Híper Heurísticas para la Selección de Métodos de Aprendizaje Profundo en la Clasificación de Textos Automatizada

Hyper-Heuristics for Selecting Deep Learning Methods in Automated Text Classification

Jonathan Estrella-Ramirez[1] and Juan Carlos Gomez[1 2]
[1]Universidad de Guanajuato, División de Ingenierías Campus Irapuato-Salamanca Gto. 36787, México
{jdj.estrellaramirez, jc.gomez}@ugto.mx

Abstract

In this paper, an evolutionary model, in the scope of automated machine learning, that learns selection hyper-heuristics for text classification is presented. A hyper-heuristic is a set of if-then rules that evaluate a set of meta-features, summarizing the data distribution of a dataset, to select the most adequate deep learning method for such a dataset. It is expected that datasets with similar distributions can use the same classification model, generalizing the selection process. The model initially creates a population of hyper-heuristics at random and then evolves them using specific mutation and crossover operators. During the evolution, each hyper-heuristic is evaluated for its classification performance with a training group of datasets. At the end of the evolution, the best hyper-heuristic is chosen and evaluated for classification with an independent group of datasets. The results indicate that the best hyper-heuristic generalizes well the selection process, by choosing adequate classification methods for the datasets; and reaches a better performance than two state-of-the-art automated machine learning systems.

Key words: Automated machine learning, Evolutionary algorithms, Hyper-heuristics, Text classification

Introduction

Text classification is a popular task in machine learning (ML), with problems such as fake news detection, news classification, and spam email filtering, among others [Li et al., 2020]. Over the years, to study these problems, several datasets have been formed, and various classification methods have been developed, such as instance-based, probabilistic, rules-based, support vector machines, and more complex ones based on deep learning (DL) such as gated recurrent units, long short-term memory, and transformer-based neural networks [Gasparetto et al., 2022].

Unfortunately, there is not a single classification method that performs equally well for all possible datasets in text classification. Usually, when dealing with a specific (set of) dataset(s), the researchers train and test sets of classification methods with different configurations to find the best among them. Depending on the number and types of methods, this process can be very expensive in terms of time and computational resources. Several aspects affect the performance of a classification method for a particular dataset, such as the number of categories in the dataset, the total number of documents, the number of words per document, and the imbalance in categories.

Automated machine learning (AutoML) is a research field that has emerged recently with the idea of automating the process of method selection and tuning for classification problems [Chauhan et al., 2020]. Some of the most popular systems from this approach were presented in the two ChaLearn AutoML challenges (held between 2015 and 2018) [Guyon et al., 2019], where the systems had to solve regression or classification problems without human intervention, handling image, text, and video data. The winning systems of each challenge were Auto-Sklearn [Feurer et al., 2015] and PoSH Auto-Sklearn [Feurer et al., 2018], both belonging to the *AAD Freiburg* team.

In recent years, various AutoML systems for text classification have been presented. In [Gomez et al., 2017], the authors presented a learning of meta-rules for selecting classification methods for text datasets by evaluating a set of 11 meta-features. In [Madrid and Escalante, 2019], the authors expanded the set of meta-features used in [Gomez et al., 2017] to 72 to identify the classification problem of a dataset. As an extension

of [Madrid and Escalante, 2019], in [Madrid, 2019] the authors presented an AutoML system, named AutoText, which transforms a dataset to the most appropriate textual representation. This new representation is entered into the Auto-Sklearn system [Feurer et al., 2015], which searches for the best classification method for this dataset.

More recently, other more complex AutoML proposals for text classification have been presented. In [Jin et al., 2019] the authors presented the AutoKeras, an efficient neural architecture system that uses Bayesian-guided network morphism to find the best classification method, with its respective configuration, of a pool of base neural networks methods. Also, the internal embedding of AutoKeras allows it to handle raw text and high-level configurations (e.g., vectorization methods, word embeddings, etc.). In [Erickson et al., 2020] the authors presented AutoGluon, a system that builds a classifier based on the architecture of a neural network, where each layer consists of stacks of different classification methods (neural networks, random forests, and K-nearest neighbors, among others). Both systems present a high complexity in the selection and building of classification methods, avoiding users from fully understanding and exploiting the capabilities of the systems.

Most of these works optimize the pre-processing and/or the method selection but only for a single dataset. A generalization of the process, i.e., a model that allows selecting suitable a classification method for multiple datasets, is still missing.

This paper introduces an evolutionary model in the scope of AutoML that intends to generalize the selection of classification methods for text datasets. The model learns hyper-heuristics (hhs), as sets of if-then rules. Each rule in a hh evaluates the data distribution of a dataset and chooses the most appropriate classification method for it. The data distribution is defined as a set of 16 meta-features that represent statistical aspects of the data, such as the number of documents and categories, the percentage of difficult words, etc. The classification method is selected from a pool of 12 available DL methods based on transformers. These methods are expensive to train, and test compared to other classical ML methods, but the model allows for easily including any kind of classification method in the pool.

The results reveal that the best-learned hh can select adequate methods (with a near-optimum performance) for a test group of text datasets. These results are also compared with those of two state-of-the-art AutoML systems, demonstrating that the hh is competitive.

The contributions of this work are the following. 1) The development and evaluation of an evolutionary model to generalize the method selection for classifying text datasets. 2) An understandable representation of hhs using if-then rules. 3) The validation of using meta-features based on data distribution to decide the most adequate classification method for datasets. 4) The comparison of the best learned hh with two state-of-the-art AutoML systems.

The rest of the paper is organized as follows. Section 2 describes the material and methods used for building, learning, and evaluating the hhs in the evolutionary model. Section 3 presents and discusses the results obtained with the evolutionary model, and the conclusions and possible directions for future research.

Material and Methods

Datasets and meta-features

Table 1 shows the 34 text datasets that were used for experiments. These datasets encompass a diversity of text classification problems and variability regarding the number of documents and categories. Two genetic groups (training and test) were created with 50% of the data (randomly taken) from each dataset, in a stratified way. Then, each dataset of both genetic groups was split into 80% for training and 20% for testing, following the same process as the first split.

Table 1. *Datasets for classification: News: News. Sent.: Sentiment. Email: Email. Bull.: Cyberbullying. Hier.: Hierarchical. Dis.: Disaster. Pol.: Political. Age: Age. F.J.: Fake job. Res.: Research articles. Mod.: Moderation.*

Dataset	Task	#Cat	#Docs	Dataset	Task	#Cat	#Docs	Dataset	Task	#Cat	#Docs
20ng	News	20	7528	imdbs	Sent.	2	1000	Rotten	Sent.	5	156060
AGNews	News	4	127600	IMDBR	Sent.	2	50000	sen_pol	Sent.	2	10662
CNNAC	News	9	37949	LvsC	Pol.	2	12854	StOvQR	Mod.	3	60000
CNNAS	News	49	37949	movies	Sent.	2	2000	SuiDect	Sent.	2	232074
CorTws	Sent.	5	44955	NewsCat	News	41	200853	TMACS	Res.	2	20972
csdmc	Email	2	4327	NYTAND	News	43	9335	TMAMt	Res.	2	20972
CybTws	Bull.	6	47692	NYTATM	News	13	9335	TMASt	Res.	2	20972
DisTws	Dis.	2	11370	oh	News	23	7399	TripAd	Sent.	5	20491
F&RNS	News	18	44919	r8	News	8	7673	wipo_l1	Hier.	114	75249
gopds	Sent.	3	13871	r52	News	51	9099	wipo_l2	Hier.	922	75249
HRCB	Age	10	3269	RFJob	F.J.	2	17880	yelp	Sent.	2	1000
HRCS	Age	48	430	-	-	-	-	-	-	-	-

Source: Author's own elaboration.

For each dataset, the set of 16 meta-features shown in Table 2 was computed. Each meta-feature represents a different statistical aspect of the data distribution at a different level (document, category, and dataset). Regarding the PCA meta-features, these were computed from a term frequency-inverse document frequency (tf-idf) representation. Words with more than two syllables were considered as difficult words.

Table 2. *Set of meta-features used to represent the data distribution of a dataset.*

Abbrev.	Description	Abbrev.	Description
nDocs	Number of documents	nTops	Number of Categories
dptAvg	Mean of documents per category	dptMed	Median of documents per category
dptStd	Standard deviation of documents per category	dptStdAvg	Ratio between dptStd and dptAvg
dptEnt	Shannon entropy of documents per category	wpdAvg	Mean of words per document
wpdMed	Median of words per document	wpdStd	Standard deviation of words per document
wpdStdAvg	Ratio between wdpStd and wpdAvg	wpdEnt	Shannon entropy of words per document
pca10	Variance in the first 10 components of PCA	pca20	Variance in the first 20 components of PCA
pca30	Variance in the first 30 components of PCA	cmxWds	% of difficult words in the dataset

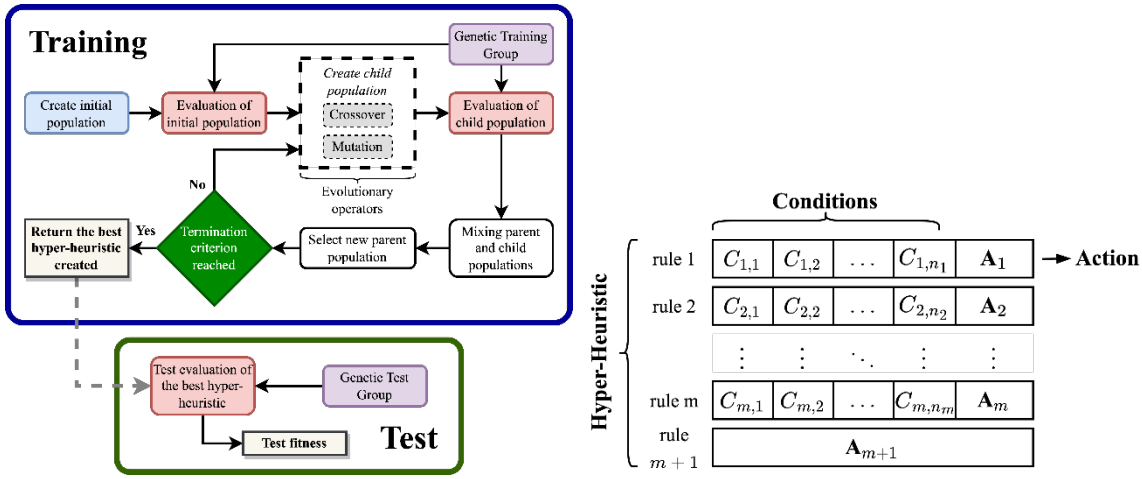
Source: Author's own elaboration.

General model and representation of hyper-heuristics

Figure 1.a shows the general process of the proposed evolutionary model, which is divided into two phases: training and test.

At the beginning of the training phase, the evolutionary model creates a population of hhs at random. A hh is a set of m (between 2 and 16) if-then rules, that can be represented by blocks as shown in Fig. 1.b. A rule is made up of a random number of conditions (between 1 and 4). A condition evaluates one meta-feature and can be defined as a 3-tuple (feature, comparison operator, reference value). The comparison operator can be '<' or '>', and the reference value is limited by the maximum value found in the genetic training group for that meta-feature. A rule $m + 1$ (else rule) is added to the end of the hh in case no if-then rule is met. If all the conditions of a rule are met, an action is selected from a pool of classification methods. The pool contains 12 instances of two base methods: ALBERT [Lan et al., 2019] and BERT [Devlin et al., 2018], both of which are deep learning transformer methods, that are pre-trained and perform fine-tuning on the dataset to be classified.

For ALBERT, the pre-trained method *albert-base-v2*[1] was used. In BERT's case, *bert-base-uncased*[2] was selected. For each method, variants for learning rates of 1e−3, 1e−4 and 1e−5 were used; and epochs of 4 and 6.



(a) *(b)*

Figure 1. (a) Schema of the evolutionary model and (b) Block representation of a hh. Source: Author's own elaboration.

Evolutionary operators

Crossover operator During evolution, the model applies three versions of crossover at different levels: block, rule, and condition, to create a child population of hhs. Each operator is used with the same probability. The child population has the same number of hhs as the parent population.

Block level. The operator applies a single-point technique by selecting two hhs at random and choosing a split point for them. The concatenation of the first subset of rules of hh 1 and the second subset of rules of hh 2 defines the first crossed hh. The second crossed hh is created with the remaining subsets.

Rule level. The operator selects two hhs at random and then swaps one rule (chosen at random) between them.

Condition level. The operator randomly selects two rules from two random hhs. Rule 1 of hh 1 swaps a random number of conditions (including the action) with rule 1 of hh 2; and the same applies to the other pair of rules.

Mutation operator This operator can explore various regions of the search space, so the model implements four levels of mutation. Each mutation operator has the same probability to be applied to some chosen hhs from the population created by crossover.

[1] Available at: https://bit.ly/3nhjnJp

[2] Available at: https://bit.ly/3K6avPK

Block level. The operator removes or adds a new rule to the chosen hh at a random position. When a rule must be added, such a rule is created randomly as in the initial population. In both cases, only if-then rules are considered.

Rule level. The operator replaces a random number (maximum half the length of the hh) of rules from the chosen hh with new, randomly created, rules. In this case, the else rule can be selected.

Condition level. The operator first selects a random number of rules from the chosen hh (the else rule cannot be selected). For each selected rule, the operator replaces a random number of conditions (including the action) with new ones that are created at random.

Operator level. This operator works similarly to the previous one, except that select conditions. For each selected condition, the comparison operator and/or the reference value associated with the rule will be mutated.

Evaluation and selection

In each evolutionary generation, the new hhs are evaluated to determine their fitness to classify the datasets of the genetic training group. For a hh, the evaluation process is as follows:

1. Obtain the meta-features of the dataset $i = 1, 2, 3, \ldots, 34$.
2. Evaluate the rules of the hh sequentially. If all the conditions of a rule are met, the model trains the associated classification method (the action) using the training part of the dataset i and tested with the respective test part. If no rule is activated, the classification method of the else rule is applied. Steps 1 and 2 are repeated until $i = 34$. The training and test parts of each dataset are transformed using the tf-idf method, and the performance of the classification method is determined by using the macro F1 metric.
3. Calculate the fitness of the hh. The macro F1 values obtained for all datasets are added and averaged.

This three-step sequence is applied for each new hh. When the evolutionary process begins, the first parent population is randomly created and evaluated as mentioned above. Then, from this parent population, the evolutionary operators create a child population that is also evaluated. The parent and child populations are mixed and sorted according to their fitness. Finally, the new parent population is created by selecting the hhs with the highest fitnesses. This process is repeated for a given number of generations.

Once the process has finished, the best hh from the last population is selected as the final solution. The rules of this hh are expected to select the most suitable classification method for any group of unseen datasets.

During the test phase, the final hh is evaluated with the datasets in the genetic test group (unseen datasets) to test the generalization abilities, applying the same three-step sequence mentioned above. If the hh generalizes well, the fitness value obtained by the hh should be close to the optimal classification value for the datasets in this group.

All the components of the evolutionary model were implemented in Python, using the modules ktrain, math, nltk, numpy, scikit-learn, spacy, and random.

Results and conclusions

The codes were executed using the Spyder environment[3]. For running all the experiments in this section, a server with 2 Intel Xeon Gold processors @2.6 GHz, 256 GB of RAM, a GPU Tesla V100 with 32 GB of RAM,

[3] Available at: https://www.spyder-ide.org/

and Windows Server 2016 was used. Both the classification methods of the evolutionary model and the baseline AutoML systems were trained using the GPU.

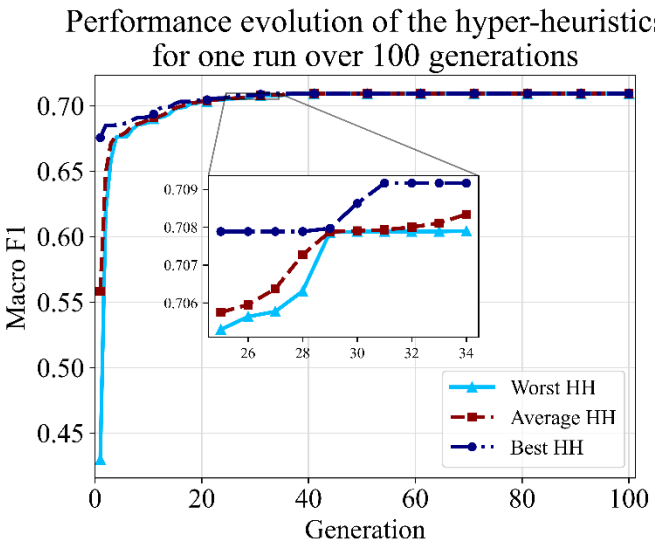
The general parameters of the evolutionary model to conduct the experiments described in this section were as follows:

- Population size: 50.
- Number of generations: 100.
- Number of rules: maximum 16 and minimum 2.
- Number of conditions: maximum 4 and minimum 1.
- Mutation probability: 20% (5% for each operator).

The mutation probability used in this work (contrary to the small values used in the literature) has the purpose that the different mutation operators are applied, increasing the diversity in the populations.

The experiments were split into two parts. In the first part, the model went to the training phase to learn the best hh. This hh went to the test part to be evaluated with the datasets from the genetic test group and compared with the optimal expected classification results. In the second part, the best hh was compared with two state-of-the-art AutoML systems.

After the training phase of the evolutionary model, a hh with the fitness of **0.7086** was obtained (fitness over the genetic training group). Figure 2.a shows a plot with the fitness behavior of the hhs during the evolutionary process of 100 generations. The behaviors of the best hh, the worst hh, and the average are shown. It is observed that after generation 40, the model reached convergence. The rules of the final hh are shown in Figure 2.b. It is observed that nine out of 16 meta-features are considered. In this hh, the most relevant meta-feature is dptEnt, which appears in four rules. This feature measures the amount of information inside the categories.



Hyper-heuristic

Rule 1: **if** *dptStdAvg* > 1.7190 **then**
BERT lr=1e-4, epochs=6

Rule 2: **if** *pca30* > 0.2160 **then**
BERT lr=1e-4, epochs=4

Rule 3: **if** *pca10* < 0.8885 **and** *nDocs* > 90576.564 **and** *dptAvg* > 46023.854 **and** *wpdStd* < 149.337 **then**
ALBERT lr=1e-5, epochs=4

Rule 4: **if** *dptAvg* > 5924.933 **and** *dptEnt* < 4.3342 **then**
BERT lr=1e-5, epochs=4

Rule 5: **if** *cmxWds* < 0.1570 **then**
BERT lr=1e-5, epochs=6

Rule 6: **if** *dptAvg* < 670.860 **and** *dptEnt* > 1.1328 **then**
BERT lr=1e-4, epochs=6

Rule 7: **if** *dptStdAvg* < 2.2418 **and** *dptEnt* > 1.1328 **and** *dptEnt* < 2.4317 **and** *dptMed* < 27905.066 **then**
ALBERT lr=1e-5, epochs=6

Rule 8: **if** *dptEnt* < 0.8618 **then**
BERT lr=1e-5, epochs=4

Rule 9: **else**
BERT lr=1e-5, epochs=6

(a) *(b)*

Figure 2. Single run of the evolutionary model: (a) Behavior of the hhs and (b) Best hh learned. Source: Author's own elaboration.

Over the datasets of the genetic test group, the average performance of the final hh was **0.6913**, which is near the optimum of **0.7133**. This optimal value was calculated as the average of the performances of the best methods for each of the datasets in the group. Additionally, the hh could select the optimal classification methods for a total of 17 datasets (50% of all the datasets). This means that using a suitable method for each dataset produces better results than always using the same method for all the datasets. This also indicates that the hh is generalizing well for selecting an adequate method for unseen datasets.

Finally, the hh was compared with two state-of-the-art AutoML systems for text classification, AutoKeras[4] and AutoGluon[5]. Both systems were left at their default settings to avoid limiting their search space. The comparison in fitnesses is shown in Figure 3. AutoGluon found a solution for all the datasets in the genetic group. On the other hand, due to technical limitations of the AutoKeras system, results were obtained for only 24 datasets. In the plot, the three behaviors are similar, mainly because the hh, and both AutoML systems are based on similar deep learning methods.

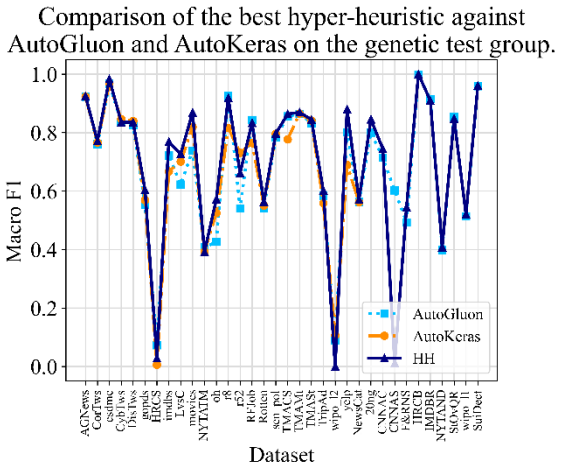


Figure 3. Comparison of the hh with AutoGluon and AutoKeras.
Source: Author's own elaboration.

For the 24 datasets, AutoKeras had an average performance of **0.6700**; and for the entire genetic group, AutoGluon obtained an average performance of **0.6860**. Both performances are below the one of the hh (**0.6963**).

Two independent Wilcoxon signed-rank tests [Demšar, 2006] were conducted to check if the differences found in the performances were significant. The null hypotheses are that hh and each system perform equally well. For the tests, the values used were a confidence level of $\alpha = 0.05$, a reference value of 81 for the AutoKeras-hh test ($N = 24$ datasets), and a reference value of 182 for the AutoGluon-hh test ($N = 34$ datasets). The values of the test were $z = 59$ for the AutoKeras-hh test and $z = 139.5$ for the AutoGluon-hh test. With those values, both null hypotheses can be rejected (z < reference value), and it could be concluded that hh performs better than either of these two systems.

In summary, the content of this section shows that the evolutionary model reached the contributions mentioned at the beginning. The model has been able to learn hhs that allow generalizing the method selection process for text classification datasets. The representation of hhs is simple and friendly, allowing any user to understand how the method selection process works. The meta-features based on data distribution are relevant to select suitable classification methods (in some cases, even optimal ones) for different datasets. The comparison between hh and AutoML systems shows that the approach addressed in this work has a

[4] Available at: https://autokeras.com/

[5] Available at: https://auto.gluon.ai/stable/index.html

future, finding that, in general, the performance of hh is superior. It is worth mentioning that hh uses a simple and efficient approach, quickly providing solutions to groups of datasets, which is different from what current AutoML systems do.

In this paper, a novel evolutionary model capable of generalizing the method selection process for text classification problems has been presented. This model learns hhs, as sets of if-then rules, through an evolutionary process using specific crossover and mutation operators. The rule representation makes it easy to understand how the method selection process is performed. For a given dataset, a hh determines the most appropriate classification method by evaluating a set of 16 meta-features that summarize the data distribution of the dataset. The classification method is selected from a pool of 12 different variants of the ALBERT and BERT deep learning transformers methods.

After the evolutionary process, the model obtained a hh that obtained a performance very close to the optimal value when tested with a group of unseen datasets (genetic test group). Inside the rules of the final hh, the instances of the classification method BERT have dominance, which also happens to be the optimal ones for several datasets. In the final hh, the meta-feature dptEnt was the most relevant since it is present in 4 rules. There are other eight meta-features evaluated by the rules of the final hh.

Finally, the performance of the hh was compared with those of two complex state-of-the-art AutoML systems, resulting in the hh obtaining better, statistically significant, performances than both systems.

There are different topics that can be addressed for future research. One of the most relevant is the expansion of the pool of classifiers, adding multiple classification methods with different approaches, such as ML methods that are less expensive than those used in this work. Another possibility is that, regardless of the hhs representation used, the problem can be further explored by developing a multi-objective model. This type of model can involve the time and computational resources that a method needs to be trained and tested. In such a way that the model can search for methods that have good performance and low computational cost. Finally, the set of meta-features that describe a dataset can be extended, adding some new ones that consider other types of textual representations (e.g., word vectors, embeddings, etc.), or based on the content of the documents (e.g., sentiment words, emojis, emoticons, etc.).

References

Chauhan, K., Jani, S., Thakkar, D., Dave, R., Bhatia, J., Tanwar, S., & Obaidat, M. S. (2020, March). Automated machine learning: The new wave of machine learning. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)* (pp. 205-212). IEEE. DOI: 10.1109/ICIMIA48430.2020.9074859

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7,1-30. DOI: 10.5555/1248547.1248548.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. DOI: 10.48550/arXiv.1810.04805

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2020). Autogluon-tabular: Robust and accurate automl for structured data. DOI: 10.48550/arXiv.2003.06505

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. (2018). Practical automated machine learning for the automl challenge 2018. In *International Workshop on Automatic Machine Learning at ICML* (pp. 1189-1232). URL: https://sites.google.com/site/automl2018icml/

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). A survey on text classification algorithms: From text to predictions. *Information*, *13*(2), 83. DOI: 10.3390/info13020083

Gomez, J. C., Hoskens, S., & Moens, M.-F. (2017). Evolutionary learning of meta-rules for text classification. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 131-132). DOI: 10.1145/3067695.3075601

Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H. J., Escalera, S., Liu, Z., Jajetic, D., Ray, B., Saeed, M., Sebag, M., et al. (2019). Analysis of the automl challenge series. *Automated Machine Learning*, *177*. DOI: 10.1007/978-3-030-05318-5_10

Jin, H., Song, Q., & Hu, X. (2019). Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1946-1956). DOI: 10.1145/3292500.3330648

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. DOI: 10.48550/arXiv.1909.11942

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P. S., & He, L. (2020). A survey on text classification: From shallow to deep learning. DOI: 10.48550/arXiv.2008.00364

Madrid, J. (2019). Autotext: Automl for text classification. URL: http://inaoe.repositorioinstitucional.mx/jspui/handle/1009/1950

Madrid, J. G., & Escalante, H. J. (2019). Meta-learning of text classification tasks. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 24th Iberoamerican Congress, CIARP 2019, Havana, Cuba, October 28-31, 2019, Proceedings 24* (pp. 107-119). Springer International Publishing. DOI: 10.1007/978-3-030-33904-3_10