Optimizing Natural Language Processing Pipelines: Opinion Mining Case Study

Suilan Estevez-Velarde[1](✉), Yoan Gutiérrez[2], Andrés Montoyo[3], and Yudivián Almeida-Cruz[1]

[1] School of Math and Computer Science,
University of Habana, Habana, Cuba
{sestevez,yudy}@matcom.uh.cu

[2] University Institute for Computing Research (IUII),
University of Alicante, Sant Vicent del Raspeig, Spain
ygutierrez@dlsi.ua.es

[3] Department of Languages and Computing Systems,
University of Alicante, Sant Vicent del Raspeig, Spain
montoyo@dlsi.ua.es

Abstract. This research presents NLP-Opt, an Auto-ML technique for optimizing pipelines of machine learning algorithms that can be applied to different Natural Language Processing tasks. The process of selecting the algorithms and their parameters is modelled as an optimization problem and a technique was proposed to find an optimal combination based on the metaheuristic Population-Based Incremental Learning (PBIL). For validation purposes, this approach is applied to a standard opinion mining problem. NLP-Opt effectively optimizes the algorithms and parameters of pipelines. Additionally, NLP-Opt outputs probabilistic information about the optimization process, revealing the most relevant components of pipelines. The proposed technique can be applied to different Natural Language Processing problems, and the information provided by NLP-Opt can be used by researchers to gain insights on the characteristics of the best-performing pipelines. The source code is made available for other researchers. In contrast with other Auto-ML approaches, NLP-Opt provides a flexible mechanism for designing generic pipelines that can be applied to NLP problems. Furthermore, the use of the probabilistic model provides a more comprehensive approach to the Auto-ML problem that enriches researcher understanding of the possible solutions.

Keywords: Natural Language Processing · Pipeline optimization · Metaheuristics · Opinion mining

1 Introduction

Text mining tasks generally require the evaluation of different machine learning and natural language processing algorithms [5]. Selecting the right algorithms

I. Nyström et al. (Eds.): CIARP 2019, LNCS 11896, pp. 163–173, 2019.
https://doi.org/10.1007/978-3-030-33904-3_15

for each task requires either a comprehensive knowledge of the problem domain, previous experience in similar contexts, or trial and error [4]. These algorithms interact with each other creating a complex pipeline [11]. As the possible combinations of algorithms for creating a pipeline could vary widely [7], it is unfeasible to evaluate all of them. Evaluating only a few combinations is risky if the combinations are manually selected because we may miss the best solutions. This is also the case when using grid-search, and similar techniques, which are either too expensive when the grid resolution is very high, or the optimal parameter combination with a lower resolution cannot be found. Consequently, a new research area has emerged recently, namely Auto-ML (Automatic Machine Learning), which designs techniques for automatically finding the optimal configuration of machine learning pipelines through the use of optimization techniques.

Current Auto-ML approaches are often designed as black-box optimization tools, and are thus applicable mainly to machine learning problems in which the input is a feature matrix. However, Natural Language Processing (NLP) problems have to deal with a variety of tasks before obtaining the feature matrix, involving preprocessing (e.g., removing stopwords, performing stemming, etc.) and vectorization (e.g., using TF-IDF or embeddings). For this reason, applying existing Auto-ML techniques to NLP problems still requires a degree of manual feature engineering. An Auto-ML tool suitable for NLP problems needs to consider also the preprocessing and vectorization tasks in order to completely automate the solution of the NLP problem.

This paper proposes a Natural Language Processing Optimizer –hereafter referred to as NLP-Opt– for enhancing pipelines of algorithms for NLP problems where several choices are possible at each stage. NLP-Opt delivers a combination of algorithms and their parameters in a feasible time frame, by performing an intelligent exploration of all possible combinations. In this paper, a pipeline is defined as a sequence of algorithms, each one receiving as input the output of the previous stage. We model the selection of algorithms and features as an optimization problem and propose using the metaheuristic Population-Based Incremental Learning (PBIL) proposed by Baluja [3] to find an optimal combination. To validate the methodology behind NLP-Opt, we focus on a specific NLP task, i.e., Opinion Mining [14] in Spanish language tweets. Opinion Mining is an NLP problem that involves several tasks for preprocessing and extracting relevant features, as well as classification, and posses an interesting challenge in terms of modelling the possible pipelines to solve it.

The following tangible results are highlighted:

– A formal optimization model for a generic pipeline of algorithms, and an optimization technique based on PBIL metaheuristic (Sect. 3) to find the best combination of algorithms (i.e., NLP-Opt).
– NLP-Opt is validated in an NLP problem, in this case opinion mining, using a specific pipeline designed for this task (Sect. 4).
– NLP-Opt permits the discovery of the most relevant parameters for each pipeline component (Sect. 4.1).

- The source code is made available to encourage other authors to further develop this line of research[1].

The rest of the paper is organized as follows: Sect. 2 gives a brief overview of current research in Auto-ML techniques; Sect. 3 explains the core of our proposal, NLP-Opt; Sect. 4 presents an application of NLP-Opt for opinion mining in Twitter, to illustrate the types of problems and domains where NLP-Opt is suitable; Subsect. 4.1 describes the experiments performed and the characteristics of the corpus used in the algorithm's evaluation; Sect. 4.2 presents a general discussion and the analysis of the results obtained; and finally, Sect. 5 presents the main conclusions of the research and outlines the possible future lines of development.

2 Related Work

The rising complexity of modern machine learning techniques for users not specialized in the ML field has motivated research into techniques for the automatic configuration and optimization of machine learning pipelines, i.e., Auto-ML.

The most common approaches for designing Auto-ML techniques are based on Evolutionary Computation (EC), Bayesian Optimization (BO) and Monte Carlo Tree Search (MCTS). Examples of EC are Recipe [17] and TPOT [13], while BO is used in Auto-Sklearn [6], Auto-Weka [19] and Hyperot [8]. ML-Plan [12] is a recent technique that models machine learning pipelines as hierarchical task networks and applies MCTS.

The previous examples provide out-of-the-box machine learning pipelines, which can be automatically configured for a specific dataset, selecting among a large collection of shallow classification, dimensionality reduction, pre-processing and feature selection techniques. Auto-ML tools provide an end-to-end solution for the optimization of machine learning pipelines. However, several techniques have been proposed to deal with specific parts of the machine learning process, rather than whole pipelines [18]. For example, Harmony Search has been used for feature selection in text clustering [1]. Also, several different metaheuristics have been compared for dynamic reduction in text document clustering [2].

The main challenge when applying current Auto-ML tools to NLP problems is related with the feature engineering phase. The existing tools assume an input representation based on feature matrix, hence the researcher still needs to manually select which preprocessing tasks (e.g., stopword removal, stemming, etc.) to perform and how to encode them in features (e.g., using TF-IDF or word embeddings).

3 Proposal for Pipeline Optimization

NLP-Opt, based on the PBIL metaheuristic [3], is designed to find out the optimal configuration of a machine learning pipeline for Natural Language Processing. Pipelines are represented as a sequence of stages for which there are different

[1] For blind-review purposes this link is omitted.

options to apply (i.e., there are different algorithms or techniques for solving each stage).

The possible combinations of a generic pipeline with m stages are modeled as a subset of the space N^m. Each vector θ in this subset corresponds to one specific combination of algorithms and vice-versa. The cost function P_θ corresponds to the evaluation of the precision in that combination. The problem of finding the best combination could be modelled as an optimization problem in integers:

$$\max_{\theta}\{P_\theta(X_t, y_t, X_p, y_p) \in [0,1]\}$$

$$\text{s.t. } \theta \in d_1 \times d_2 \times \ldots \times d_m \subset N^m$$

where:

P_θ is a quality metric of the solution for the combination θ;
X_t, y_t is a list of data points and classes of the training set;
X_p, y_p is a list of data points and classes of the test set; and,
d_i is the domain of the i^{th} component, i.e., the number of options in that stage.

The optimization process of NLP-Opt consists of a generation and evaluation loop that simultaneously searches at each stage for the combination of options that provides the optimal fitness (i.e., the optimal performance when trained on a specific dataset). Pipelines are generated by sampling from a probability distribution that represents which options are more likely to produce the best performance. During the optimization loop, the probability distribution is adjusted according to a selection of the best performing pipelines discovered for each iteration. Figure 1 shows a graphic representation of this process.

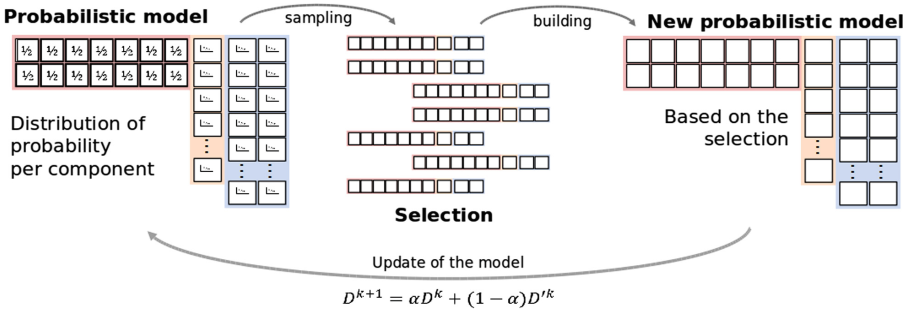

Fig. 1. Representation of process of optimization reusing PBIL.

In the optimization problem the search space is considered as a stochastic cost function. Hence, it is desirable to obtain not only an optimal solution, but also a structure that helps to identify the best combinations, irrespective of the bias introduced by the noisy evaluations. In this sense, we decided to reuse PBIL for providing not only a particular final solution, but also a distribution function for each component of the solution. The analysis of this distribution enables the selection of the most promising algorithms according to their behavior during the entire experimentation period and avoids bias towards a single noisy result.

PBIL [3] is one of the first Estimation of Distribution Algorithms—EDA—created [10]. This is a population-based algorithm with an incremental learning designed for combinatorial problems. Every component of the solution vector is modeled as a gene, which is associated with a probability distribution over its discrete values.

The algorithm NLP-OPT initialize with a uniform distribution for each gene. These distributions change in time, based on the probability distribution of the genes of the best solutions in the population. In every iteration the best n solutions are selected and the probability distribution of each component is modified. The speed of the model adjustment is controlled by a parameter α called "learning factor". Algorithm 1 shows a simplified implementation of this process.

Algorithm 1. Human Language Technology Optimization (NLP-Opt)

```
—— **Define Problem**
$p \leftarrow$ defines a pipeline (number of stages)
$p_i \leftarrow$ defines for each component of p all the possibles options

—— **Parameters of the optimization algorithm**
$popsize \leftarrow$ number of pipelines to evaluate in each iteration
$n \leftarrow$ number of best pipelines selected each iteration
$\alpha \leftarrow$ learning factor

—— **Initialization**
$D \leftarrow \{D_1, ..., D_m\}$ marginal distribution by $p_i$ component (uniform)
    In this moment all options are equally likely
Best $\leftarrow \square$

—— **Optimization process**
**while** generations remain **do**
    $P \leftarrow \{\}$

    **for** $i = 1 \ldots popsize$ **do**
        $S_i \leftarrow$ pipeline built by sampling of $D$
        $f(S_i) \leftarrow$ calculate *fitness* of $S_i$
        **if** Best $= \square$ or $f(S_i) < f(\text{Best})$ **then**
            Best $\leftarrow S_i$
        **end if**
        $P \leftarrow P \bigcup \{S_i\}$
    **end for**

    $P^* \leftarrow$ best $n$ pipelines of $P$

    —— **Promote generation of the best pipelines**
    **for** each gene $j$ in $D$ **do**
        $N_j \leftarrow$ marginal distribution of the gene $j$ in $P^*$
        $D_j \leftarrow \alpha N_j + (1 - \alpha) D_j$
    **end for**
    ——
**end while**
——
**return** Best
```

4 NLP-Opt Applied to Opinion Mining

This section presents a case study involving a classic NLP problem: opinion mining in Twitter. The process of solving and evaluating the Opinion Mining problem has been tackled using different approaches [9,14,21], of which Supervised classification is the most common [14,16]. Hence, for this case study, we decide to define the process of classifying messages in different stages that can be analyzed independently. The pipeline consists of 4 stages: (I) text preprocessing; (II) dimensionality reduction; (III) classification into objective-subjective; and (IV) classification into positive-neutral-negative. For each stage we select a wide range of possible algorithms to explore. This structure, with several different strategies available for the preprocessing and dimensionality reduction steps, and having two different classification steps, makes Opinion Mining a non-standard learning problem unsuitable for black-box Auto-ML tools.

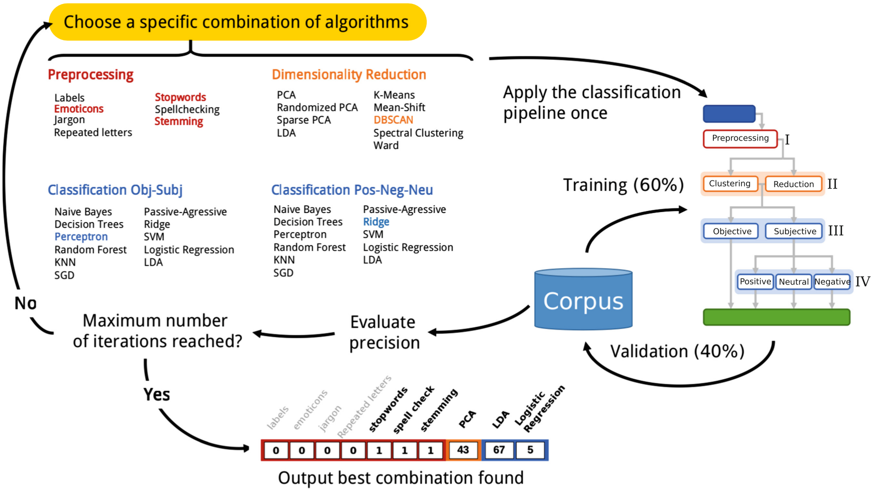

Fig. 2. Graphic representation of the optimization process, from the point of view of the Opinion Mining case study. The represented loop occurs inside the fitness evaluation of the PBIL algorithm, for each of the combinations analyzed during the whole optimization process. The color-highlighted options are an illustrative example of a possible combination.

In our proposal we model the algorithmic pipeline as a vector and apply the optimization process described in Sect. 3 over the space of all possible solutions. Each solution (combination of algorithms) is fitted on a training set, and its performance is measured on a separate test set. This process is repeated with various combinations of algorithms, until convergence is achieved or a predefined number of iterations is reached. At this point, we obtain the combination of algorithms that performs best on the validation set. Figure 2 shows this optimization process in the opinion mining task. The source code is made available to encourage other authors to further develop this line of research (See footnote 1.).

4.1 NLP-Opt Experimental Results

The optimization process described in this section optimizes all the steps of the pipeline *simultaneously*. For description purposes, we will present an analysis independently of each step of the pipeline. However, it is important to keep in mind that in each step, the "optimal" algorithm is determined in correspondence with the "optimal" options for the other steps. The algorithms used are implemented in *sklearn* [15], and highlighted in Fig. 2, Tables 2 and 3.

The parameters selected for the experimentation are 100 individuals per generation, with a selection by truncation of the best 20, and a learning factor $\alpha = 0.1$, following the recommended parameters in the literature. A higher learning rate has been shown to hinder convergence, and 0.1 seems to provide a fairly good performance in several experiments performed by Baluja et al. [3]. The population size is chosen to allow sufficient variation in the pipelines sampled for each generation. We establish a limit of a sufficiently high number of iterations –1,000 generations– and the metaheuristic evolution was monitored until a convergence in most of the components was observed. At this point, the experimentation reached 126 iterations, for a total of $12,600$ evaluations of the objective function.[2]

The experimentation was performed in a single commodity computer, with an i5 microprocessor and GB of RAM memory. A total of approximately 720 computing hours were used, with an average 3.4 min per pipeline, although the actual computational cost of each pipeline varies with the complexity of the algorithms involved. The total number of possible combinations is $2^7 \cdot 52 \cdot 67 \cdot 67$, which equates to $29,878,784$. At this rate, performing the experiment for all the possible combinations would require more than 1175 years of computing power.

Figure 3 shows the best combination found by NLP-Opt expressed in terms of the corresponding algorithms associated to each solution.

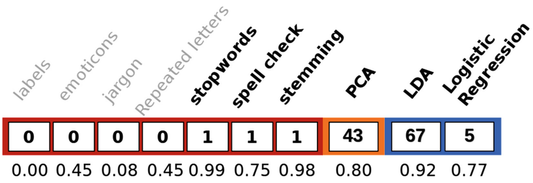

Fig. 3. Best solution found by NLP-Opt.

The experimentation process was performed on the *general* corpus, from the collection of tweets published in TASS 2014 [20]. This corpus has the advantage of being small ($6,000$ training examples), which decreases the computational cost of the experimentation.

[2] As illustrated in Fig. 2, each evaluation consists of a full run of the classification pipeline with a specific combination of preprocessing, reduction and classification.

The corpus is a collection of Spanish language messages from approximately 150 personalities in the field of politics, economics, communication, mass media and culture. The authors are of different nationalities, including Spanish, Mexican, Colombian, Puerto Rican and United States, among other countries, which provides a more diverse linguistic scenario. The Spanish language was chosen given that is a less pervasive language in terms of linguistic resources and thus an interesting challenge. In all cases, a cross-validation was carried out comprising a subset of 60% of the examples for the training set and the remaining 40% for validation.

The mean accuracy obtained is of 0.542 with a standard deviation of 0.0095. Comparatively, the accuracy obtained with a random baseline is of 0.299 with a standard deviation of 0.0078, for a relative improvement of the 44.9%. Table 1 summarizes these results, including a test for a statistical significant (t-Student).

Table 1. Comparative results of the final proposal according to NLP-Opt and a random classification. μ is the mean accuracy and σ is the standard deviation of the mean accuracy in 30 independent evaluations with 60% training and 40% validation.

Algorithm	μ	σ	p-value
Random	0.299	0.0078	-
NLP-Opt (Fig. 3)	0.542	0.0095	$4.294e{-68}$

4.2 Results and Discussion

The probability model generated determines, for each stage of the pipeline, a probability distribution of the best options (i.e., algorithms) to use in that stage. By analyzing this probability distribution, we can obtain insights about the influence that each specific step, technique or algorithm produces in the overall process.

Table 2 shows the probability distribution obtained for each stage of preprocessing and dimensionality reduction. As it can be seen, the first 4 tasks are not beneficial, while the last ones are recommended for use. The dimensionality reduction algorithms were modeled as a single component in the input vector. According to Table 2 the best options are two variants of the algorithm Principal Component Analysis (PCA). These two options are also the only ones that display an improvement in the final precision with respect to not using any algorithm for dimensionality reduction.

The process of classification is modeled similar to the dimensionality reduction process. With this aim (i.e. classifying) we defined two components, mentioned in Sect. 4, which were labelled as Objective-Subjective and Positive-Negative-Neutral, respectively.

According to the probabilities measured, see Table 3, in the first classification stage (i.e. III) the algorithm that obtains the highest probability is Linear

Discriminant Analysis. In the second classification stage (i.e IV), the algorithm that obtains the highest probability is Logistic Regression. The probability distribution obtained by the algorithms highlights an advantage for algorithms with a linear decision surface. This fact can be justified since lineal algorithms have less internal parameters to learn and therefore are less likely to overfit a small training set as the one used in this study.

Table 2. Best success probability of preprocessing algorithms (**left**) and dimensionality reduction algorithms (**right**).

Phase	**Don't use**	**Use**
tags	**1.000**	0.000
smiles	**0.553**	0.447
jargon	**0.916**	0.084
repetition	**0.548**	0.452
stop words	0.008	**0.992**
spelling	0.255	**0.745**
stemming	0.021	**0.979**

Algorithm	**Probability**
LDA	0.0000
S-PCA	0.0000
ICA	0.0000
None	0.0026
Mean Shift	0.0034
DBSCAN	0.0036
K-Means	0.0039
Ward	0.0047
Spectral Clustering	0.0066
R-PCA	**0.1724**
PCA	**0.8029**

Table 3. Best success probability of classification algorithms, discovered.

Algorithm	Obj-Subj	Pos-Neg-Neu
Random	0.0000	0.0000
Naive Bayes	0.0000	0.0000
Decision Tree	0.0000	0.0000
Perceptron	0.0000	0.0000
Random Forest	0.0004	0.0000
KNN	0.0006	0.0000
SGD	0.0007	0.0001
Passive-Aggressive	0.0042	0.0000
Ridge	0.0278	0.0079
SVM	0.0062	**0.2245**
Logistic Regression	0.0411	**0.7672**
Linear Discriminant	**0.9189**	0.0002

5 Conclusions and Future Work

This paper presents NLP-Opt, a metaheuristics-based Auto-ML strategy for optimizing NLP pipelines. In contrast with other Auto-ML approaches, NLP-

Opt provides a flexible mechanism for designing generic pipelines that can be applied to NLP problems, where the preprocessing and vectorization tasks are also automated. Furthermore, NLP-Opt not only finds the best pipeline architecture, but also provides probabilistic information about the exploration processes involved. This information can be used by researchers to gain insights on the characteristics of the best performing pipelines.

Improving the representation for including continuous features is a promising direction for future work. This would allow researchers to optimize not only what algorithms to select, but also the specific parameters for those algorithms, without having to define a priori a fixed subset of parameter combinations. Even though the focus of this work is NLP classification problems, the technique proposed can be easily extended to any domain where a pipeline of algorithms is used. Therefore, in future research, the technique will be adapted to more complex pipelines that are not limited to the NLP domain. Specifically, we will explore extending NLP-Opt to the optimization of deep learning architectures.

Acknowledgments. This research has been supported by a Carolina Foundation grant in accordance with the University of Alicante and the University of Havana. This work has also been partially funded by both aforementioned universities, the Generalitat Valenciana and the Spanish Government through the projects SIIA (PROMETEU/2018/089), LIVINGLANG (RTI2018-094653-B-C22) and INTEGER (RTI2018-094649-B-I00).

References

1. Abualigah, L.M., Khader, A.T., Al-Betar, M.A.: Unsupervised feature selection technique based on genetic algorithm for improving the text clustering. In: 2016 7th International Conference on Computer Science and Information Technology (CSIT), pp. 1–6. IEEE (2016)
2. Abualigah, L.M., Khader, A.T., Al-Betar, M.A., Alomari, O.A.: Text feature selection with a robust weight scheme and dynamic dimension reduction to text document clustering. Expert Syst. Appl. **84**, 24–36 (2017). https://doi.org/10.1016/j.eswa.2017.05.002. http://www.sciencedirect.com/science/article/pii/S0957417417303172
3. Baluja, S.: Population-based incremental learning. A method for integrating genetic search based function optimization and competitive learning. Technical report, DTIC Document (1994)
4. Bishop, C.M.: Model-based machine learning. Phil. Trans. R. Soc. A **371**(1984), 20120222 (2013)
5. Dinakar, K., Reichart, R., Lieberman, H.: Modeling the detection of textual cyberbullying. Soc. Mobile Web **11**(02), 11–17 (2011)
6. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Advances in Neural Information Processing Systems, pp. 2962–2970 (2015)
7. Jain, S., Shukla, S., Wadhvani, R.: Dynamic selection of normalization techniques using data complexity measures. Expert Syst. Appl. **106**, 252–262 (2018). https://doi.org/10.1016/j.eswa.2018.04.008. http://www.sciencedirect.com/science/article/pii/S095741741830232X

8. Komer, B., Bergstra, J., Eliasmith, C.: Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In: ICML Workshop on AutoML, pp. 2825–2830. Citeseer (2014)
9. Kontopoulos, E., Berberidis, C., Dergiades, T., Bassiliades, N.: Ontology-based sentiment analysis of twitter posts. Expert Syst. Appl. **40**, 4065–4074 (2013)
10. Luke, S.: Essentials of Metaheuristics. Lulu 2009. http://cs.gmu.edu/~sean/book/metaheuristics/ (2011)
11. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55–60 (2014)
12. Mohr, F., Wever, M., Hüllermeier, E.: ML-Plan: automated machine learning via hierarchical planning. Mach. Learn. **107**(8), 1495–1515 (2018). https://doi.org/10.1007/s10994-018-5735-z
13. Olson, R.S., Moore, J.H.: TPOT: a tree-based pipeline optimization tool for automating machine learning. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Automated Machine Learning. TSSCML, pp. 151–160. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_8
14. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retrieval **2**(1–2), 1–135 (2008)
15. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
16. Rosenthal, S., Farra, N., Nakov, P.: Semeval-2017 task 4: sentiment analysis in Twitter. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pp. 502–518 (2017)
17. de Sá, A.G.C., Pinto, W.J.G.S., Oliveira, L.O.V.B., Pappa, G.L.: RECIPE: a grammar-based framework for automatically evolving classification pipelines. In: McDermott, J., Castelli, M., Sekanina, L., Haasdijk, E., García-Sánchez, P. (eds.) EuroGP 2017. LNCS, vol. 10196, pp. 246–261. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55696-3_16
18. Salimans, T., Ho, J., Chen, X., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864 (2017)
19. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: 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, pp. 847–855. ACM (2013)
20. Villena-román, J., Lana Serrano, S., Martínez Cámara, E., González Cristóbal, J.C.: TASS workshop on sentiment analysis at SEPLN. Procesamiento del Lenguaje Natural (2013)
21. Zhang, L., Ghosh, R., Dekhil, M., Hsu, M., Liu, B.: Combining lexicon based and learning-based methods for twitter sentiment analysis. HP Laboratories, Technical Report HPL-2011 89 (2011)