Automatically Designing Machine Learning Models out of Natural Language

Ernesto Luis Estevanell-Valladares[1,2]

[1]Department of Software and Computer Systems, University of Alicante
[2]Faculty of Mathematics and Computer Science, University of Havana

Abstract
The popularity of artificial intelligence has led to an increasing need for machine learning models tailored to specific needs. AutoML aims to automate the process of creating effective machine-learning solutions, but current systems need to be more versatile to meet the demand. While more flexible and extensible heterogeneous systems have overcome many limitations of traditional AutoML systems, they lack accessibility due to their programmatic interfaces. We propose a research project to address this issue to develop a heterogeneous AutoML system that can produce optimal machine-learning pipelines using a natural language interface.

Keywords
AutoML, Natural Language Processing, Large Language Models.

1. Introduction

Machine learning has expanded rapidly, presenting researchers and practitioners with many new algorithms and data sets. However, selecting the most suitable strategy for a given issue has become increasingly complex, requiring extensive experimentation and technical expertise. AutoML has emerged as a solution to this problem by providing powerful tools to search through large machine-learning pipelines [1]. Nevertheless, the range of possible techniques for natural language processing is vast, making it hard to combine and compare different algorithms. Thus, AutoML algorithms must agree on a standard protocol for sharing outputs as inputs for any other algorithm.

To achieve the primary goal of AutoML, the systems must have interfaces that are easy to use for those with limited computer science and machine learning knowledge. Furthermore, the systems should have strong generalization capabilities to create tools that can be utilized in various scenarios and produce machine learning models that can be applied to a wide range of applications. However, current AutoML systems focus on a specific set of algorithms, often tailored to a library or toolkit [2, 3, 4]. This reduces their ability to explore various algorithms from different domains and find optimal solutions to complex, multifaceted problems. In contrast, Heterogeneous AutoML systems generate learning solutions by mixing techniques from different domains [5]. However, they do not provide natural interfaces for novice users

Doctoral Symposium on Natural Language Processing from the Proyecto ILENIA, 28 September 2023, Jaén, Spain.
elev1@alu.ua.es (E. L. Estevanell-Valladares)
0000-0002-1168-1767 (E. L. Estevanell-Valladares)

in programming or AutoML, and their execution requires preparation of the environment, the definition of the problem in appropriate terms, and the provision of data in specific formats [2, 3, 4, 6, 7, 8, 9].

Using natural language as a user interface can significantly enhance the accessibility and user-friendliness of AutoML. Large Language Models (LLMs) are trendy for their ability to process raw text and effectively identify patterns and connections in data [10, 11]. However, these models can sometimes generate incorrect responses or need help with inference tasks [12]. Recent studies suggest that incorporating external knowledge significantly improves the performance of LLMs [13].

Researchers are exploring combining these techniques to address the limitations of both AutoML and LLMs. Shen et al. [14] employed an LLM to process queries and generate learning task planning using pre-trained models. This approach has successfully integrated image, audio, and text prediction, classification, and processing capabilities into a single system. However, it does not focus on optimizing model selection or hyperparameter optimization. It does not produce tuned learning models in standalone programs or allow for the export of inference power to arbitrary environments.

The research project consists of producing a Heterogeneous AutoML system that integrates natural language processing as its primary interface. The ultimate goal is to design a tool to generate optimal machine learning models that are flexible and adaptable to different contexts and heterogeneous situations. This leads to our Main Research Question: "**In what way can we integrate Natural Language into a Heterogeneous AutoML process?**".

2. Related Work

Most AutoML systems only use limited algorithms specific to a particular library or toolkit. This limitation hinders their ability to solve complex problems by exploring various algorithms from different areas. On the other hand, Heterogeneous AutoML systems combine techniques from multiple domains to create better learning solutions. However, they require a user-friendly interface for those new to programming or AutoML. This setup involves defining the problem correctly, providing data in specific formats, and setting up the environment.

Table 1 contrasts several existing AutoML systems with the system proposed in this research regarding their capabilities of dealing with heterogeneous scenarios. This evaluation focused on their capacity to handle diverse scenarios encompassing multiple algorithms. It is worth noting that this assessment was solely based on their ability to handle heterogeneous algorithms without consideration for their overall performance, capacity, or applicability.

AutoML systems vary in capabilities and limitations, depending on the specific learning libraries they are built on. Some, like Auto-Sklearn [3], Auto-Weka [4], and Auto-Keras[2], are restricted to using Scikit-learn [15], Weka [16], and Keras [17], respectively. Other systems, such as RECIPE [9] and Hyperopt [7], can incorporate algorithms from different libraries but require a concrete implementation. TPOT [6] and ML-Plan [8] provide a more flexible approach, combining technologies from multiple learning libraries to create concrete implementations of learning pipelines. AutoML systems are mostly focused on supervised learning, but some offer the potential to integrate unsupervised learning functionality, like Hyperopt.

AutoML Systems	Multiple libraries	Multiple ML problems	Probabilistic	Extensible	Automatic Discovery	Multiobjective	Distributed	Deployable Pipelines	Year
Hyperopt	≈		✓	✓		✓	✓	✓	2013
Auto-Weka 2.0			✓	✓			✓	✓	2017
RECIPE	≈					≈			2017
TPOT	✓					✓		✓	2018
ML-Plan	✓	✓		✓	✓	✓	✓		2018
Auto-Keras			✓	✓		✓		✓	2019
Auto-Sklearn 2.0			✓	✓		✓	✓	✓	2020
AutoGOAL	✓	✓	✓	✓	✓	✓	✓	✓	2023

Table 1
Comparison of several existing AutoML systems' capabilities to deal with heterogeneous machine learning problems. Entries marked with ≈ indicate that the system design enables the given capability, but we have no record of its implementation.

Systems like Auto-Sklearn and Auto-Keras benefit from a unified underlying API, while modularly designed systems such as ML-Plan allow the addition and modification of algorithms. Several systems allow balancing different objectives or metrics during optimization, which is relevant in multiple development and research scenarios. For example, Auto-Keras and ML-Plan use a weighted sum approach to combine multiple evaluation metrics into a single objective function. These systems allow you to specify both the metrics and the weights assigned to each metric, which allows you to control their relative importance in the optimization process.

Hyperopt, ML-Plan, Auto-WEKA, and Auto-Sklearn include mechanisms to distribute search processes and resources among multiple computers, optimizing search time and generating learning pipelines more quickly. To meet the goal of AutoML, system solutions should be easily usable in arbitrary environments and applicable as portable machine learning algorithms outside the AutoML context. Systems such as Hyperopt, TPOT, and Auto-Sklearn allow exporting the best pipeline found during the search process as a Python script, while Auto-WEKA can export to JAVA [18] code. Auto-Keras allows exporting the models in various formats, including TensorFlow [19], PyTorch [20], and Keras [17].

Using probabilistic models to describe the space of possible pipelines is another interesting feature of AutoML systems. AutoML systems based on Bayesian optimization build an internal representation of the space of possible algorithm pipelines, which can be interpreted as assigning a probability distribution to each particular pipeline. This feature describes the algorithm pipeline space and allows researchers to gather additional information by analyzing which regions have higher or lower probabilities of generating effective Pipeline components and allow researchers to gather additional information.

Previous research addressed the limitations of AutoML systems. Estevez-Velarde et al. [21]

introduced the concept of Heterogeneous AutoML, a more general formulation of the AutoML Problem. Additionally, they introduced AutoGOAL, a flexible and efficient system for heterogeneous AutoML implemented in Python. With AutoGOAL, users can describe a specific machine problem, input and output requirements, and a set of objectives. The system then automatically finds the best pipeline of algorithms from various libraries, including Scikit-learn [15], NLTK [22], Keras [17], and Gensim [23]. It is also customizable, allowing users to add and integrate new algorithms into the existing pipelines. AutoGOAL uses a Pareto Front approach to multiobjective optimization and an optimization process based on probabilistic grammatical evolution for context-free grammar [24].

3. Proposed Research

AutoGOAL has achieved state-of-the-art performance against other AutoML systems and has been able to solve machine-learning tasks outside of supervised learning. Additionally, it can build complex pipelines targeting difficult NLP tasks like Named Entity Recognition, being able to connect algorithms of different natures. This research project will use AutoGOAL as a baseline for its capabilities regarding Heterogeneous AutoML.

3.1. Heterogeneous AutoML

According to Estevez-Velarde et al. [21], the Heterogeneous AutoML problem's space of all possible pipelines can be represented as a G_A graph. This graph consists of nodes representing each known algorithm a_i, and edges exist between all pairs a_i, a_j such that their corresponding output and input types are compatible. Given a machine-learning task defined as a function that transforms an input type into an output type, we can build a specific search space graph G'_A that only models valid pipelines. To extract G'_A, we must introduce two additional nodes to G_A: Input and Output. These nodes are then connected to all algorithms capable of producing the desired output from the specific input. By identifying any path in G'_A that connects the Input and Output nodes, we can obtain a pipeline that addresses the machine-learning problem at hand.

A suitable computational implementation of this process requires solving the following problems:

1. Defining each algorithm and their respective input and output, such that it is computationally feasible to determine if two algorithms can be connected and construct the graph.
2. Designing an optimization strategy that can effectively search in the space of all pipelines, algorithms, and their hyperparameters, given restricted computational resources.

AutoGOAL utilizes a set of Semantic Type objects to implement this compatibility function. Each semantic data type is a Python class that belongs to a hierarchy in which object inheritance directly represents the relation for type compatibility (e.g., `Word` can also be interpreted as `Text`, a more general type). The data types have a semantic interpretation beyond their underlying computational structure. For example, a string in computational terms can either be a Document, a Sentence, or a Word. Each algorithm is implemented as a class with a `run(input: Tin) -> Tout` method that performs the corresponding processing, potentially wrapping an underlying implementation from a machine learning library. Each algorithm's input and output types are specified using Semantic Types and represented by the `Tin` and `Tout` annotations.

While this method for computing compatibility has advantages, it is rigid and difficult to maintain. Due to the closed nature of the type system, precise type definitions must be matched for any new algorithms identified and added to the AutoGOAL search space. Also, because adding new Semantic Types does not automatically update current algorithm annotations, users need to check on every existing algorithm to identify which should be annotated accordingly. Moreover, this mechanism assumes a tree-like structure of type compatibility when there might be more complex relationships (e.g., a `Stem` might also be considered a `Word`, albeit these two types do not inherit from each other). This leads to an interesting question: "**Can we model algorithm compatibility more openly?**".

Recent proposals suggest using natural language to store information describing algorithms. Shen et al. [14] uses Jsons, mainly text-based, to store information about pre-trained models. They parse natural language prompts into multiple tasks that are matched with suitable algorithms using an LLM. However, this tool does not address the AutoML problem, as it does not optimize model selection or hyperparameter configurations for algorithms. In contrast, Zhang et al. [25] aims to develop an AutoML system called AutoML-GPT, which uses LLMs to train models on datasets with user inputs and descriptions automatically. The LLMs serve as an automatic training system to establish connections with models and process inputs.

3.2. Research Questions

In this research, we propose the integration of description cards based on text data for algorithms in AutoGOAL. By leveraging the power of LLMs, we can match algorithms based on their description. This method adds a vital factor of generalization that might solve the previous limitations of AutoGOAL. Moreover, by substituting the current Semantic Type system with natural language, we open the tool's interface to accept user text prompts.

To achieve our main objective, we must address various questions within the proposal:

1. **Which LLM should we use?**
2. **What language should we support in the interface?**
3. **What machine learning tasks should we target?**

To answer the first question, we must conduct experiments to determine which LLM will be most suitable for our project. One idea is to use two different LLMs that are each fine-tuned for specific purposes. For example, one LLM can help determine the compatibility between algorithms used during the optimization process. At the same time, the other can identify problem definitions (input and output types and objective functions) out of natural language for better user interaction with the system.

For the second question, we aim to develop an interface not tailored to a specific language to make it fully inclusive. However, the performance of LLMs can vary significantly in different languages due to differences in available training data. Therefore, we will compare the performance of multilingual models against specific-language models in our objective tasks before deciding which approach to follow.

Finally, our main objective is to extend existing tools, specifically AutoGOAL, to saturate the definition of Heterogeneous AutoML, thus achieving more flexibility and integrating more tasks seamlessly. We can achieve this by using the compatibility function to discover algorithms that were once bound to a specific machine-learning problem but can be part of the solution of another one. The diversity and amount of algorithms determine the limit of tasks we can solve in the system.

The proposed system has valuable scientific, economic, and social implications. It can enhance our understanding of artificial intelligence and apply it to robotics and process automation. Economically, it can speed up the development of applications and decrease the cost and time required for building machine learning solutions.

From a social standpoint, an AutoML system based on natural language can improve the accessibility and ease of use of machine learning, especially in critical areas such as healthcare and education. Furthermore, automating the building of learning models can lessen the need for human intervention in repetitive and monotonous tasks, thus reducing the carbon footprint associated with computer system operations.

4. Proposed Experimentation

To evaluate the potential of our proposed system, we plan to develop a benchmark that incorporates challenging tasks from multiple domains such as vision, NLP, audio, and others. We will perform ablation studies to comprehend the significance of different LLMs and optimization strategies in the overall performance of our system. In addition, to make the evaluation more comprehensive, we will compare our new system with its previous version and other state-of-the-art AutoML systems. By providing more flexibility, we aim to test the capabilities of our system against human adversaries in various challenges. Furthermore, we will explore the possibility of integrating human feedback into the learning process, which can provide valuable insights and lead to further improvement.

5. Conclusions and Future work

The purpose of this publication is to present the research framework for a thesis that aims to investigate the intersection between AutoML and Large Language Models (LLMs). Our objective is to improve AutoML systems, making them more accessible, user-friendly, and versatile. In order to achieve this goal, we will begin by examining the current state of the art in this field. Subsequently, we will develop corresponding description cards for each algorithm available in AutoGOAL and also include new algorithms, such as pre-trained models from Hugging Face along with their respective cards. The next step will be integrating an LLM to model the compatibility function between algorithms, thereby enabling a natural language interface for user interaction. Ultimately, our aim is to pave the way for more inclusive and efficient machine learning applications in various domains.

References

[1] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning*. Springer, 2019.

[2] Haifeng Jin, Qingquan Song, and Xia Hu. “Auto-Keras: An Efficient Neural Architecture Search System”. In: ACM, 2019, pp. 1946–1956. DOI: 10.1145/3292500.3330648.

[3] Matthias Feurer et al. “Auto-Sklearn 2.0: The Next Generation”. In: *arXiv: Learning* (2020).

[4] Chris Thornton et al. “Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms”. In: ACM, 2013, pp. 847–855. DOI: 10.1145/2487575.2487629.

[5] Suilan Estevez-Velarde et al. “Automl strategy based on grammatical evolution: A case study about knowledge discovery from text”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 4356–4365.

[6] Randal S. Olson and Jason H. Moore. “TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning”. In: vol. 2019. Springer, Cham, 2018, pp. 66–74. DOI: 10.1007/978-3-030-05318-5_8.

[7] Brent Komer, James Bergstra, and Chris Eliasmith. “Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn”. In: (2013), pp. 32–37. DOI: 10.25080/MAJORA-14BD3278-006.

[8] Felix Mohr, Marcel Dominik Wever, and Eyke Hüllermeier. “ML-Plan: Automated machine learning via hierarchical planning”. In: *Machine Learning* 107.8 (2018), pp. 1495–1515. DOI: 10.1007/S10994-018-5735-Z.

[9] Alex Guimarães Cardoso de Sá et al. “RECIPE: A Grammar-Based Framework for Automatically Evolving Classification Pipelines”. In: Springer, Cham, 2017, pp. 246–261. DOI: 10.1007/978-3-319-55696-3_16.

[10] Bonan Min et al. *Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey*. 2021. arXiv: 2111.01243 `[cs.CL]`.

[11] Frank F. Xu et al. “A Systematic Evaluation of Large Language Models of Code”. In: (Feb. 2022). DOI: 10.48550/ARXIV.2202.13169. arXiv: 2202.13169 `[cs.PL]`.

[12] Abubakar Abid, Maheen Farooqi, and James Zou. *Persistent Anti-Muslim Bias in Large Language Models*. 2021. DOI: 10.1145/3461702.3462624.

[13] Baolin Peng et al. “Check your facts and try again: Improving large language models with external knowledge and automated feedback”. In: *arXiv preprint arXiv:2302.12813* (2023).

[14] Yongliang Shen et al. “HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace”. In: *arXiv preprint arXiv:2303.17580* (2023).

[15] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830.

[16] Geoffrey Holmes, A. Donkin, and Ian H. Witten. “WEKA: a machine learning workbench”. In: IEEE, 1994, pp. 357–361. DOI: 10.1109/ANZIIS.1994.396988.

[17] François Chollet. *Keras: The Python Deep Learning library*. 2018.

[18] James Gosling et al. *The Java language specification*. Addison-Wesley Professional, 2000.

[19] Martín Abadi et al. "TensorFlow: a system for large-scale machine learning". In: USENIX Association, 2016, pp. 265–283. DOI: 10.5555/3026877.3026899.

[20] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: vol. 32. hgpu.org, 2018, pp. 8026–8037.

[21] Suilan Estevez-Velarde et al. "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.

[22] Edward Loper and Steven Bird. "NLTK: the natural language toolkit". In: *arXiv preprint cs/0205028* (2002).

[23] Petr Sojka. "Gensim—statistical semantics in python". In: *Retrieved from genism. org* (2011).

[24] Hyun-Tae Kim and Chang Wook 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.

[25] Shujian Zhang et al. "AutoML-GPT: Automatic Machine Learning with GPT". In: *arXiv preprint arXiv:2305.02499* (2023).