Meta-learning of Textual Representations

Jorge G. Madrid[1], Hugo Jair Escalante[1,2(✉)], and Eduardo Morales[1]

[1] Computer Science Department, INAOE, Puebla, Mexico
{jgmadrid,hugojair,emorales}@inaoep.mx
[2] Computer Science Department, CINVESTAV, Zacatenco, Mexico City, Mexico

Abstract. Recent progress in AutoML has lead to state-of-the-art methods (e.g., AutoSKLearn) that can be readily used by non-experts to approach *any* supervised learning problem. Whereas these methods are quite effective, they are still limited in the sense that they work for tabular (matrix formatted) data only. This paper describes one step forward in trying to automate the design of supervised learning methods in the context of text mining. We introduce a meta learning methodology for automatically obtaining a representation for text mining tasks starting from raw text. We report experiments considering 60 different textual representations and more than 80 text mining datasets associated to a wide variety of tasks. Experimental results show the proposed methodology is a promising solution to obtain highly effective off the shell text classification pipelines.

Keywords: Text mining · Meta-features · Text classification

1 Introduction

Nowadays, the success of machine learning systems relies on the knowledge of human-experts that according to their experience design and test multiple models extensively to select the best modeling option. Although effective, this strategy is not only time consuming but also impractical since an expert is not always available. This has motivated the increasing demand for easy-to-use automated machine learning solutions. In this context, AutoML is the field of research aiming to generate machine learning models without any human supervision. Recent progress in AutoML has lead to quite effective and competitive solutions when dealing with tabular (matrix formatted) data, see e.g., [5,7,16]. However, these techniques are still limited in the sense that they require the user to transform raw data into a tabular representation. This step relies heavily on the expertise of users.

Text classification is one of the most studied tasks in Natural Language Processing (NLP), this is because of the number of applications that can be

This work was partially supported by CONACyT under grant *Aprender Objetos de Internet para Buscarlos con un Robot*, ID: CB-250938.

approached as text classification problems (e.g. sentiment analysis, topic labeling, spam detection, and author profiling among others). Many techniques for pre-processing, feature extraction, feature selection and document representation have been developed over the last decades. Despite all this progress by the NLP community, it is still an expert who designs the pipeline for text classification systems that includes one or many of such techniques, each of which usually requiring the fine-tuning of hyperparameters.

In this work we take a step towards the automated generation of the classification pipelines for text classification by focusing on the representation. Thus, our goal is to automate the process of determining the best representation to approach a text classification task starting from raw text. Unlike other data types, language/text provides an unstructured and rich source of information, hence selecting the adequate representation for text will have a direct impact on the performance of text mining solutions. In fact, representing texts has been one of the most studied venues in NLP. We propose a meta-learning solution to automatically determine the best representation given a dataset of raw text. This is a first step towards the full automation of the generation of text classification pipelines. We propose a number of meta features some of which are extracted directly from raw text, and most of them not used before for meta-learning. We report experimental results considering 60 textual representations and more than 80 text mining tasks. Our results show that the proposed meta features successfully characterize text mining tasks and that an AutoML solution for text mining (AutoText) is feasible. Our work is among the first to approach text classification via meta-learning from raw data and it is by far the largest study on meta learning in the context of text mining.

2 Related Work

In the context of text mining, few works have explored the automated selection of different parts of classification pipelines. With experiments in the Reuters-21578 corpus, Lam and Lai [11] proposed to characterize documents with 9 (meta) features and to predict the classification error of different models using data from a previous phase, thus recommending a classification model. More recently, [19] searched for text representations with Bayesian Representation [15]. Their search space was limited to only word n-grams and experiments were performed in 8 datasets: 4 sentiment analysis tasks and 4 topic classification tasks. Nevertheless, they outperformed every linear classifier reported until their publication date. Despite their limited scope, given the lack of data and computational resources of the time, this work represents one of the first meta-learning approaches for text classification.

Other works have explored different meta-learning approaches for text classification in small-scale, for example, Gomez et al. [9] addressed the problem with evolutionary computation methods and using 11 meta-features. In a broader approach Ferreira and Brazdil [6] recommend *pipelines* with Active Testing Method, in their work they also present statistical analysis of 48 preprocessing methods and 8 classifiers.

Another related work is that by [14] where a set of features derived from the text was proposed for characterizing short-text corpora in the context of clustering. Although the goal of such reference was to characterize the harness of text corpora and not AutoML, such work is relevant because their features inspired some of the meta-features considered in this paper. In Sect. 3 we describe how we combined this set of features with other proposed ones for characterizing text collections in the context of automatic text mining.

Meta-learning has been studied for a while in the broad machine learning context [1–3,17,18]. However, it is only recently that it has become a mainstream topic, this mainly because of its successes in several tasks. For instance, Feurer et al. [8] successfully used a set of meta-features to warm-start a hyperparameter optimization technique in the popular state-of-the-art AutoML solution *Autosklearn*. Likewise, the success of deep learning together with the difficulty in defining appropriate architectures and hyperparameters for users, has motivated a boom on neural architecture search, where meta-learning is becoming common [4].

In this paper we propose a novel approach to meta-learning of text representations. We propose a novel set of meta-features, comprising standard meta-features from the machine learning literature, features that have been used for other problems than meta-learning and novel meta-features that have not been used previously. Some of which are derived directly from raw text and aim at capturing complex language patterns. We approach the problem of recommending textual representations. Whereas this problem has been addressed in previous work, such references have considered only a few representations and a very limited number of meta-features (up to 11).

To the best of our knowledge this is the largest scale study on meta-learning in the context of text mining. Whereas results are promising, please note that this is only a first step towards the ultimate goal of automating the text mining process.

3 Recommending Textual Representations

We introduce a meta-learning method that takes as input the labeled raw text from a corpus associated to a text classification task and automatically selects a representation. The method recommends vector representations for text classification tasks based on which one worked best for *similar* tasks. In order to do so we define a set of meta-features and perform extensive experiments on 81 different text classification tasks. Although this approach is common within meta-learning [17], it has not been widely explored for text classification. In fact, previous work (see Sect. 2) has considered small subsets of generic meta-features.

Table 1 sums up the feature extraction methods that with some preprocessing processes or hyper-parameters gives a total of 60 representations, while not exhaustive, our work is the first to consider representations not only based on simple features, but also those based on topic modeling, embeddings, and semantic analysis, we also included a representation based on the word percentage of categories from LIWC2007 dictionary. Furthermore, the output of our method can be useful for both human experts designing text classification pipelines and for complementing other optimization methods for AutoML (e.g. it can be used for warm-starting Bayesian Optimization [8] for a wider search space of text representations or easily combined with existing AutoML solutions).

Table 1. Representations considered

Features	Hyper-parameters
N-grams	[words, char], stop_words[None, 'English'], range[1, 3], weight[bi, tf, tfidf]
LDA	stop_words[None, 'English']
LSA	stop_words[None, 'English'], weight[tf, tfidf]
LIWC [13]	categories[64]
W2V [12]	pre_trained[True, False], vector[mean, sum], dimension[300]

The proposed method comprises 2 stages, an offline phase where it *learns how to learn* and a predicting phase where it uses the data collected in phase 1 to recommend a text representation for classifying.

A human-expert uses knowledge acquired in the past when a new task is presented, equivalently, *meta-learning* imitates this reasoning. Our method applies meta-learning to learn from the performance of different representations on a number of corpora. Namely, we defined 72 meta-features to characterize 81 text corpora and performed an exhaustive search for the performance of 60 representations. A *knowledge base* is built associating the performance of each representation with a task, described by the vector of meta-features. Traditionally, meta-features extract meta-data from a dataset such as statistics of its distribution or simple characteristics like the number of classes and attributes, in our proposed set we contemplate this type of features as well as other attributes extracted directly from the raw text. The proposed meta-features are described below. For clarity we have divided them in groups.

– **General meta-features.** The *number of documents* and the *number of categories.*
– **Corpus hardness.** Most of these originally used in [14] to determine the hardness of short text-corpora.
  Domain broadness. Measures related to the thematic broadness/narrowness of words in documents. We included measures based on the vocabulary length and overlap: *Supervised Vocabulary Based (SVB)*, *Unsupervised Vocabulary Based (UVD)* and *Macro-averaged Relative Hardness (MRH).*
  Class imbalance. Class Imbalance (CI) ratio.
  Stylometry. Stylometric Evaluation Measure (SEM)
  Shortness. Vocabulary Length (VL), *Vocabulary Document Ratio (VDR)* and average *word length.*
– **Statistical and information theoretic.** We derive meta-features from a document-term matrix representation of the corpus.

min, max, average, standard deviation, skewness, kurtosis, ratio average-standard deviation, and entropy of: vocabulary distribution, documents-per-category and words-per-document:
Landmarking. 70% of the documents are used to train 4 simple classifiers and their performance on the remaining 30% was used based on the intuition that some aspects of the dataset can be inferred: *data sparsity - 1NN, data separability - Decision Tree, linear separability - Linear Discriminant Analysis, feature independence Naïve Bayes.* The *percentage of zeros* in the matrix was also added as a measure for sparsity.
Principal Components (PC) statistics. Statistics derived from a PC analysis: *pcac* from [9]; for the first 100 components, the same statistics from documents per category and their *singular values sum, explained ratio and explained variance*, and for the first component its *explained variance.*
- **Lexical features.** We incorporated the distribution of parts of speech tags. We intuitively believe that the frequency of some lexical items will be higher depending on the task associated to a corpus, for instance a corpus for sentiment analysis may have more adjectives while a news corpus may have less. We tagged the words in the document and computed the average number of *adjectives, adpositions, adverbs, conjunctions, articles, nouns, numerals, particles, pronouns, verbs, punctuation marks* and *untagged words* in the corpus.
- **Corpus readability.** Statistics from text that determine readability, complexity and grade from textstat library[1]: *Flesch Reading Ease, SMOG grade, Flesch-Kincaid grade level, Coleman-Liau index, automated readability index, Dale-Chall readability score, the number of difficult words, Linsear Write formula, Fog scale,* and *estimated school level to understand the text.*

Apart from general, statistical and PC based, the rest of the listed features have not been used in a meta-learning context. After the offline phase takes place, for a new task the same meta-features are extracted and compared with the prior knowledge, to recommend a representation. We considered 4 strategies that leverage learned experiences and make predictions for a new task, these are described below

(1) Using directly the representation with best performance of the *nearest* corpus. This strategy directly follows the idea of finding the most similar task in order to know what model will work best. The euclidean distance is used to determine the *similarity* between the new task and those in the knowledge base. This approach can also be seen as classifying unseen tasks with a Nearest Neighbors algorithms using only 1 neighbor, in which case each of the 60 representations constitutes a class.
(2) Predicting the representation as a classification problem, where each representation is a class and every prior task is a sample represented by its 72 meta-features. In this case every sample was labeled with the representation with best performance as its class, thus, the problem is to select the *correct* class finding patterns among the tasks and using 81 samples for training.

[1] https://github.com/shivam5992/textstat.

Since the dimensionality of this problem is big given the number of samples available this isn't an easy task, so we tested different classification models. In the end, a Random Forest was selected as the classifier for this strategy. Hyper-parameters of this RF model are listed in Table 2.

(3) Predicting the performance for every representation and selecting the one with the smallest error. In this strategy 60 different regression models are needed, one for each representation, they are trained using the performance of each representation for the different tasks, the objective is to correctly predict the performance for each representation given a new task (described by the same 72 meta-features). As for (2) several models were trained and compared, finally, a Random Forest Regressor was found to work best.

(4) Predicting the rank of each representation and selecting the one with best predicted rank. 60 regression models are trained with performances in 81 different tasks. Given a new task the 60 trained models predict the expected rank for each representation, the results are ordered and the representation with lowest rank is recommended. Like before we compared various regression models and again regression with Random Forest was selected (see Table 3).

Table 2. Hyper-parameter for the Random Forest Classifier used in strategy (2).

Hyper-parameter	Value
Estimators	200
Quality criteria	Gini
Max depth	Unlimited
Min features	2
Max features	$\sqrt{\lvert features \rvert}$

Table 3. Hyper-parameter for the Random Forest Regressor used in strategies (3) and (4).

Hyper-parameter	Value
Estimators	200
Quality criteria	Mean absolute error
Max depth	Unlimited
Min features	2
Max features	$\lvert features \rvert$

For strategies 2–4 different classification and regression models were tested, a Random Forest classifier was selected for strategy 2 and Random Forest regressor for both strategies 3 and 4. Once the representation is chosen, an SVM classifier with linear kernel is used in every case to train and make predictions with the new corpus.

4 Experiments and Results

For the experimental evaluation we collected 81 publicly available text corpora, each associated with a different classification task, most of which can be categorized as one of 6 common NLP tasks: authorship attribution, author profiling, topic/thematic classification, irony and deception detection. Some of these datasets are commonly used for benchmarks in text classification (e.g. Amazon, Dbpedia, 20NGs) while others have been used in competitions. After processing each corpus to share the same format and codification we extracted the 72 meta-features for each of the 81 collections. To accelerate the meta-feature extraction process we limited the number of documents to 90,000 per category. The resultant matrix of size 81×72 comprises our *knowledge base* characterizing multiple corpora.

In an offline phase, for each classification task every representation was used for training and testing a classification model, the *performance* of each representation was calculated with 3-fold Cross validation, they were also ranked from best (1) to worst (60).

We evaluated the 4 meta-learning strategies with unseen tasks following a leave-one-out setting, using the results from 60 representations in the rest of the tasks as knowledge to decide which representation to recommend. The objective for the strategies, then, is to select what in exhaustive search was found to be the *best* representation. We compared the average performance achieved by our strategies in 5 runs against the best solution found and the average performance of all of the considered representations. Table 4 shows the average performance for each strategy after 5 runs in terms of the average accuracy and average rank. Figure 1 depicts the performance of our method and the baselines in 9 corpora (we selected these representative corpora to cover a wide variety of tasks and because they are well known benchmarks).

Table 4. Average accuracy [0, 1] and average rank [60, 1] of different strategies in 81 corpus, the last row indicates the number of times the best representation was predicted. (1) Nearest corpus, (2) classification, (3) performance regression, (4) rank prediction.

Method	Best	(1)	(2)	(3)	(4)	Random
Avg accu	77.06 ± 0	73.75 ± 0	$\mathbf{75.25 \pm 0.12}$	73.34 ± 0.34	$\mathbf{75.20 \pm 0.07}$	68.45 ± 0
Avg rank	1.00 ± 0	14.20 ± 0	$\mathbf{8.71 \pm 0.46}$	14.30 ± 1.31	$\mathbf{8.51 \pm 0.34}$	30.30 ± 0
# of 1s	81.00 ± 0	17.00 ± 0	$\mathbf{25.80 \pm 0.45}$	4.20 ± 0.84	14.80 ± 0.84	0.00 ± 0

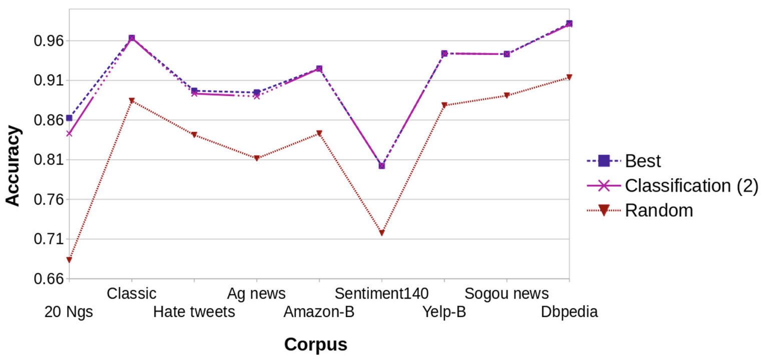

Fig. 1. Accuracy of (2) in 9 selected corpora.

The 4 strategies clearly outperform selecting a *random* representation (we illustrate this by averaging the results of all the representations). While in terms of average ranking they could be closer to the optimal, the average accuracy of (2) and (4) strategies was only 2% behind the best. (2) also found the best representation 35% of the time. Results show strong evidence that our meta-learning approach finds relations between corpora and pipeline performances that exploits prior knowledge for the autonomous classification of texts (Table 5).

From the 72 proposed meta-features we tested different subsets according to their Gini importance from the Random Forest used in strategy (2). A subset of 38 meta-features improved our results relatively by 8% with (2) and 38% with (1) in terms of average ranking. We also compared this subset against a subset comprised of 19 *traditional* meta-features used in related work. Using strategy (2) our subset outperformed the *traditional* one by almost 0.8% in average accuracy and 3 places in average rank. The results also showed a significant difference between both subsets ($p < .001$ Student's t-test). The subset of 38 meta-features is detailed in Table 6.

Table 5. Results after meta-feature selection

Method	(1)	(2)	(3)	(4)
Avg accu	75.16 ± 0	$\mathbf{75.39 \pm 0.13}$	73.57 ± 0.14	75.16 ± 0.05
Avg rank	8.68 ± 0	$\mathbf{8.00 \pm 0.47}$	14.42 ± 0.53	9.05 ± 0.25
# of 1s	$\mathbf{27.00 \pm 0}$	$\mathbf{26.44 \pm 0.56}$	6.40 ± 1.34	16.00 ± 0.87

In addition, we compared our strategies with commonly used representations such as pre-trained Word2Vec and Bag-of-Words outperforming them in average by 9% and 3% respectively, Fig. 2 depicts this comparison (between strategy (4) and W2V) in the 9 corpora we selected. Despite the robustness of such common representations their performance can usually be improved by fine tuning some

Table 6. 38 Meta-features selected by Gini importance

Meta-feature selection
average word length
document per category:
min
max
average
standard deviation
average/stdev
entropy
word per document:
average
skewness
entropy
Imalance Degree
SEM
UVB
SVB
MRH_J
VDR
max vocabulary
average vocabulary
sd vocabulary
skweness vocabulary
avg/stdev vocabulary
pca:
singular values sum
explained ratio
explained variance
explained variance (1)
pca max
pca skewness
pca kurtosis
data sparsity
data separability
linear separability
% of zeros
% of adpositions
% of adverbs
% of conjunctions
% of nouns
% of numbers
% of untagged words
difficult words

of their hyper-parameters or they are largely outperformed by another, as shown in the results the strategies are able to find these improvements.

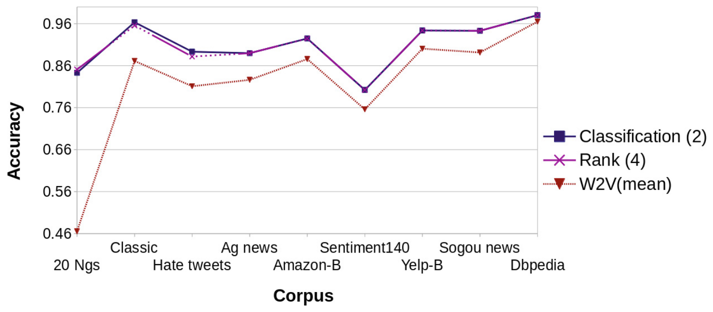

Fig. 2. Accuracy comparison between (2), (4) and Word2Vec in 9 corpora.

5 Conclusion and Future Work

We introduced a meta-learning method that takes as input a corpus and without human intervention builds a model to solve a text classification task focusing on the selection of a vector-based representation. The results show empirically that this approach is able to characterize tasks and approximate an optimal representation. Our work can not only recommend a single representation but also the best n representations using one of the strategies proposed to rank them, these can later be used to *warm-start* an optimization technique allowing us to expand the search space and, like in similar works on different fields [10], ideally finding pipelines that perform better than those designed by humans. Our also work comprises a first step towards the automated recommendation of full text classification pipelines. The source code of our method is available under an open source license at: https://github.com/jorgegus/autotext.

References

1. Bengio, Y.: Gradient-based optimization of hyperparameters. Neural Comput. **12**(8), 1889–1900 (2000)
2. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, pp. 2546–2554 (2011)
3. Bozdogan, H.: Model selection and Akaike's information criterion (AIC): the general theory and its analytical extensions. Psychometrika **52**(3), 345–370 (1987)
4. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: a survey. J. Mach. Learn. Res. **20**(55), 1–21 (2019)
5. Escalante, H.J., Montes, M., Sucar, L.E.: Particle swarm model selection. J. Mach. Learn. Res. **10**, 405–440 (2009). http://dl.acm.org/citation.cfm?id=1577069.1577084

6. Ferreira, M.J., Brazdil, P.: Workflow recommendation for text classification with active testing method. In: Workshop AutoML 2018@ ICML/IJCAI-ECAI (2018)
7. 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)
8. Feurer, M., Springenberg, J.T., Hutter, F.: Initializing Bayesian hyperparameter optimization via meta-learning. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
9. Gomez, J.C., Hoskens, S., Moens, M.F.: Evolutionary learning of meta-rules for text classification. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 131–132. ACM (2017)
10. Guyon, I., et al.: Analysis of the AutoML challenge series 2015–2018. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Automated Machine Learning. TSSCML, pp. 177–219. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_10
11. Lam, W., Lai, K.Y.: A meta-learning approach for text categorization. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 303–309. ACM (2001)
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
13. Pennebaker, J.W., Boyd, R.L., Jordan, K., Blackburn, K.: The development and psychometric properties of LIWC2015. Technical report (2015)
14. Pinto, D.: On clustering and evaluation of narrow domain short-text corpora. Ph.D., UPV (2008)
15. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, pp. 2951–2959 (2012)
16. 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, KDD 2013, pp. 847–855. ACM, New York (2013). https://doi.org/10.1145/2487575.2487629. http://doi.acm.org/10.1145/2487575.2487629
17. Vanschoren, J.: Meta-learning. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Automated Machine Learning. TSSCML, pp. 35–61. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_2. http://automl.org/book
18. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artif. Intell. Rev. **18**(2), 77–95 (2002)
19. Yogatama, D., Kong, L., Smith, N.A.: Bayesian optimization of text representations. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2100–2105 (2015)