Demo Application for the AutoGOAL Framework

Suilan Estevez-Velarde[1], **Alejandro Piad-Morffis**[1], **Yoan Gutiérrez**[2,3], **Andrés Montoyo**[2,3], **Rafael Muñoz**[2,3], and **Yudivián Almeida-Cruz**[1]

[1]School of Math and Computer Science, University of Havana, Cuba
{sestevez,apiad,yudy}@matcom.uh.cu
[2]University Institute for Computing Research (IUII), University of Alicante, Spain
[3]Department of Languages and Computing Systems, University of Alicante, Spain
{ygutierrez,montoyo,rafael}@dlsi.ua.es

Abstract

This paper introduces a web demo that showcases the main characteristics of the AutoGOAL framework. AutoGOAL is a framework in Python for automatically finding the best way to solve a given task. It has been designed mainly for automatic machine learning (AutoML) but it can be used in any scenario where several possible strategies are available to solve a given computational task. In contrast with alternative frameworks, AutoGOAL can be applied seamlessly to Natural Language Processing as well as structured classification problems. This paper presents an overview of the framework's design and experimental evaluation in several machine learning problems, including two recent NLP challenges. The accompanying software demo is available online[1] and full source code is provided under the MIT open-source license[2].

1 Introduction

The field of machine learning has applications across a wide range of computational problems in different domains. However, given the vast quantity of resources and technologies available, often one of the most difficult challenges is to select the best combination of them when a specific problem is faced. Researchers often spend a significant amount of time and computational resources exploring multiple approaches in search of optimal configurations. The field of automatic machine learning (AutoML) has risen to prominence as a principled alternative for finding optimal or close to optimal solutions to complex machine learning problems (Hutter et al., 2018). Several software libraries have been created, which leverage existing machine learning technologies and provide AutoML features built on them. Most existing AutoML tools focus on a specific family of algorithms (such as neural networks) or a specific problem setting (such as supervised learning from tabular data). Hence, despite the recent success of AutoML, several challenges still remain, specially in complex domains such as natural language processing, where tools and technologies from different sources must be combined.

This work presents AutoGOAL, a software library for AutoML that can seamlessly combine technologies and resources from different frameworks. To unify disparate APIs into a single interface, AutoGOAL proposes a novel graph-based representation for machine learning pipelines. Furthermore, a search strategy based on probabilistic grammatical evolution is used to discover optimal machine learning pipelines (Estevez-Velarde et al., 2019) whose components can be from different back-end libraries. A showcase web application is provided to illustrate the use of the framework.

The key features of AutoGOAL are:

Ease of use: AutoGOAL provides high-level classes that non-experts can use as black-box AutoML solutions, compatible with several data types including text, images and structured (tabular) data.

Multiple domains: AutoGOAL comes prepackaged with 133 plus adapters of existing algorithms, from 7 different back-end libraries, for multiple problems including text preprocessing, feature extraction,

[1]https://autogoal.github.io/demo
[2]https://autogoal.github.io

Proceedings of the 27th International Conference on Computational Linguistics, pages 18–22
Barcelona, Spain (Online), December 12, 2020.

dataset augmentation, dimensionality reduction, as well as supervised and unsupervised learning techniques.

Extensibility: AutoGOAL proposes a simple programming interface that developers can implement to create an automatically discoverable adapter for an existing technology from any machine learning framework.

This paper focuses on the engineering design of AutoGOAL by introducing a demo application using the library to solve several machine learning problems, organised as follow: Section 2 describes AutoGOAL from the perspective of a user of the library. Section 3 describes the library's design. Section 4 presents experimental results of the application of AutoGOAL to several different machine learning problems. Finally, Section 5 presents the conclusions and recommendations for future work.

2 Library Usage

AutoGOAL can be used as a software library in the Python programming language. This library is oriented towards two machine learning user profiles: non-experts and experts, and provides a High-Level and Low-Level API respectively.

High-Level API (non-experts): This API allows AutoGOAL to be used as a black-box classification or regression algorithm with an interface similar to the *scikit-learn* library (Pedregosa et al., 2011). Behind this interface, a complete process including preprocessing, feature selection, dimensionality reduction, and learning is performed. The user must define a dataset for training and evaluation, a metric to optimise (which defaults to *accuracy*) and the type of input and output data. In many cases AutoGOAL can automatically infer the input and output type from the dataset. Input and output types can vary from tabular data to complex types such as images, natural language text with different semantic structures, and combinations thereof. Figure 1 shows an illustrative example source code, specifically in the context of a text classification problem.

```
from autogoal.ml import AutoML
from autogoal.datasets import haha
# import lines for semantic datatypes

automl = AutoML(
    # problem-specific input and output (semantic datatypes)
    input=List(Sentences()),
    output=CategoricalVector()
    # additional parameters for timeout, memory, iterations, etc.
)

X, y = haha.load() # load problem-specific dataset
automl.fit(X, y)   # run optimisation
```

Figure 1: Example source code for running AutoGOAL on a specific dataset, in this case an NLP problem.

Low-Level API (experts): This API is designed for users with more experience that need control over the AutoML process. For this type of user, AutoGOAL provides a simple language for defining a grammar that describes the solution space. This is done using an object oriented approach were the user defines a Python class for each component of the solution (e.g., each algorithm) and annotates the parameters of these classes with attributes that describe the space of possible values, which can be primitive value types (i.e., numeric, string, etc.) and instances of other classes, recursively. Based on the annotations, AutoGOAL can automatically construct all possible ways in which the user classes can be instantiated. An example code of this process is available in the online documentation.[3]

[3] https://autogoal.github.io/examples/sklearn_simple_grammar/

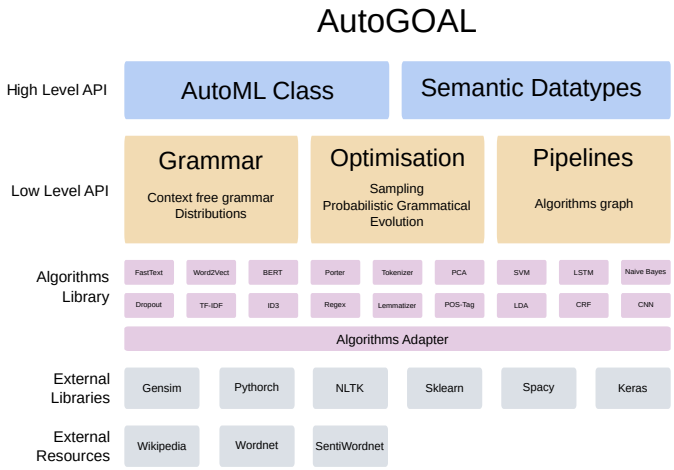

Figure 2: Overall architecture of the AutoGOAL framework.

3 Implementation Details

This section presents the overall architecture of the AutoGOAL library. For reference purposes, Figure 2 illustrates the most relevant components of AutoGOAL, ranging from the High-Level API down to the actual implementation of algorithm adapters and the interfaces to external resources and back-end libraries. The core of the AutoGOAL library is the Low-Level API, composed of the following elements (see Figure 2): a probabilistic context-free grammar module (*Grammar*); a sampling and optimisation module (*Optimisation*); and, a pipeline discovery module (*Pipelines*).

The *Grammar* module provides a set of type annotations that are used for defining the hyperparameter space of an arbitrary technique or algorithm. Each technique is represented as a Python class, and the corresponding hyperparameters are represented as annotated arguments of the `__init__` method, either primitive values (e.g., numeric, string, etc.) or instances of other classes, recursively annotated. Given a collection of annotated classes, this module automatically infers a context-free grammar that describes the space of all possible instances of those classes.

The *Optimisation* module provides sampling strategies that traverse a context-free grammar and recursively construct one specific instance following the annotations. Two optimisation strategies are implemented: random search and probabilistic grammatical evolution (O'Neill and Ryan, 2001). The latter performs a sampling/update cycle that selects the best performing instances according to some predefined metric (e.g., accuracy on a development set) and iteratively updates the internal probabilistic model of the sampler (Estévez-Velarde et al., 2020).

The *Pipelines* module provides an abstraction for algorithms to communicate with each other via an Facade pattern, i.e., the implementation of a method `run` with type-annotated input and output. Classes implementing this pattern are automatically connected in a graph of algorithms where each path represents a possible pipeline for solving a specific problem (defined by the input and output datatypes).

The High-Level API that provides the `AutoML` class (see Listing 1) and the *Semantic Datatypes* is built on top of this architecture, knitting together all the components of AutoGOAL. AutoGOAL also provides an Algorithms Library of pre-made adapters for existing machine learning technologies from External Libraries and Resources. A total of 133 algorithms from 7 different back-end libraries[4] are provided, several of which are semi-automatically created by code introspection (e.g., algorithms from *scikit-learn* and *nltk* which conform to a consistent API and the hyperparameters are documented in a consistent format), and the rest are manually added by the library developers. This library is undergoing continuous expansion. Additionally, a Docker image is provided with all optional dependencies and

[4]Including *scikit-learn*, *nltk*, *gensim*, *spacy*, *keras*, *pytorch*, among others.

back-end libraries already installed[5].

4 Evaluation

AutoGOAL has been evaluated in different domains and compared to other AutoML tools, including Auto-Weka (Thornton et al., 2013), TPOT (Olson and Moore, 2016), Auto-Sklearn (Feurer et al., 2015), and ML-Plan (Mohr et al., 2018), see Table 1. AutoGOAL is compared with other AutoML approaches in classic datasets (Dua and Graff, 2017) but can applied to more complex domains such as text classification in HAHA (Chiruzzo et al., 2019) and entity recognition in MEDDOCAN (Lara-Clares and Garcia-Serrano, 2019).

In terms of performance, AutoGOAL achieves comparative results with other AutoML tools in classic datasets, with similar computational cost (1 hour per run). However, AutoGOAL's main strength lies in its ability to combine different tools for solving complex problems beyond structured supervised learning, such as natural language processing, using virtually the same code, by specifying the input and output datatypes. In these domains AutoGOAL performs comparable to state-of-the-art solutions hand-crafted by human experts, while requiring considerably less expertise and effort (48 hours of execution).

Dataset	**Cars**	**Credit G.**	**Abalone**	**Shuttle**	**Yeast**	**Dorothea**	**Gisette**	**HAHA**	**MEDD.**
ML-Plan (Weka)	1.27	25.54	73.72	0.01	39.37	6.49	2.92	-	-
Auto-WEKA	0.66	26.50	73.46	0.12	39.72	-	3.90	-	-
ML-Plan (Sklearn)	0.34	24.56	73.77	0.02	39.52	8.69	2.76	-	-
Auto-Sklearn-v	1.38	25.95	82.92	0.02	40.51	6.32	2.56	-	-
Auto-Sklearn-we	1.26	25.39	80.59	0.02	38.99	6.02	2.24	-	-
TPOT	0.37	23.91	73.14	0.02	38.47	-	-	-	-
AutoGOAL	0.60	27.01	74.33	0.11	39.94	5.97	2.25	21.1	3.99

Table 1: Comparison of AutoGOAL and other AutoML systems for 9 classic machine learning datasets in terms of accuracy, except for MEDDOCAN, in which F_1 is used. Values for other systems were obtained from *ML-Plan* (Mohr et al., 2018).

5 Conclusion

In this paper we presented AutoGOAL, a new tool for AutoML that allows resources from different machine learning libraries to be combined and applied to different domains with little effort. AutoGOAL greatly simplifies the application of machine learning for non-expert users while providing powerful low-level components for experts to effectively optimise complex machine learning pipelines. The framework has been designed with extensibility as a priority, enabling the addition of new algorithms from any conceivable machine learning library by conforming to a simple interface. To demonstrate its usefulness, AutoGOAL is applied to different domains —including classic numeric datasets, text classification, and entity recognition— achieving competitive results with the state of the art. The software is provided freely for the research community along with a vast library of algorithms already implemented.

Acknowledgements

Funding: This research has been supported by a Carolina Foundation grant in agreement with University of Alicante and University of Havana. Moreover, it has also been partially funded by both aforementioned universities, the Generalitat Valenciana (*Conselleria d'Educació, Investigació, Cultura i Esport*) and the Spanish Government through the projects LIVING-LANG (`RTI2018-094653-B-C22`) and SIIA (`PROMETEO/2018/089`, `PROMETEU/2018/089`).

[5] `https://hub.docker.com/repository/docker/autogoal/autogoal`

References

Luis Chiruzzo, S Castro, Mathias Etcheverry, Diego Garat, Juan José Prada, and Aiala Rosá. 2019. Overview of haha at iberlef 2019: Humor analysis based on human annotation. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2019). CEUR Workshop Proceedings, CEUR-WS, Bilbao, Spain (9 2019)*.

Dheeru Dua and Casey Graff. 2017. UCI machine learning repository.

Suilan Estevez-Velarde, Yoan Gutiérrez, Andrés Montoyo, and Yudivián Almeida-Cruz. 2019. 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*, pages 4356–4365, Florence, Italy, July. Association for Computational Linguistics.

Suilan Estévez-Velarde, Yoan Gutiérrez, Yudivián Almeida-Cruz, and Andrés Montoyo. 2020. General-purpose hierarchical optimisation of machine learning pipelines with grammatical evolution. *Information Sciences*, 543:58–71.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pages 2962–2970.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. 2018. *Automated Machine Learning: Methods, Systems, Challenges*. Springer. In press, available at http://automl.org/book.

Alicia Lara-Clares and Ana Garcia-Serrano. 2019. Key phrases annotation in medical documents: Meddocan 2019 anonymization task.

Felix Mohr, Marcel Wever, and Eyke Hüllermeier. 2018. ML-Plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107(8):1495–1515, sep.

Randal S Olson and Jason H Moore. 2016. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on Automatic Machine Learning*, pages 66–74.

Michael O'Neill and Conor Ryan. 2001. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358.

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.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855. ACM.