Improving AutoML for LLMs via Knowledge-Based Meta-Learning

Ernesto Luis Estevanell-Valladares[1,2]

[1]*Faculty of Mathematics and Computer Science, University of Havana*
[2]*Natural Language Processing and Information Systems Group, University of Alicante*

Abstract

Recent advancements in Large Language Models (LLMs) such as BERT, GPT-4, and T5 have revolutionized the field of Natural Language Processing (NLP), unlocking numerous applications. However, fine-tuning these models for specific tasks remains a complex and resource-intensive process, often relying heavily on expert knowledge. This research proposes integrating meta-learning into Automatic Machine Learning (AutoML) systems to optimize LLM fine-tuning and pipeline construction. We hypothesize that knowledge-based meta-learning can overcome the inefficiencies of current AutoML approaches by embedding expert-derived heuristics into the optimization process. Our methodology involves compiling extensive LLM usage data, training meta-learning estimators, and integrating these into the AutoGOAL AutoML framework. By doing so, we aim to reduce computational costs and enhance the efficiency of LLM-based NLP applications. The proposed system will be evaluated against traditional AutoML methods and human experts on various text classification tasks to validate its effectiveness. This research can further democratize NLP by making advanced LLM capabilities more accessible and efficient.

Keywords

AutoML, Large Language Model, Meta-Learning, Natural Language Processing

1. Introduction

Recent advances in large language models (LLMs), such as BERT [1], the different versions of GPT [2, 3], and others like T5 [4] or Mistral [5], have unlocked a whole new landscape of applications. With their sophisticated internal language representations, these models have demonstrated the potential to generalize across numerous tasks [6, 7, 8], thus democratizing access to advanced NLP capabilities. However, achieving satisfactory performance typically requires model fine-tuning, which involves selecting the appropriate model, fine-tuning method, and hyperparameters, often relying on researchers' prior experience and trial-and-error approaches [9].

On the other hand, Automatic Machine Learning (AutoML) [10] democratizes traditional Machine Learning (ML) by automating the process of building adequate ML pipelines for specific tasks, reducing user interaction. These systems have proven their efficacy in Model Selection (MS) [11] and Hyper-parameter Optimization (HPO) [12], showing relevant results in various ML tasks [11, 13, 14, 15]. Some systems, like AutoGOAL [15], can even tackle NLP tasks and

Doctoral Symposium on Natural Language Processing, 26 September 2024, Valladolid, Spain.
elev1@alu.ua.es (E. L. Estevanell-Valladares)
0000-0002-1168-1767 (E. L. Estevanell-Valladares)

have shown the ability to compete with manually designed models by human experts within a fraction of the time.

Building ML pipelines and LLM solutions are similar in that both depend on numerous design decisions. NLP pipelines could include multiple steps (e.g., data preprocessing, feature extraction, classification), combining algorithms and hyper-parameters that work in conjunction. On the other hand, LLMs have many life-cycle stages, each consisting of different tasks and metrics that need optimization [16]. However, it is more common (and accessible) to fine-tune an LM rather than retrain it from the beginning. This is mainly due to the massive computational cost of pre-training, the considerable availability of pretrained LLMs, and the reported performance of even dated LLMs (e.g., BERT [1], RoBERTa [6], and DistilBERT [17]) when fine-tuned.

Just as AutoML is used for building traditional ML pipelines, it can automatically create LLM pipelines or fine-tune LLMs based on pretrained models, as there is no technical difference between both types of pipelines. However, evaluating an LLM pipeline incurs a significant computational cost, and fine-tuning a model could take hours, depending on the training data and available computational resources. Additionally, the complexity of the search spaces, which include multiple LLMs, fine-tuning methods, and hyperparameters, could make zero-shot AutoML less efficient than human experts who rely on prior knowledge.

Our research proposes modeling knowledge from the fine-tuning stage of LLMs and integrating it into an AutoML process to efficiently generate optimal LLM pipelines for any specific NLP task. As such, our central hypothesis is that **(H1) knowledge-based meta-learning can mitigate the drawbacks of AutoML for LLMs** and help build LLM-based applications more effectively. To test our hypothesis, we will design, develop, and integrate such meta-learning components into an AutoML system. In particular, we will focus on the Text Classification task as it is relevant and allows for a more straightforward proposal evaluation process. Then, we will compare our meta-learning-based AutoML system against zero-shot AutoML and human experts.

1.1. Motivational Example

Imagine a mid-sized company wanting to implement an advanced customer support chatbot using pre-trained LLMs like the GPTs [2, 3] or T5 [4]. Traditionally, customizing one of these models could take weeks or months, delaying deployment and impacting productivity. Our proposed knowledge-based meta-learning approach within an AutoML framework aims to automatically predict the most suitable LLM, tuning method, and settings for the specific task.

This approach reduces time and computational resources, improving model development efficiency and quality. Integrating expert knowledge into the AutoML process can speed up the entire production pipeline and lead to faster and more effective deployment of LLM-based applications.

2. Related Work

Much research is currently being conducted to explore how AutoML and LLMs can benefit each other. [16] summarizes this symbiotic relationship from two different viewpoints: **LLMs for AutoML** and **AutoML for LLMs**.

2.1. LLMs for AutoML

The term **LLMs for AutoML** refers to using Language Models to enhance an AutoML process or system. The two most popular approaches in this category involve using LLMs to improve human interaction with AutoML systems or using the knowledge embedded in LLMs to actively contribute to the solution-building process of AutoML [16].

> **Human-to-Machine Interaction**: LLM-based applications like ChatGPT [3] from OpenAI and Gemini [18] from Google demonstrate how LLMs can be employed for human-to-machine interaction with millions of users. From this experience stems the potential of LLMs for improving user interaction with complex AutoML systems. According to Tornede et al. [16], language models could serve as the interface for setting up the necessary configurations for the AutoML system to function properly and could also facilitate some level of result interpretability.
>
> **LLMs as Controllers**: Due to the vital amount of knowledge embedded into LLMs during training, they can also be used to participate in the solution-building process of AutoML actively. Shen et al. [19] and Luo and Shen [20] proposed using LLMs as controllers for building pipelines. HuggingGPT [19] parses user inputs into sorted tasks, finds suitable huggingface [21] models for each, and computes the response orderly. AutoM3L [20] goes a step further, allowing users to have a more active role in each step of the system via directives to the LLM. Other proposals by Sayed et al. [22], Morris et al. [23], and Zhang et al. [24] also implement this type of approach.

2.2. AutoML for LLMs

Another point of interest in the relationship between AutoML and LLMs is the fact that AutoML could be used to produce optimal LLM solutions streamlined for specific scenarios automatically. This approach is known as **AutoML for LLMs** Tornede et al. [16]. However, this direction stems several challenges that must be addressed, namely:

(i) The different stages of the life-cycle of LLMs require optimization on different objectives, of which current AutoML systems are incapable.
(ii) LLMs are extremely resource-intensive [25], even when only considering their latest stages (e.g., fine-tuning, inference).

In their work, Mallik et al. [9] emphasize the gap between current HPO algorithms and modern Deep Learning (DL) methods. They introduce an HPO approach incorporating expert knowledge and inexpensive proxy tasks to reduce optimization costs. On the other hand, Zhang et al. [24] proposes AutoML-GPT, capable of optimizing LLM pipelines for many tasks. This system optimizes the hyperparameters of such pipelines by simulating their training. This way, all responsibility falls into the coordinator LLM (and collaborator models), and no actual evaluation is executed. Both methods leverage expert knowledge to minimize resource consumption during their hyperparameter optimization search.

Furthermore, Zhang et al. [26] investigated the impact of data, model, and fine-tuning method selection on various NLP tasks, concluding that the optimal approach varies depending on the task. Currently, no system combines Model Selection and HPO. Therefore, we propose an AutoML system with these specifications.

3. Proposal and Methodology

The expertise in machine learning, mainly when data is limited and training is not feasible, involves leveraging expert knowledge to navigate the complexities of ML tasks. Experts utilize scalability rules and heuristics to make informed decisions about model architecture, training data selection, and fine-tuning techniques based on the specific requirements of each task. These decisions help optimize resource usage and achieve efficient outcomes. Our proposal aims to model these heuristics within an AutoML system using meta-learning to avoid sub-optimal decisions. We propose the following specific objectives:

O1 Extract, compile, and store knowledge from AutoML logs
We will analyze AutoML logs to identify patterns and insights that can be extracted from the exploration experience. This involves collecting data on configurations, performance metrics, and outcomes of AutoML processes.

O2 Open a federated knowledge venue (Optional, Long Term)
The knowledge extracted from every AutoML instance will be transformed into a reusable format, stored, and shared across multiple devices. We can recycle all the unused knowledge on LLM experimentation by providing a logging framework connecting to the centralized knowledge base. This federated knowledge will be a foundation for training models that can be generalized across diverse tasks and settings.

O3 Train and test an estimator on such knowledge
We will develop and evaluate an estimator trained on the compiled knowledge to predict optimal configurations and settings for new tasks. Federated Knowledge is not required to test our main hypotheses but would enhance our estimators. Hence, we can simply train and test our estimator using the initially generated data.

O4 Integrate the estimator into an AutoML system
Finally, the trained estimator will be integrated into an AutoML system. This integration will enable the system to automatically apply expert-derived heuristics and avoid sub-optimal decisions, improving overall efficiency and performance.

3.1. Knowledge Compilation

The initial step involves collecting and organizing LLM usage data from various scenarios, specifically AutoML logs. Our focus will be on text classification to support the testing of our hypotheses. The data we gather will cover the following components:

- ML task specifics (text classification).
- Dataset characteristics (e.g., number of samples and classes, mean length of samples, domain).

- LLM features (e.g., number of parameters and layers, pre-training target task, pre-training data domains).
- Fine-tuning method features (e.g., method name, hyperparameters).
- Outcome metrics (e.g., performance, resource utilization).

We acknowledge that a limited amount of data is available for experiments that align with our specific requirements for fine-tuning LLMs. Additionally, many models are not open-source, making it difficult to access necessary features. Therefore, our proposal involves generating the required data for our research. At the time of writing, we have over 2000 LLM evaluation entries on three text classification tasks: IDMB, Yelp Reviews Full [27], and AG News [27].

First, we should select an appropriate set of LLMs, fine-tuning methods (with their hyperparameters), and NLP tasks to evaluate. Table 1 lists the LLMs we have selected for study participation. We amount to 44 models (accounting for variants), half of which are generative. Models range from 65.8 million parameters (DistilBERT) to 11 billion (T5).

LLM	Variants
BERT [1]	(cased, uncased) base, large, base-multilingual (only cased)
DistilBERT [17]	base (cased, uncased), base-multilingual (cased)
RoBERTa [6]	base, large
XLM-RoBERTa [28]	base, large
DeBERTa [29]	base
DeBERTaV3 [30]	base
MDeBERTaV3 [30]	base
ALBERT-v1 [31]	base, large, xlarge, xxlarge
ELECTRA [32]	(discriminator) small, base, large
T5 [4]	small, base, large, 3B, 11B
FLAN-T5 [33]	base, large, xxl, xl
GPT-2 [2]	base, medium, large, xl

Table 1
List of LLMs selected for study participation.

Fine-tuning has been the preferred choice for adapting Language Models to specific tasks [34]. However, some methods might render different results depending on their use case. For our research, we have included vanilla fine-tuning, the Low-Rank Adaptation (LoRA) adapter [35] as a Parameter Efficient Fine-Tuning alternative. Lastly, we added a naive Partial Fine-tuning method consisting of freezing the initial layers so general knowledge is not lost during training [36], a way of adaptive fine-tuning.

Because our hypothesis is domain-agnostic, we propose testing these LLMs and fine-tuning methods in Text Classification tasks. However, evaluating every possible combination is inefficient due to the high cost of experimentation and the sheer number of combinations available (taking into account fine-tuning hyperparameters). Therefore, we resort to AutoML to sample good-performing and efficient samples.

AutoGOAL[15] is a heterogeneous AutoML system capable of multi-objective optimization that includes LLMs in its algorithm pool. However, one of its limitations is that it can only employ LLMs for inference. Hence, we also need to extend the system to support fine-tuning.

Optimizing performance and training time could help us produce substantial data in the shortest possible time. Moreover, training time is a substantial estimator of how compute-intensive training certain LLM is [37]; hence, optimizing it would help steer the data towards the greener combinations. However, although theoretically, this could raise the number of samples generated in a period, we could lean onto other venues for recollecting more data.

3.1.1. Federated Knowledge and Knowledge Recycling

Due to the rise in popularity of LLMs, a massive amount of work is directed toward fine-tuning these models to specific tasks. Only Huggingface [21] hosts around 60000 models for text classification, and many of these could have been the final products of a long series of experiments that ended up under-performing or straight-out invalid. If correctly reported and utilized, this (disposed of) knowledge could be of great value for meta-learning.

We propose exploiting this venue by building a logging framework to collect relevant data from experiments regarding LLMs and store them in a centralized knowledge base. This Federated Knowledge Base could be the base of further meta-learning approaches to optimizing LLMs and potentially support many researchers.

3.2. Meta-Learning Estimator

Once we have our Dataset, we will design multiple estimators that utilize (and represent) the extracted knowledge to predict how adequate a particular combination of LLM and fine-tuning method (and hyperparameters) are for a target task. We will follow multiple strategies for generating such estimators. AutoGOAL (or any other system) could again be employed to find optimal ML pipelines for our dataset automatically. Additionally, experts will manually design some explainable solutions and add them to the pool of candidate solutions.

3.3. AutoML Integration

Depending on the chosen system optimization strategy, the integration of the meta-learning estimator into AutoML can be approached in various ways. We selected AutoGOAL as the target framework because, to our knowledge, it is the only AutoML system capable of modeling a broad search space of LLM pipelines.

AutoGOAL follows a Probabilistic Grammatical Evolution [38] strategy consisting of a cycle in which each generation produces a population of solutions (pipelines). These pipelines are then evaluated and ranked by their performance. The top solutions are then selected to shift the system's probabilistic model, from which all pipelines are sampled. This way, AutoGOAL converges into the section of the space more likely to generate optimal solutions.

A meta-learning component could determine whether an LLM pipeline should be evaluated based on its predicted performance. If the predicted performance is notably lower than the current best by a certain threshold, such evaluation could be considered a waste of resources. If not, the LLM could be trained, and its logs could be stored (or published) for later use by newer estimators.

Another potential benefit is leveraging the extracted knowledge to provide the system with an initial advantage. Specifically, we could initialize the probabilistic distribution (which is uniform by default) with a bias for the best-performing methods we previously found for similar tasks. This approach could improve the system's speed and performance in converging to optimal solutions.

4. Experiments

To challenge **H1** (See Section 1), we propose to test first whether inference based on the extracted knowledge effectively predicts new scenarios independently from AutoML. Then, we must evaluate the benefits of integrating the meta-learning component into AutoML.

4.1. Knowledge

To gauge the quality of our compiled knowledge, we must evaluate the performance of our inferred rules and estimators against our proposed baselines:

- Random Estimator.
- LLM Estimators.

Evaluating estimators can be done as evaluating any ML model. We will automatically compare each via k-fold cross-validation on our dataset. We will selectively hide LLMs and Tasks from our dataset to further support our results and test whether the estimators can generalize to unseen data points. This can also be achieved by repeating the dataset generation procedure and sampling a test dataset for a new task or with LLMs not previously included.

4.2. Meta-Learning for AutoML

To empirically test the effectiveness of our meta-learning approach, we propose comparing our meta-learning enhanced AutoGOAL against its original implementation, other AutoML systems, and human experts on text classification tasks. By doing so, we intend to test whether our tool can generalize to different, previously unseen tasks. This would also highlight the quality of our selected features for both the dataset and the models.

References

[1] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (2019) 9.

[3] OpenAI, GPT-4 Technical Report, Technical Report, 2023. ArXiv:2303.08774.

[4] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, Journal of Machine Learning Research 21 (2020) 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[5] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al., Mistral 7b, arXiv preprint arXiv:2310.06825 (2023).

[6] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019. arXiv:1907.11692.

[7] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, Q. V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, Advances in neural information processing systems 32 (2019).

[8] C. Sun, X. Qiu, Y. Xu, X. Huang, How to fine-tune bert for text classification?, in: Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18, Springer, 2019, pp. 194–206.

[9] N. Mallik, E. Bergman, C. Hvarfner, D. Stoll, M. Janowski, M. Lindauer, L. Nardi, F. Hutter, Priorband: Practical hyperparameter optimization in the age of deep learning, Advances in Neural Information Processing Systems 36 (2024).

[10] F. Hutter, L. Kotthoff, J. Vanschoren, Automated Machine Learning, Springer, 2019.

[11] C. Thornton, F. Hutter, H. H. Hoos, K. Leyton-Brown, Auto-weka: combined selection and hyperparameter optimization of classification algorithms, ACM, 2013, pp. 847–855. doi:10.1145/2487575.2487629.

[12] M. Feurer, F. Hutter, Hyperparameter Optimization, Springer International Publishing, Cham, 2019, pp. 3–33. URL: https://doi.org/10.1007/978-3-030-05318-5_1. doi:10.1007/978-3-030-05318-5_1.

[13] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, F. Hutter, Auto-sklearn 2.0: The next generation, arXiv: Learning (2020).

[14] E. LeDell, S. Poirier, H2o automl: Scalable automatic machine learning, in: Proceedings of the AutoML Workshop at ICML, volume 2020, 2020.

[15] S. Estevez-Velarde, Y. Gutiérrez, A. Montoyo, Y. A. Cruz, Automatic discovery of heterogeneous machine learning pipelines: An application to natural language processing, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 3558–3568.

[16] A. Tornede, D. Deng, T. Eimer, J. Giovanelli, A. Mohan, T. Ruhkopf, S. Segel, D. Theodorakopoulos, T. Tornede, H. Wachsmuth, et al., Automl in the age of large language models: Current challenges, future opportunities and risks, arXiv preprint arXiv:2306.08107 (2023).

[17] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. arXiv:1910.01108.

[18] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al., Gemini: a family of highly capable multimodal models, arXiv preprint arXiv:2312.11805 (2023).

[19] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, Y. Zhuang, Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface, arXiv preprint arXiv:2303.17580 (2023).

[20] D. Luo, Y. Shen, Autom3l: Automated multimodal machine learning with large language model (2023).

[21] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al., Huggingface's transformers: State-of-the-art natural language processing, arXiv preprint arXiv:1910.03771 (2019).

[22] E. Sayed, M. Maher, O. Sedeek, A. Eldamaty, A. Kamel, R. El Shawi, Gizaml: A collaborative meta-learning based framework using llm for automated time-series forecasting., in: EDBT, 2024, pp. 830–833.

[23] C. Morris, M. Jurado, J. Zutty, Llm guided evolution-the automation of models advancing models, arXiv preprint arXiv:2403.11446 (2024).

[24] S. Zhang, C. Gong, L. Wu, X. Liu, M. Zhou, Automl-gpt: Automatic machine learning with gpt, arXiv preprint arXiv:2305.02499 (2023).

[25] N. Bannour, S. Ghannay, A. Névéol, A.-L. Ligozat, Evaluating the carbon footprint of nlp methods: a survey and analysis of existing tools, in: Proceedings of the second workshop on simple and efficient natural language processing, 2021, pp. 11–21.

[26] B. Zhang, Z. Liu, C. Cherry, O. Firat, When scaling meets llm finetuning: The effect of data, model and finetuning method, 2024. URL: https://arxiv.org/abs/2402.17193. arXiv:2402.17193.

[27] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, Advances in neural information processing systems 28 (2015).

[28] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, Unsupervised cross-lingual representation learning at scale, 2020. arXiv:1911.02116.

[29] P. He, X. Liu, J. Gao, W. Chen, Deberta: Decoding-enhanced bert with disentangled attention, in: International Conference on Learning Representations, 2021. URL: https://openreview.net/forum?id=XPZIaotutsD.

[30] P. He, J. Gao, W. Chen, Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2023. arXiv:2111.09543.

[31] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, ALBERT: A lite BERT for self-supervised learning of language representations, CoRR abs/1909.11942 (2019). URL: http://arxiv.org/abs/1909.11942. arXiv:1909.11942.

[32] K. Clark, M.-T. Luong, Q. V. Le, C. D. Manning, Electra: Pre-training text encoders as discriminators rather than generators, arXiv preprint arXiv:2003.10555 (2020).

[33] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al., Scaling instruction-finetuned language models, Journal of Machine Learning Research 25 (2024) 1–53.

[34] H. Yang, Y. Zhang, J. Xu, H. Lu, P. A. Heng, W. Lam, Unveiling the generalization power of fine-tuned large language models, arXiv preprint arXiv:2403.09162 (2024).

[35] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, Lora: Low-rank adaptation of large language models, arXiv preprint arXiv:2106.09685 (2021).

[36] K. Prasad Varadarajan Srinivasan, P. Gumpena, M. Yattapu, V. H. Brahmbhatt, Comparative analysis of different efficient fine tuning methods of large language models (llms) in low-resource setting, arXiv e-prints (2024) arXiv–2405.

[37] X. Wang, C. Na, E. Strubell, S. Friedler, S. Luccioni, Energy and carbon considerations of fine-tuning bert, arXiv preprint arXiv:2311.10267 (2023).

[38] H.-T. Kim, C. W. Ahn, A new grammatical evolution based on probabilistic context-free grammar, in: Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems-Volume 2, Springer, 2015, pp. 1–12.