et al., 2018).
- The introduction of Prompt Tuned Embedding Classification (PTEC), which concurrently optimizes the *soft prompt* and the classification head with differential learning rates.
- A comparative analysis of the performance and computational requirements of the proposed and baseline methods on two datasets: our proprietary *IndustrySector* classification task and the publicly available *HateSpeech* classification benchmark.
- Empirical evidence demonstrating that evaluating PTEC on data it has more pretraining knowledge about does not lead to an overestimation of the its classification performance when applied to data it has less pretraining knowledge about.

The paper first outlines existing text classification methodologies and their limitations. We then introduce constrained Trie Search decoding and PTEC as potential solutions to these limitations. Subsequently, we describe our experimental setup and compare the efficiency and performance of current and proposed methods. Our codebase and the *HateSpeech* dataset can be accessed at `https://github.com/EQTPartners/PTEC`.

2 Related Methods

2.1 Parameter-Free Classification with gzip

A very simple approach to text classification makes use of compression algorithms such as gzip (Jiang et al., 2023). This method leverages the principle of lossless compression, where frequently occurring symbols are encoded with shorter codes. Similar texts are likely to have more common symbols, leading to a shorter compressed length when concatenated. This phenomenon forms the basis for a low-computation distance metric for nearest-neighbors classification methods.

2.2 In-Context Learning

In-Context Learning (ICL), or N-shot prompting, involves prepending N input-output example pairs to the prompt before the actual input (Brown et al., 2020; Min et al., 2022). This method is particularly appealing for text classification as it obviates the need for any LLM fine-tuning.

2.3 Embedding Proximity

Another approach to text classification not requiring LLM fine-tuning uses text embeddings generated with LLMs. These can be used as input features for a separate classification model. The most parameter-efficient classification models are K-Nearest Neighbors (KNN) or Radius Nearest Neighbors (RadiusNN) (Guo et al., 2003; Cover and Hart, 1967). Alternatively, text embeddings can be used as input to a classification layer, which can be trained to perform the respective classification task (Kowsari et al., 2019).

2.4 Prompt Tuning

To emulate fine-tuning's effectiveness with reduced computational expense, various Parameter-Efficient Fine-Tuning (PEFT) techniques have been developed. These include Pattern-Exploiting Training (Schick and Schütze, 2021), Prefix-Tuning (Li and Liang, 2021), Low-Rank Adaptation (LoRa, Hu et al., 2021), and Prompt Tuning (Lester et al., 2021; Liu et al., 2022; Tam et al., 2022). These methods limit trainable parameters compared to full LLM fine-tuning. PT involves training the smallest amount of parameters ($< 0.1\%$), while still being reported to outperform fine-tuning (Liu et al., 2021). It prepends a *soft prompt* — a sequence of virtual token embeddings — to the token embeddings of the input text, as depicted in Fig. 1. During this process, only the

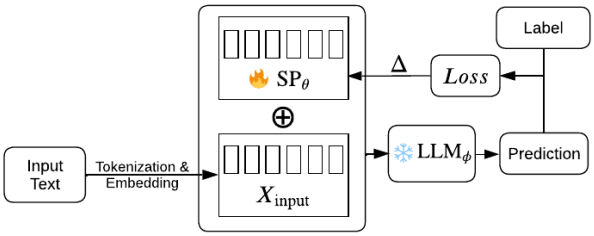

Figure 1: Schematic overview of Prompt Tuning, showing the trainable *soft prompt* (matrix SP_θ), the tokenized and embedded input text (X_{input}), and the LLM with frozen parameters (LLM_ϕ).

soft prompt undergoes training, leaving the LLM's parameters unchanged. This approach not only demands fewer computational resources but also supports multi-task processing in a single batch and mitigates the risk of "catastrophic forgetting."

2.5 T2T Classification for Multi-Label Tasks

Text-to-Text (T2T) classification leverages generative language models to produce the token(s) representing target categories. Historically, T2T has surpassed other methods in public benchmarks, aligning with the notion that text generation closely mirrors the LLM's pretraining tasks (Raffel et al., 2020). For multi-label scenarios, T2T classification sequentially generates labels, separated by a separator token (SEP) and concluded with an end-of-sequence (EOS) token (Yang et al., 2018, 2023). However, this approach faces several limitations: (a) The model might generate semantically similar but incorrect labels due to non-intuitive class labels. For instance, in our proprietary taxonomy, the model could misclassify "Healthcare IT" as "Healthcare Software". (b) In multi-label instances, labels must be provided in an arbitrary order during fine-tuning. If the model's correct label predictions deviate from this order, it is penalized by the loss function. Augmenting the label order at random would result in an inconsistent learning signal and unstable convergence. (c) The model computes the probability of a subsequent label based on the previously decoded label, expressed as $P(Y_2|X, Y_1)$, where X is the input and Y_i represents the i-th label (Simig et al., 2022). This approach fails to provide independent confidence scores for each label $P(Y_2|X)$, which are vital in real-world applications for balancing the trade-off between false positives and false negatives. Additionally, this limitation does not allow for achieving optimal performance in metrics like Precision@K, which depend on label probabilities.

3 Proposed Methods

3.1 Prompt Tuning + Trie Search

To address limitation (a) as detailed in Section 2.5, constrained decoding methods such as Trie Search, which are effective in generating only valid labels, can be employed (De Cao et al., 2020; Yang et al., 2023). Trie Search, a constrained decoding method, utilizes a label trie structure for organizing target labels, as illustrated in Fig. 2. The label trie, beginning from the root node (BOS) and ending at leaf nodes (EOS or SEP), enables valid label retrieval during label generation by guiding the LLM to select tokens only from the trie. In the context of multi-label classification, labels are generated sequentially and separated by the SEP token. Upon reaching a leaf node, the LLM chooses either to generate the SEP token, restarting the Trie Search, or the EOS token, concluding label prediction. However, this method may lead to repetitive generation of the same label, a known issue with LLMs (Fu et al., 2021). To mitigate this, our approach extends the Trie Search method by removing a label from the trie once it is generated, an idea inspired by (Chen et al., 2018). While this method effectively addresses limitations (a), it does not resolve limitations (b) and (c) since it requires labels provided in an arbitrary order during training and does not allow the calculation of appropriate confidence scores.

3.2 Prompt Tuned Embedding Classification

PTEC addresses all limitations by combining PT with Embedding Classification rather than T2T classification. This is done by using a single linear layer with a sigmoid activation function to process the text embeddings generated by the Prompt Tuned LLM. This layer produces a probability distribution over industry sectors in the taxonomy, thus (a) ensuring valid industry selection, (b) enabling the application of label order-independent loss functions, and (c) providing probability scores useful for ranking or adjusting model prediction sensitivity. This process is mathematically represented as:

$$p = \begin{cases} 1 & \text{if } \sigma(\mathbf{W}\text{LLM}_\phi(\text{SP}_\theta \oplus \mathbf{X}_{\text{input}}) + \mathbf{b}) \geq \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Here, $\text{LLM}_\phi(\text{SP}_\theta \oplus \mathbf{X}_{\text{input}})$ parameterized by ϕ yields an embedding vector. The tokenized and embedded input text is represented by $\mathbf{X}_{\text{input}}$, and τ is

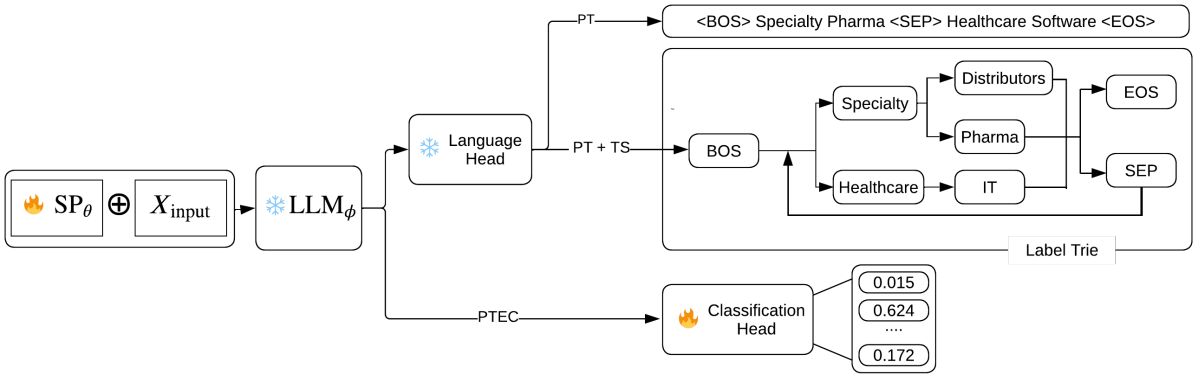

Figure 2: A schematic comparison of Prompt Tuning with T2T classification (PT + T2T), Prompt Tuning with Trie Search (PT + TS), and PTEC. Note that *Healthcare Software* would not be a valid label name, while *Healthcare IT* would be.

the threshold used. The weight matrix $\mathbf{W} \in R^{d \times l}$ and bias vector $\mathbf{b} \in R^{l}$ are components of the linear layer, with d representing the dimensionality of the LLM's embedding vector and l the number of labels. During training, the task-specific classification layer and the *soft prompt* are optimized concurrently, while the rest of the LLM's parameters are kept frozen. This approach is akin to strategies used in Named Entity Recognition (Liu et al., 2022) and multi-class text classification (Hambardzumyan et al., 2021). Following the observation by Lester et al. (2021), we found that a *soft prompt* typically benefits from a higher learning rate, while the classification head performs optimally with a lower rate. Hence, in our PTEC implementation, differential learning rates are applied to the *soft prompt* and the classification head. Besides addressing the limitations listed in Section 2.4, PTEC offers the advantage of faster inference times, requiring only a single forward pass per prediction compared to one forward pass for each predicted token.

4 Experiments

4.1 Dataset

Based on an investment firm's proprietary database we constructed the *IndustrySector* dataset of around 5500 companies. Each company is annotated with 1 to 4 of 76 different industries, and each industry is labeled at least 20 times. For each company, its legal name, keywords, and a description are available. This information is concatenated to one text used as the input prompt in all experiments. Appendix A.2 describes dataset analytics and preprocessing steps. To facility reproducibility, we further constructed the public *HateSpeech* benchmark, which is elaborated on in Appendix A.5.

4.2 Model Training

Our PT set-up follows the architecture described in Section 3. Since for T2T classification the labels need to be provided in a predefined order during training, we sort the labels for each sample descending by their frequency in the training data as this has been confirmed to provide the best performance (Yang et al., 2018; Jung et al., 2023). We noticed that classes with class labels consisting of more tokens have more influence on the cross-entropy loss than classes with shorter labels. Consequently, we developed the Normalized Token Entropy (NTE) Loss, which is motivated and elaborated on in Appendix A.3. Further, we use token embeddings of the target classes to initialize the *soft prompt*'s weights, as Lester et al. (2021) showed this to be beneficial for task performance. As there are more tokens available for the target classes than there are tokens in the *soft prompt*, we randomly sample the tokens to be used for *soft prompt* initialization. All methods are compared using the $7B$ parameter version of LLaMa (LLaMa 7B, Touvron et al., 2023) and the $1.7B$ parameter version of Bloom (Bloom 1B7, Scao et al., 2022). A detailed description of our hyperparameter tuning strategy can be found in Appendix A.4.

4.3 Metrics

To achieve optimal business impact, it is crucial to predict all industry sectors similarly well. This enables an investment firm to not only find companies in well-explored sectors but also in novel or niche sectors. Consequently, we use the macro-averaged F1 score to compare model performance. Further, it becomes important to be cost-effective when frequently retraining and running inference over a large database. Therefore, we report on the com-

putational resources required for fine-tuning and for inference over $10M$ companies by measuring the consumed floating point operations (FLOPs). These were measured using Pytorch's profiler (PyTorch, 2024) for a representative sample of batches, and the results were extrapolated on the full training and inference process. The FLOPs consumption of KNN and RadiusNN were estimated as motivated in Appendix A.1. To investigate the subjectivity of this industry classification task, an exhaustive list of labels was created for a representative subsample of the test set ($N = 104$) and annotated by 3 independent professional raters. Chance-corrected inter-annotater agreement was calculated using Cohen's kappa (κ, McHugh, 2012).

4.4 The Impact of Pretraining Knowledge

Companies in our *IndustrySector* dataset were annotated depending on investment professionals' interests and are not a representative sample of the inference dataset. On the contrary, investment professionals are more likely to annotate companies that are more widely known, which are companies the LLM may have encountered during pretraining. The LLM may thus perform the desired downstream task better for the annotated companies in our test set than for the full set of less-known companies in the inference dataset, resulting in an overestimation of model performance. To investigate whether this is the case, we prompted an LLM to indicate about which companies in the test set it has pretraining knowledge, following the logic that LLMs mostly know what they know (Kadavath et al., 2022). We then conducted a nonparametric Mann-Whitney U test (Nachar et al., 2008) to test the hypothesis H_1 that *classification performance is higher for the companies the LLM indicates to have pretraining knowledge about*.

5 Results

5.1 Performance and Computational Cost

The computational efficiency and average performance over 3 runs of various methods on the *IndustrySector* dataset are presented in Table 1. PTEC shows an improvement ranging from 3.6 to 11.7 percentage points over the next best method while being more efficient than other PT methods for both training and inference. Additionally, PTEC shows less variability between runs than PT with T2T classification, particularly for Bloom 1B7.

Contrasting prior findings where T2T classification outperformed classification heads (Raffel et al., 2020), PTEC outperforms PT + T2T in our study. Several arguments can be made to explain this: (1) T2T classification often outperforms because the LLM can make a reasonable guess. However, the proprietary and domain-specific nature of the industry taxonomy limits the LLM's ability to leverage its pretraining knowledge. (2) While most tasks used to evaluate T2T classification can be reduced to singular-token targets ("good" or "bad"), the *IndustrySector* dataset consists of multi-token labels and therefore presents a more complex label space.

Trie Search enhances T2T classification performance by 0.17 to 10.9 percentage points with N-shot prompting. However, it does not improve LLaMa 7B's performance when used with PT, suggesting that PT effectively learns to predict valid labels such that Trie Search does not result in any additional performance gain.

Classification heads demonstrate comparable performance to PT with T2T classification but are significantly more computationally efficient. While N-shot prompting eliminates training FLOPs, it necessitates a higher number of inference FLOPs. Table 2 summarizes the techniques each method employs. Results on our public *HateSpeech* benchmarking dataset followed nearly the same pattern and can be inspected in Appendix A.5.

	Method	FLOPs		Macro F1	
		Training	Inference	Mean	Std
Bloom 1B7	PTEC	1.12e+17	1.09e+18	**0.398**	0.019
	PT + TS	8.96e+16	1.65e+18	0.240	0.060
	PT	8.96e+16	1.65e+18	0.221	0.068
	CH	3.29e+16	**3.97e+17**	0.281	0.006
	KNN	3.29e+16	**3.97e+17**	0.230	0
	RadiusNN	3.29e+16	**3.97e+17**	0.101	0
	N-shot + TS	**0**	8.51e+18	0.134	0.004
	N-shot	**0**	5.68e+18	0.025	0.005
LLaMa 7B	PTEC	1.69e+17	4.27e+18	**0.448**	0.001
	PT + TS	9.73e+17	5.62e+18	0.412	0.005
	PT	9.73e+17	5.62e+18	0.412	0.002
	CH	2.13e+17	**2.56e+18**	0.400	0.007
	KNN	2.13e+17	**2.56e+18**	0.332	0
	RadiusNN	2.13e+17	**2.56e+18**	0.237	0
	N-shot + TS	**0**	2.59e+19	0.032	0.001
	N-shot	**0**	2.55e+19	0.015	0.002
-	gzip	–	–	0.271	0

CH = classification head; gzip = parameter-free classification with gzip. Other abbreviations as defined in Fig. 2.

Table 1: Results on the *IndustrySector* dataset. The method with the lowest FLOPs and highest Macro F1 Score is highlighted in **bold** for each LLM. A dash (–) indicates unavailable data or no LLM required.

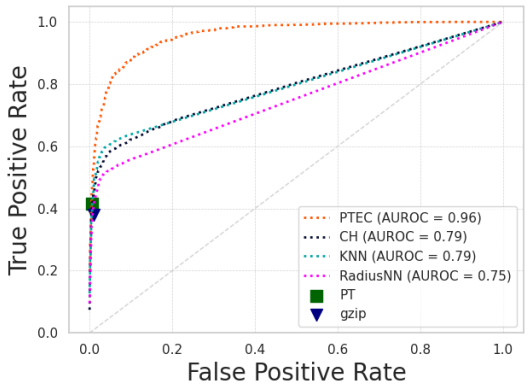

Figure 3: ROC curves using LLaMa 7B. Methods that cannot be thresholded are displayed as individual points. AUROC = Area Under the ROC curve. Other abbreviations as defined in Fig. 2 and Table 1.

	valid labels	order invariant	conf. scores	LLM tuning	Macro F1
N-shot		✓			0.015
N-shot + TS	✓	✓			0.032
RadiusNN	✓	✓	✓		0.237
KNN	✓	✓	✓		0.332
CH	✓	✓	✓		0.4
PT + T2T				✓	0.412
PT + TS	✓			✓	0.412
PTEC	✓	✓	✓	✓	**0.448**

Abbreviations as defined in Fig. 2 and Table 1

Table 2: Overview of methods used and their performance on the *IndustrySector* dataset using LLaMa 7B. The highest F1 score is highlighted in **bold**.

Methods such as PTEC offer the advantage of predicting appropriate confidence scores. This attribute is evident in Fig. 3, which displays the Receiver Operating Characteristic (ROC) curves for multiple methods. These confidence scores allow for selecting a threshold to choose the appropriate trade-off between precision and recall, a crucial attribute for deploying a model in production.

5.2 The Impact of Pretraining Knowledge

In the *IndustrySector* dataset's test split, 159 of the 839 companies were recognized from pretraining, while 680 were not. A qualitative review confirmed that known companies had more accessible online information than unknown companies. A Mann-Whitney U test indicated that differences in task performance using LLaMa 7B between both groups were nonsignificant at a p-value of 0.243 (U = 50993.5; r = 0.0385). This results in the rejection of H_1 that *classification performance is higher for the companies the LLM indicates to have pretraining knowledge about*. This indicates that we likely do not overestimate performance on the inference dataset.

	Rater2	Rater3	Gold	PTEC	$\Delta_{\text{Gold}-\text{PTEC}}$ [a]
Rater1	0.477	0.401	0.389	0.36	0.029
Rater2		0.444	0.551	0.422	0.129
Rater3			0.311	0.245	0.066
Average			0.417	0.342	0.075
Gold				0.562	

[a] the difference in agreement of a given rater with the gold annotations and the PTEC predictions.

Table 3: Agreement Matrix using Cohen's Kappa comparing three independent human raters, gold labels, and predictions made with PTEC LLaMA 7B.

5.3 Inter-rater Agreement

Table 3 displays the interrater agreement between three independent human raters, the gold labels used to train PTEC, and PTEC predictions on the subsample described in Section 4.3. The moderate agreement between human raters verifies the subjectivity of our *IndustrySector* classification task. Out of 104 companies, unanimous agreement was reached on just 6 companies. Importantly, PTEC's agreement with the gold labels is up to 15.1 percentage points higher than the agreement between human raters and the gold labels. This shows that PTEC outperforms human professionals, meaning that it provides value by accelerating and objectifying the industry classification process.

6 Conclusion

This study benchmarks computational cost and multi-label text classification performance of PT as a parameter-efficient alternative to fine-tuning all LLM parameters. To address the limitations of a T2T approach on multi-label classification problems, PT is extended with Trie Search as a constrained decoding strategy, and with Embedding Classification as an alternative to T2T classification. Results indicate that Trie Search can significantly improve the performance of N-shot prompting. PT can outperform popular text classification approaches on both our domain-specific *IndustrySector* classification task, and the publicly released *HateSpeech* classification benchmark. Both performance and efficiency can be further improved by combining PT with Embedding Classification. The proposed solution, PTEC, outperforms baselines and human professionals and can be deployed at scale to accelerate and objectify industry sector allocation.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881.

Shang-Fu Chen, Yi-Chen Chen, Chih-Kuan Yeh, and Yu-Chiang Wang. 2018. Order-free rnn with visual attention for multi-label classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Crunchbase. 2024. Crunchbase. Accessed: 2024-01-22.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. 2021. A theoretical analysis of the repetition problem in text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12848–12856.

Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. 2003. Knn model-based approach in classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, pages 986–996. Springer.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online. Association for Computational Linguistics.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiying Jiang, Matthew Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, and Jimmy Lin. 2023. Low-resource text classification: A parameter-free classification method with compressors. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6810–6828.

Taehee Jung, Joo-Kyung Kim, Sungjin Lee, and Dongyeop Kang. 2023. Cluster-guided label generation in extreme multi-label classification. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1662–1677.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information*, 10(4):150.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

X Liu, Y Zheng, Z Du, M Ding, Y Qian, Z Yang, and J Tang. 2021. Gpt understands, too. arxiv. *arXiv preprint arXiv:2103.10385*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064.

Nadim Nachar et al. 2008. The mann-whitney u: A test for assessing whether two independent samples come from the same distribution. *Tutorials in quantitative Methods for Psychology*, 4(1):13–20.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pitchbook. 2024. Pitchbook. Accessed: 2024-01-22.

PyTorch. 2024. Pytorch. Accessed: 2024-01-22.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Joni Salminen, Hind Almerekhi, Milica Milenković, Soon-gyo Jung, Jisun An, Haewoon Kwak, and Bernard Jansen. 2018. Anatomy of online hate: developing a taxonomy and machine learning models for identifying and classifying hate in online news media. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 12.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352.

Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2011. On the stratification of multi-label data. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part III 22*, pages 145–158. Springer.

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

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Derek Tam, Anisha Mascarenhas, Shiyue Zhang, Sarah Kwan, Mohit Bansal, and Colin Raffel. 2023. Evaluating the factual consistency of large language models through news summarization. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5220–5255.

Weng Lam Tam, Xiao Liu, Kaixuan Ji, Lilong Xue, Xingjian Zhang, Yuxiao Dong, Jiahua Liu, Maodi Hu, and Jie Tang. 2022. Parameter-efficient prompt tuning makes generalized and calibrated neural text retrievers. *arXiv preprint arXiv:2207.07087*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Lifu Tu, Caiming Xiong, and Yingbo Zhou. 2022. Prompt-tuning can be much better than fine-tuning on cross-lingual understanding with multilingual language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5478–5485, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. Sgm: Sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3915–3926.

Zhichao Yang, Sunjae Kwon, Zonghai Yao, and Hong Yu. 2023. Multi-label few-shot icd coding as autoregressive generation with prompt. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5366–5374.

A Appendix

A.1 Inference FLOPs Calculation for Nearest-Neighbors Methods

KNN and RadiusNN were implemented using sklearn (Pedregosa et al., 2011). There is to our knowledge no existing method to measure their FLOPs consumption for nearest-neighbor methods implemented with sklearn during inference. Instead, their inference FLOPs were estimated as:

$$\text{FLOPs} \approx E(T+I) + 3(D \cdot T \cdot I) \qquad (2)$$

Here, D represents the dimensionality of the text embeddings, T denotes the number of training samples, I indicates the number of inference samples, and E the FLOPs required to embed one example. This equation can be derived as follows: The term $E(T+I)$ refers to calculating the embeddings for the training and inference samples, and $3(D \cdot T \cdot I)$ estimates the number of floating point operations (FLOPs) for performing classification with the KNN and RadiusNN algorithms. The average value of E is calculated by measuring the FLOPs used for generating one embedding with PyTorch's profiler. Assuming a brute-force implementation, for both KNN and RadiusNN, each inference embedding is compared with every training embedding. The term $3 \cdot D$ corresponds to calculating the Euclidean distance between two embeddings. This calculation involves the subtraction of one embedding from the other (D FLOPs), squaring each element of the new vector (D FLOPs), taking the sum of these values ($D-1$ FLOPs) and finally taking the square root of this sum (1 FLOP). As this is done once for each pair of training and inference examples, the distance calculations will need $3(D \cdot N \cdot M)$ FLOPs in total.

As this is only an estimate, the exact number can vary based on the specifics of the operations used. While the formula provided here assumes a brute-force method for KNN and RadiusNN, it is important to note that more efficient methods are often employed in practice, especially in popular machine learning libraries such as scikit-learn (Pedregosa et al., 2011). True computational resources required by KNN and RadiusNN methods may therefore be lower than estimated in this paper. However, this estimation provides a general idea of the computational resources needed. For both RadiusNN and KNN the FLOPs used for calculating the text embeddings of the training data are considered as 'training' FLOPs.

A.2 IndustrySector Dataset Preprocessing

The average number of labels in the *IndustrySector* dataset per example is 1.1. This indicates that while the problem, in theory, is a multi-label classification problem, most examples in our dataset are not exhaustively annotated and only carry one label (see Fig. 4). The dataset is split into 75% training set, 10% validation set, and 15% test set. Fig. 4 shows the highly imbalanced, long-tail class distribution: some industries occur only ~ 25 times, while the most frequent industry occurs > 300 times. Importantly, this distribution only shows the classes included in the *IndustrySector* dataset, and our database contains many more classes with even fewer annotations. To ensure that each industry in the *IndustrySector* dataset is represented in similar proportions in all splits, and with a minimum frequency in both validation and test split, stratification is performed using multi-label stratified shuffle splitting, as proposed by Sechidis et al. (2011). During this process, it is ensured that each industry is represented at least 2 times in the validation set, 3 times in the test set, and 15 times in the training set. The imbalanced annotations were accounted for by reweighing the loss: Class weights are calculated for each class with $n_{\text{max}}/n_{\text{class}}$. The loss for each instance is weighted by its class weight before updating the gradients.

Since the LLM's self-attention mechanism's complexity increases quadratically with prompt length, long input prompts will easily result in out-of-memory (OOM) errors. Therefore, descriptions and keyword lists that consist of more than 1000 characters are summarized using the $250M$ parameter instruction fine-tuned FLAN T5 model (Chung et al., 2022), such that no input prompt supersedes a length of 1000 characters. The result of this summarizing step is displayed in Fig. 4.

A.3 Normalized Token Entropy (NTE) Loss

Careful attention has to be paid to the loss calculation when performing mini-batch gradient descent. As PyTorch's (PyTorch, 2024) cross-entropy loss function by default averages the loss over all label tokens in a batch, industries with names consisting of more tokens ("Circular Economy & Sustainable Materials") have a larger influence on the batch loss than industries with shorter names ("Marketplaces"). This results in the model learning industries with longer names better than industries with shorter names. To avoid this, we adjust the cross-

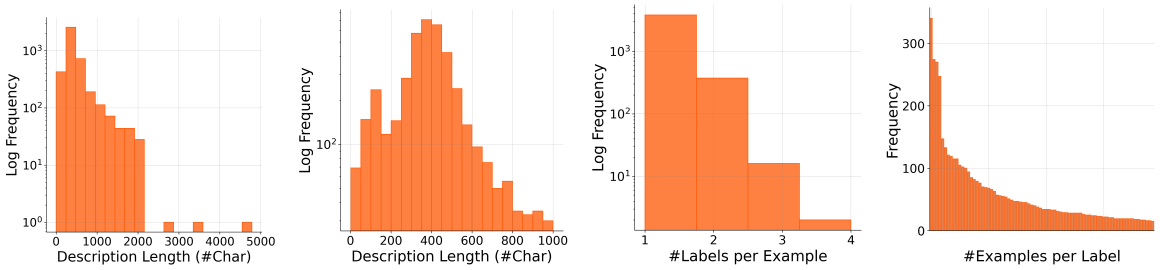

Figure 4: Distributions of (a) original description lengths, (b) preprocessed description lengths, (c) number of labels per example, and (d) number of examples per label

entropy loss calculation such that each label has the same influence on the batch loss by reweighting the influence that each token has on the loss. This can be done by first taking the average loss of all tokens belonging to one label, and then averaging all individual losses over the batch. This is denoted in (3), where L is the aggregated loss of the batch, N is the number of examples of the batch, y_i is the label tokens for the i-th example in the batch, $|y_i|$ is the length of the label of the i-th example measured in it's number of tokens, y_{ij} is the target value of the j-th token of the i-th label, and p_{ij} is the predicted probability of the j-th token of the i-th label.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{|y_i|}\sum_{j=1}^{|y_i|} y_{ij}\log(p_{ij})\right) \quad (3)$$

Method	Hyperp	Scl	Searched Space	Value
N-shot	n	lin	$\{0, 1, ..., 8\}$	7
RadiusNN	radius	lin	$[0.1, 150]$	25.25
KNN	k	lin	$\{1, 2, ..., 150\}$	1
CH	lr	log	$[1e^{-6}, 1e^{-3}]$	$1e^{-3.58}$
	wd	log	$[0, 1e^{-3}]$	0
PT (+ TS)	SP lr	log	$[1e^{-9}, 1]$	$1e^{-1.66}$
	SP length	lin	$\{50, 51, ..., 200\}$	156
	epochs	lin	$\{5, 6, ..., 18\}$	18
PTEC	SP lr	log	$[1e^{-9}, 1]$	$1e^{-4.95}$
	SP length	lin	$\{50, 51, ..., 200\}$	53
	CH lr	log	$[1e^{-9}, 0.1]$	$1e^{-4.23}$
	wd	log	$[1e^{-9}, 0.5]$	$1e^{-8.72}$
	epochs	lin	$\{5, 6, ..., 18\}$	13

Abbreviations as defined in Fig. 2 and Table 1

Table 4: Overview of hyperparameters (hyperp), scales (scl), and search space. To ensure reproducibility, value refers to the selected value for LLaMa 7B on the public HateSpeech dataset.

A.4 Hyperparameter Tuning

The hyperparameters for all methods were optimized using Bayesian Optimization (Snoek et al., 2012) with 25 random initializations of hyperparameter combinations and 15 iterations of Bayesian Optimization. Models involving PT are trained using the AdamW optimizer. Hyperparamters such as the learning rate and weight decay were searched on a logarithmic scale, such that the probability to sample values from the interval $[0.01 \leq x \leq 0.1]$ equals the probability to sample values from $[0.001 \leq x \leq 0.01]$, given that both intervals are included in the searched hyperparameter space. For the KNN and RadiusNN methods, the optimal hyperparameter values have large variability between different models. For this reason, if a hyperparameter was close to the boundary of the searched hyperparameter space, Bayesian Optimization was continued with a broader hyperparameter range. An overview over the optimized hyperparamters, the scale of searching, and the ranges of hyperparameter values searched are provided in Table 4. Hyperparameter tuning was performed using the validation set, while all results reported in Section 5 were calculated over the test set. While the maximum batch size fitting on one A100 GPU was used for model training, an effective batch size of 32 was used for gradient updates. Threshold τ mentioned in (1) is not considered a hyperparameter, since we automatically select the value that optimized the F1 score.

A.5 Public Benchmarking

To enable reproducibility, we constructed a public benchmark from Salminen et al.'s (2018) hatespeech classification dataset. The task of this dataset is to classify social media comments into different kinds of hatespeech, where each comment can have one or multiple labels. This dataset was chosen because it is structurally similar to our *In-*

dustrySector dataset: It covers a set of 22 different classes, its data is highly imbalanced, and the length of the social media comments is similarly distributed as the length of the company descriptions. Each hate speech comment is annotated with 1 to 4 labels, and a comment has 1.45 annotations on average. It should be noted that we could only find a substantially smaller and differently distributed subset of the original dataset, implying that our results cannot directly be compared with Salminen et al. (2018). Nevertheless, this benchmark serves as a possibility to verify our methodology and results. The constructed *HateSpeech* dataset can be found in our released codebase.

We achieved very similar results to the *IndustrySector* dataset on our public *HateSpeech* dataset, as shown in Table 5. The most notable difference is that for LLaMa 7B, PT outperforms PTEC. For both models, Trie Search decreases the performance of the Prompt Tuned LLM, while it slightly improves the performance for N-shot prompting of Bloom 1B7. A relevant observation made is the high standard deviation of T2T classification performance when using Bloom 1B7. This goes along with results of recent research showing that models from the Bloom family produce the most inconsistent summaries, as judged by other language models (Tam et al., 2023).

	Method	FLOPs		Macro F1	
		Training	Inference	Mean	Std
Bloom 1B7	PTEC	6.99e+16	3.96e+17	**0.48**	0.015
	PT + TS	8.69e+16	6.85e+17	0.233	0.123
	PT	8.69e+16	7.94e+17	0.318	0.088
	CH	6.82e+12	**3.59e+17**	0.063	0.011
	KNN	8.39e+14	**3.59e+17**	0.12	0
	RadiusNN	8.39e+14	**3.59e+17**	0	0
	N-shot + TS	**0**	2.81e+18	0.082	0.002
	N-shot	**0**	2.51e+18	0.055	0.005
LLaMa 7B	PTEC	1.31e+17	2.27e+18	0.437	0.007
	PT + TS	2.22e+17	2.37e+18	0.47	0.032
	PT	2.22e+17	3.20e+18	**0.526**	0.021
	CH	3.07e+13	**1.59e+18**	0.365	0.014
	KNN	3.72e+15	**1.59e+18**	0.195	0
	RadiusNN	3.72e+15	**1.59e+18**	0.142	0
	N-shot + TS	**0**	4.40e+18	0.094	0.008
	N-shot	**0**	1.16e+19	0.107	0.021
-	gzip	–	–	0.128	0

gzip = Parameter-Free Classification with gzip. Other abbreviations as defined in Table 4.

Table 5: Experimental results on the *HateSpeech* benchmark. The method requiring the lowest FLOPs and achieving the highest macro-averaged F1 Score is highlighted in **bold** for each model. A dash (−) indicates that a value could not be estimated.