On the Cost-Effectiveness of Stacking of Neural and Non-Neural Methods for Text Classification: Scenarios and Performance Prediction

Christian Gomes
UFMG - Brazil
christianreis@dcc.ufmg.br

Marcos André Gonçalves
UFMG - Brazil
mgoncalv@dcc.ufmg.br

Leonardo Rocha
UFSJ - Brazil
lcrocha@ufsj.edu.br

Sergio Canuto
IFG - Brazil
sergio.canuto@ifg.edu.br

Abstract

Neural network algorithms such as those based on transformers and attention models have excelled on Automatic Text Classification (ATC) tasks. However, such enhanced performance comes at high computational costs. Ensembles of simpler classifiers (i.e., stacking) that exploit algorithmic and representational complementarities have also been shown to produce top-notch performance in ATC, enjoying high effectiveness and potentially lower computational costs. In this context, we present the first and largest comparative study to exploit the cost-effectiveness of stacking of ATC classifiers, consisting of transformers and non-neural algorithms. In particular, we are interested in answering research questions such as: Is it possible to obtain an effective ensemble with significantly less computational cost than the best learning model for a given dataset? Disregarding the computational cost, can an ensemble improve the effectiveness of the best learning model? Besides answering such questions, another main contribution of this paper is the proposal of a low-cost oracle-based method that can predict the best ensemble in each scenario (with and without computational cost limitations) using only a fraction of the available training data.

1 Introduction

Natural Language Processing, Machine Learning and Data Mining techniques work together to automate the fundamental task of Automatic Text Classification (ATC). ATC automatically associates documents with classes, providing means to organize information, allowing better comprehension and interpretation of the data. Algorithms based on neural networks (e.g., BERT (Devlin et al., 2018), XLNet (Yang et al., 2019)) have become the highlight in the area, where they are used both to learn features for text representation and as classification algorithms. The main problem of such methods is the very high computational costs needed for learning the model parameters (Sun et al., 2019; Cunha et al., 2021).

Ensemble approaches, such as stacking, which combine the outputs of several base classification models to form an integrated output, have also been shown to excel in ATC (Džeroski and Ženko, 2004; Ding and Wu, 2020), enjoying high effectiveness and computational costs that depend on the selected learning methods of the ensemble. They are motivated by the fact that distinct learning models or text representations may complement each other, uncovering specific structures that underlie the input/output relationship of the data. Early works (Larkey and Croft, 1996) aimed at showing combinations of different classification algorithms capable of producing better effectiveness results than any single type of classifier.

However, the benefits of ensemble techniques against a strong classifier are not always clear (Yan-Shi Dong and Ke-Song Han, 2004), in part, due to the excellent generalization power of the best classifiers. In fact, previous ensemble works mostly focus on improving the overall classification effectiveness using the results of traditional classification algorithms (Campos et al., 2017; Ding and Wu, 2020), paying little or no attention to practical issues such as the execution time or which combination of efficient base algorithms can bring effective results at a lower cost.

Accordingly, our *first contribution* in this paper is a thorough study of the cost-effectiveness tradeoff of stacking techniques for text classification tasks. Rather than just evaluating the effectiveness of an ensemble of various recent and effective methods, including those based on transformers and attention models, we focus on the study of stackers capable of achieving a better compromise between low cost (or high efficiency) and high effectiveness when compared to a single base model (i.e., the most effective single classifier in a given dataset). A wide range of comparative experiments with stacked ensemble models and state-of-the-art base algorithms are conducted on six datasets widely used in text classification. We seek answers based on empirical evidence to the following questions, considering the best learning model for each given dataset: *(RQ1): Is it possible to obtain an effective ensemble with significantly less computational time than the best learning model? (RQ2): Is it possible to improve the effectiveness of the best learning model using an ensemble without increasing the computational time? (RQ3): Disregarding the computational time, can an ensemble improve the effectiveness of the best learning model?* As far as we know, we are the first to investigate the cost-effectiveness trade-offs (Cunha et al., 2021) of stacking of neural and non-neural text classifiers from the described perspectives.

A second main contribution of our work is the proposal of a low-cost oracle-based method that can predict the best ensemble in each scenario (with and without computational cost limitations) using only a fraction of the available training data. Our **"Oracle"** first estimates the best base algorithm (which can be seen as a baseline for effectiveness) to perform an efficient greedy search of ensembles guided by both their effectiveness and efficiency concerning the best base algorithm. Particularly, the Oracle predicts effective ensembles by successively including base algorithms that improve their combined majority voting effectiveness. Moreover, our method avoids the inclusion of expensive base algorithms (concerning the best base algorithm) to guarantee the ensemble efficiency. Our proposed Oracle is the first known strategy to provide an efficient prediction of effective ensembles capable of tackling practical efficiency issues related to our research questions. In more details, our proposal aims at predicting three ensembles corresponding to the time restrictions of RQ1, RQ2 and RQ3, respectively, while avoiding the potential high computational cost of evaluating expensive base models and their ensembles, especially on large datasets.

Our experimental results show affirmative answers to our three research questions in most experiments. In most datasets, it is possible to obtain an ensemble of base algorithms that is as good as or better than the base algorithm, at a lower cost. In 5 out of 6 datasets it is possible to obtain an ensemble with statistically significant gains against the best algorithm with no increase in cost. Similarly, in 5 out of 6 datasets, our oracle provides as good as or better results than the best base algorithm with no increase in cost, providing empirical evidence to the practical benefits of the proposed oracle.

2 Background and Related Work

2.1 Text Classification Strategies

Early efforts in ATC focused on improving machine learning algorithms such as Naïve Bayes, kNN, Logistic Regression and SVM (Howard and Ruder, 2020) using a simple bag of (TFIDF weighted) words representation. Even with such simple document representation, the use of methods such as LinearSVM (Fan et al., 2008a) and XGBoost (Chen and Guestrin, 2016a) produced high effectiveness with and efficient convergence for large datasets (Fan et al., 2008a).

More recent strategies, such as meta-features (Canuto et al., 2018) and neural networks (NNs) (Joulin et al., 2017; Tang et al., 2015a), exploit the training data to build more informative document representations. Particularly, strategies based on metafeatures (Canuto et al., 2018; Canuto et al., 2019b) extract information from more basic (bag-of-words) features to enhance the feature space by smartly exploiting a document's neighborhood. Strategies based on NNs enhance word representations (and thus documents) also exploiting the training data. FastText (Joulin et al., 2017) and PTE (Tang et al., 2015a), for instance, presented high effectiveness in comparison to (costly) deep learning approaches.

Considerable advances on deep learning for ATC were achieved by using pre-trained language models with fine-tuning (Howard and Ruder, 2018), mainly when combined with attention mechanisms (Kokkinos and Potamianos, 2017; Yang et al., 2016) and the parallelization benefits of transformers, better exemplified by BERT (Devlin et al., 2018). Following BERT's success, the recent XLNet network (Yang et al., 2019) proposes a new autoregressive formulation to improve the exploitation of contextual information. Though effective, the fine-tuning process of methods such as BERT and XLNet still takes substantial computational time, requiring powerful hardware (GPUs) (Sun et al., 2019). Such requirements might bring practical limitations for these solutions.

2.2 Stacking

Stacking (Wolpert, 1992) is a widely known ensemble technique that combines the predictions of heterogeneous algorithms (i.e., base algorithms) to improve effectiveness concerning these base algorithms. To implement stacking, we first need to train each base algorithm. With the trained models, we can make predictions in a different validation set, which was not used for training. With the saved models and the predictions in the validation set, a *metalayer* (another learning algorithm) is used to learn how to combine the predictions in the combination. Recent work reported high effectiveness with stacking for multiple ATC tasks, such as topic classification (Campos et al., 2017; Abuhaiba and Dawoud, 2017), sentiment analysis (Carvalho and Plastino, 2020; Onan et al., 2016) and multi-label classification (Xia et al., 2020; Weng et al., 2019). Particularly, stacking provided substantial effectiveness improvements on recently proposed decision-tree-based algorithms (Campos et al., 2017) and with methods trained on different representations (including word embeddings) (Carvalho and Plastino, 2020; Pelle et al., 2018; Onan et al., 2016).

A careful choice of base algorithms is necessary due to the potential degradation of the stacking effectiveness and efficiency. The literature reported low effectiveness on stacking due to overfitting issues with multiple base algorithms (Reid and Grudic, 2009; Ledezma et al., 2010). Previous works that optimize the choice of a subset of base algorithms (Ledezma et al., 2010; Gupta and Thakkar, 2014) focused on maximizing the ensemble effectiveness with no concern for efficiency. Stacking efficiency is usually attached to the most expensive method. In fact, (Hou et al., 2021) reportedly avoids the cost of using expensive deep learning methods by including gradient boosting base algorithms comparable to convolutional NNs.

In this work, we provide a thorough evaluation of the effectiveness and efficiency tradeoffs of stacking, i.e., we investigate whether there are combinations of algorithms that overcome (in both, efficiency and effectiveness) the best base ones in a given dataset. Our proposed oracle in turn is the first method to explicitly tackle a time-constrained stacking prediction goal by explicitly and efficiently exploiting the relationships between stacking and the best base algorithms.

3 Methodology

3.1 Time-Constrained Stacking

We aim to answer the following research questions: *(RQ1): Is it possible to obtain an effective ensemble with significantly less computational time than the best learning model? (RQ2): Is it possible to improve the effectiveness of the best learning model using an ensemble without increasing the computational time? (RQ3): Disregarding the computational time, can an ensemble improve the effectiveness of the best learning model?*

With RQ1 we aim to identify whether it is possible to obtain a stacking of (a subset of) base algorithms that is as effective or better than the best (i.e., most effective) base algorithm and takes strictly less computational time than the best base. Favorable evidence towards a positive answer is important to indicate the existence of cost-effective stacking solutions, especially if the best base algorithm is a costly strong/high generalization power baseline. RQ2 keeps the same effectiveness demands of RQ1, but considering the following relaxation on the time constraint: the parallel execution of the base models can take at most the same execution time as the best base algorithm. This time constraint allows the best base algorithm to be included in the stacking. With this, we intend to evaluate if effectiveness improvements are possible with the (time) cost of the best base algorithm as an upper limit. In RQ3 we remove all time constraints to obtain the best possible stacking regardless of cost. With RQ3 we want to evaluate the potential effectiveness improvements of an stacking over the best base algorithm, in exchange for additional (time) cost.

3.2 Oracle-Based Prediction of Stacking Performance

The proposed strategy is implemented as follows: (i) each base algorithm is trained with a reduced amount of the training set (e.g., 30%); (ii) we run an algorithm, called "Oracle" (Algorithm 1), which aims at finding the best combination of base algorithms with less training by a greedy strategy. First, we select the best base algorithm obtained with the reduced training to start the combination, where $\mathbb{A}$ is the set of all base algorithms executed with less training. For this, we use the $Best(\mathbb{A})$ function, which simply returns the best algorithm based on the validation set. For each iteration, the next best algorithm, as estimated in a validation set, is added and we verify whether the combined result presents a statistically significant improvement ($\alpha = 0.05$) in relation to the previous iteration. If positive, it is permanently included in the combination. The process continues until all base algorithms are considered. The strategy is greedy since it makes the best choice in the current iteration.

To perform the comparison and statistical tests in each iteration, we use a separated piece within the training set (validation) that is not contained in the smaller part used in training. Besides, as a meta-layer, we use a simple average, i.e., we simply add the probabilities of the predictions dividing it by the number of base algorithms. The meta-layer average is represented by the function $Avg(\mathbb{E})$ in the pseudocode, where $\mathbb{E} \subset \mathbb{A}$. As it is a simple meta-layer and not a learning algorithm, the cost can be considered insignificant in the choice process.

Algorithm 1 Oracle Algorithm

```
procedure ORACLE($\mathbb{A}$)
    $\mathbb{C} \leftarrow Best(\mathbb{A})$
    $\mathbb{S} \leftarrow \mathbb{A} - \mathbb{C}$
    while $\mathbb{S} \neq \emptyset$ do
        $X \leftarrow Best(\mathbb{S})$
        $\mathbb{E} \leftarrow \mathbb{C} + X$
        if $Avg(\mathbb{E}) > Avg(\mathbb{C})$ then
            $\mathbb{C} \leftarrow \mathbb{E}$
            $\mathbb{S} \leftarrow \mathbb{A} - \mathbb{C}$
        else
            $\mathbb{S} \leftarrow \mathbb{S} - X$
    return $\mathbb{C}$                                ▷ Best combination
```

With the oracle defined, we raise the following research questions: *(ORQ1): Can we predict, using a fraction of the training data, an effective stacking that will tie or outperform the best learning model when trained with all the available training, at a smaller cost than that of the best model? (ORQ2): Can we make a similar prediction than in ORQ1, but now with cost smaller or at the maximum equal to that of the best model when trained with all training data? (ORQ3): with no constraints in time, can we predict a combination that will be better than the best learning algorithm in a dataset?*

4 Experiments

4.1 Experimental Setup

We consider the effectiveness and efficiency of the models on two large-scale ATC datasets (Zhang et al., 2015) (more than 100,000 documents) – AG's News (AGNews) and IMdB Reviews – and four mid-sized datasets very known in the ATC community – 20 Newsgroups (20NG), WebKB (WebKB), Reuters (Reut) and ACM Digital Library (ACM). Table 1 shows the datasets details.

In terms of classification (base) algorithms, we consider the LinearSVM (Fan et al., 2008b), kNN (Altman, 1992), LogisticRegression (Fan et al., 2008b), XGBoost (Chen and Guestrin, 2016b), XLNet (Yang et al., 2019) and BERT (Devlin et al., 2018). In terms of representations, beyond the traditional term-weighting alternatives (TFIDF), we consider distributional and other types of word embeddings, such as FastText (Joulin et al., 2016; Bojanowski et al., 2017) and PTE (Tang et al., 2015b), as well as recent representations based on MetaFeatures that have obtained state-of-the-art (SOTA) effectiveness in some of the experimented datasets (Canuto et al., 2019a, 2018, 2016; Cunha et al., 2020, 2021). Table 2 has a summary of the base algorithms and respective representations used in each of them.

Algorithm	Representation	ID	Algorithm	Representation	ID
LinearSVM	FastText	A	Logistic	FastText	J
	PTE	B	Regression	PTE	K
	TFIDF	C		TFIDF	L
	Metafeatures	D		Metafeatures	M
kNN	FastText	F	XGBoost	FastText	N
	PTE	G		PTE	O
	TFIDF	H		TFIDF	P
	Metafeatures	I		Metafeatures	Q
			XLNet	Raw Documents	R
			BERT	Raw Documents	S

Table 2: Base Algorithms and Representations IDs.

We run the stacking process with the following variants: all combinations of the same base algorithm with different representations, all combinations of different base algorithms with their best representations, and a combination that includes all the base algorithms. For example, we perform all possible combinations of LinearSVM with FastText, PTE, TFIDF and MetaFeatures, resulting in total of 11 combinations: $\binom{4}{2} + \binom{4}{3} + \binom{4}{4}$. This limitation of combinations has a main reason: all combinations of all algorithms and representations, 18 in our case, would result in an impracticable number of possible combinations for execution: 262,125 experiments $= \binom{18}{2} + \binom{18}{3} + .. + \binom{18}{18}$.

An important observation is that we assume that the base algorithms can be run in parallel (e.g. different machines). Thus, a stacking or oracle combination has the execution time limited by the most costly base algorithm in the respective combination. Even if this assumption is not true and it is necessary to execute the base algorithms and combinations on one single machine, this would only

Dataset	Size	# Feat.	Class Distribution					Avg Doc. Size (words)	Skewness
			# Classes	Minor Class	Median	Mean	Major Class		
20NG	18,846	97,401	20	628	984	942	999	296	Balanced
ACM	24,897	48,867	11	63	2,041	2,263	6,562	65	Imbalanced
Reut	13,327	27,302	90	2	29	148	3964	171	Extremely Imbalanced
WebKB	8,199	23,047	7	137	926	1,171	3705	209	Imbalanced
AGNews	127,600	39,837	4	31,900	31,900	31,900	31,900	37	Balanced
IMdB Reviews	348,415	115,831	10	12,836	31,551	34,841	63,233	326	Imbalanced

Table 1: Datasets statistics.

aggravate the cost problem and allow an unfair comparison in our favor. Therefore, to avoid this unfair comparison, we maintain the assumption of parallel execution.

The experiments in the smaller datasets were executed using a 10-fold cross-validation procedure while in the larger we used 5 folds due to the cost of the procedure. The algorithms parameters were tuned using the Bayesian Optimization (Bergstra et al., 2015) approach with 10 iterations, with the 5-fold stratified strategy and the training set (nested cross-validation). In Table 14 in the Appendix, we have the values of each parameter that we optimize in the non-neural base algorithms. The parameters and pre-trained models for BERT and XLNet are also shown in the Appendix (Table 12). For the neural networks, we adopted the same parameters defined by the authors of their respective methods (Devlin et al., 2018; Yang et al., 2019). In our experiments, we adopt AWS EC2 instances to run and measure the execution time for both neural and non-neural algorithms. For the non-neural algorithms, we use the instance model *c5a.12xlarge*, which has 48 CPUs, 96GB of RAM (without GPU). For the neural algorithms, we use the instance model *p2.xlarge*, which has one NVIDIA K80 GPU (12 GB of memory), 4 CPUs and 61 GB of RAM.

We evaluate all methods, combined with different representations, with respect to classification effectiveness and training time. We assess classification effectiveness in the test partitions using MicroF1 and MacroF1 (Sokolova and Lapalme, 2009). While MicroF1 measures the classification effectiveness over all decisions, MacroF1 measures the classification effectiveness for each individual class, averaging them, being very important for skewed datasets. In addition to effectiveness, we also assess the cost of each method in terms of the training execution time aiming at analyzing the cost-effectiveness trade-offs for all methods. The metric is the overall time in seconds (average of folds). To compare the average test results on our cross-validation experiments, we assess the statistical significance employing the paired t-test with 95% confidence, which is strongly recommended over signed-rank tests for hypothesis testing on mean effectiveness and arguably robust to potential violations of the normality assumption in this context (Urbano et al., 2019; Hull, 1993).

4.2 Experimental Results

4.2.1 Stacking Results

Effectiveness and Time results for the base algorithms in each dataset are shown in Table 3. The results of these best base algorithms are considered in the next analyses. Results for RQ1, RQ2 and RQ3, in terms of MacroF1 for each dataset are shown in Tables 4, 5, and 6, respectively, while Figure 1, shows the analysis of the cost (time). For each dataset, the tables show the effectiveness (MacroF1) of the best base algorithm along with the stacking combination that best answered the respective research question (if any), the respective combination of methods (the letters refer to the index of algorithms described in Table 2), and finally, in the last column, (Most Costly) the most costly algorithm that entered in the combination, according to the constraints imposed by the question. We present only MacroF1 results due to space constraints and the fact that is it harder to improve them in the highly skewed scenario that occurs in most of the experimented datasets. However, we also consider MicroF1, whose results are summarized in Table 7.

In Table 4, which focuses on RQ1 that has a strong constraint in terms of cost (time), we can see that in 4 out of 6 datasets, it is possible to obtain a combination of classifiers (stacking) that is

Dataset	Metric	A	B	C	D	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
		Algorithm																	
20NG	MicroF1	80.4	88.66	89.46	90.52	80.96	81.25	84.63	90.42	81.54	88.96	88.89	**90.73**	80.39	86.24	79.86	90.06	74.38	85.88
	MacroF1	79.96	88.34	89.2	90.32	80.45	80.46	84.06	90.27	81.09	88.69	88.65	**90.58**	79.94	85.9	79.35	89.9	73.97	85.43
	Time (S)	79	66	22	702	76	76	28	78	323	413	163	**467**	632	1,392	1,428	1,163	3,972	3,959
ACM	MicroF1	73.28	75.34	76.62	79.01	73.88	75.73	73.64	78.19	73.90	75.76	77.21	79.19	73.49	75.98	74.67	**78.98**	71.65	78.25
	MacroF1	60.64	62.57	66.88	68.66	62.47	61.37	59.25	65.22	61.95	63.19	68.43	69.11	60.93	61.82	64.12	**69.32**	60.58	65.56
	Time (S)	89	112	13	776	136	140	35	125	481	2057	201	922	512	1,239	530	**1,542**	4,153	3,482
Reut	MicroF1	66.57	66.07	67.19	78.85	68.53	68.96	69.41	75.19	65.64	66.55	66.80	76.99	64.68	63.70	65.92	**82.17**	72.72	72.72
	MacroF1	30.13	31.75	34.54	42.48	31.45	26.31	32.74	35.68	30.45	31.22	33.88	43.17	28.73	23.91	31.85	**47.37**	40.29	33.60
	Time (S)	120	150	38	3,427	39	38	12	163	4,434	4,316	2,026	15,185	2,363	2,633	1,482	**6,426**	3,537	3,897
WebKB	MicroF1	75.69	71.31	81.35	77.4	77.16	72.53	75.02	77.27	76.64	71.41	82.64	77.62	75.63	78.58	83.65	79.47	**84.60**	86.04
	MacroF1	65.81	58.19	71.45	65.29	69.17	59.25	58.76	64.72	68.25	58.66	74.67	66.19	65.54	66.21	74.33	69.23	**77.76**	66.41
	Time (S)	37	43	11	106	19	18	9	23	50	224	41	175	97	229	197	242	**2,210**	1,735
AGNews	MicroF1	89.52	91.95	91.08	91.03	89.32	91.25	90.35	91.55	89.23	91.66	91.87	91.29	89.17	92.23	91.97	91.69	91.99	**93.74**
	MacroF1	89.50	91.93	91.06	90.99	89.30	91.25	90.32	91.53	89.23	91.65	91.85	91.27	89.16	92.22	91.96	91.68	91.97	**93.74**
	Time (S)	2,308	4,198	115	1,099	10,433	12,707	2,097	4,373	8,434	881	2,180	3,364	5,243	7,019	1,151	2,769	33,206	**11,257**
IMdB Reviews	MicroF1	29.69	37.53	30.51	20.31	32.88	32.99	27.01	28.00	29.17	38.64	33.75	29.02	29.76	37.03	34.58	30.02	25.08	**39.00**
	MacroF1	24.96	27.43	26.10	12.00	28.12	25.39	19.50	19.62	26.55	32.35	28.19	20.31	26.39	30.72	27.96	20.93	16.26	**34.09**
	Time (S)	50,817	23,658	1,977	26,885	43,005	42,525	13,476	68,117	72,658	4,818	35,155	15,988	35,543	123,705	59,634	91,562	59,749	**34,426**

Table 3: Effectiveness, based on MicroF1 and MacroF1, and efficiency, based on execution time, for all classification algorithms and datasets considered in the our experiments. The algorithms are: (A) LinearSVM + FastText; (B) LinearSVM + PTE; (C) LinearSVM + TFIDF; (D) LinearSVM + TFIDF+Mf; (F) KNN + FastTex; (G) KNN + PTE; (H) KNN + TFIDF; (I) KNN + TFIDF+Mf; (J) LogisticRegression + FastTex; (K) LogisticRegression + PTE; (L) LogisticRegression + TFIDF; (M) LogisticRegression + TFIDF+Mf; (N) XGBoost + FastTex; (O) XGBoost + PTE; (P) XGBoost + TFIDF; (Q) XGBoost + TFIDF+Mf; (R) XLNet + Raw Documents; and (S) BERT + Raw Documents. The best results are highlighted in bold. When we identified statistically ties between two or more algorithms, we chose the one with lower execution time. For example, for the 20NG, we identified statistical ties for the algorithms M (LogisticRegression + TFIDF+Mf) and Q (XGBoost + TFIDF+Mf) and choose the M that presents lower execution time.

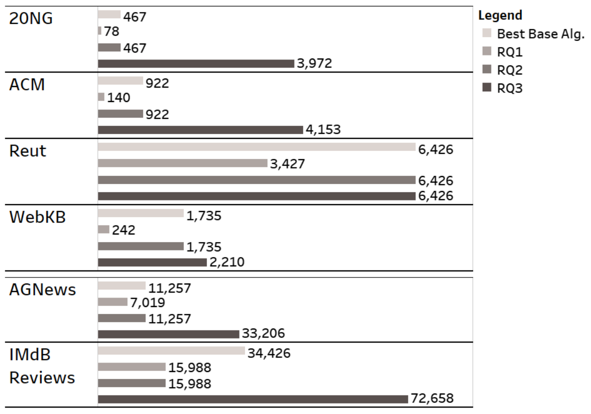

Figure 1: Stacking Times

Dataset	Experiment	MacroF1	Combination	Most Costly
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	RQ1	● 91.02	FGHI	I
ACM	Best Base	69.32	Q (XGboost + MetaFeat)	
	RQ1	▲ 71.50	JLM	M
Reut	Best Base	47.37	Q (XGboost + MetaFeat)	
	RQ1	● 46.98	KL	K
WebKB	Best Base	77.76	R (XLNet)	
	RQ1	● 79.61	DIS	S
AGNews	Best Base	93.74	S (BERT)	
	RQ1	▼ 93.37	NOPQ	O
IMdB Reviews	Best Base	34.09	S (BERT)	
	RQ1	▼ 33.18	KM	M

Table 4: RQ1.

as good as (statistical tie) or better (see the *ACM* case, with statistically significant gains of 3.1%) than the best base algorithm, at a lower cost. In fact, the gains in terms of cost (time) are very significant (see Figure 1), ranging from 1.87x speedup improvement (in Reuters) to 7.16x (in WebKB)[1]. Even if we consider the two cases in which there were some minimum effectiveness losses (0.39% in AGNews and 2.66% in IMdB), there are some significant speedups: 1.6x in AGNews and 2.15x in IMdB. Some chosen stacked combinations are interesting, such as FGHI in 20NG that contains all versions of kNN, and JLM in ACM, containing three versions of Logistic Regression. Both combinations contains classifiers with Metafeatures.

Results for RQ2 (Table 5) are also very interesting. In 5 out of 6 datasets it is possible to obtain effectiveness gains with no increase in cost (remind that in this scenario the cost is limited by that of the best base algorithm). Effectiveness gains vary from 0.4% in AGNews, 1.15% in 20NG[2], 3.1% in ACM, 5.4% in IMdB and 9% in WebKB. Reuters is only considered a tie because of the high variability of the results across folds in this dataset (due to the large number of classes and very high skewness), which generates large standard deviations/confidence intervals. In absolute terms, there was a positive variation (non-statistically significant gain) of more than 9.7%. Indeed, the MicroF1 stacking results confirms statistically significant gains in Reuters (See Table 7).

As expected, to obtain gains in this scenario it is necessary to include the best base algorithm in the combination in most datasets, inserting diversity/complementarity into the combination. Only

[1]Speedups in 20NG and ACM were 6x and 6.6x

[2]Notice that improvements in 20NG and AGNews are very hard to obtain given the already high effectiveness values.

Dataset	Experiment	MacroF1	Combination	Most Costly
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	RQ2	▲ 91.63	JKLM	M
ACM	Best Base	69.32	Q (XGBoost + MetaFeat)	
	RQ2	▲ 71.50	JLM	M
Reut	Best Base	47.37	Q (XGBoost + MetaFeat)	
	RQ2	● 51.99	OPQ	Q
WebKB	Best Base	77.76	R (XLNet)	
	RQ2	▲ 84.07	All	R
AGNews	Best Base	93.74	S (BERT)	
	RQ2	▲ 94.12	DIQS	S
IMdB	Best Base	34.09	S (BERT)	
Reviews	RQ2	▲ 35.95	MS	S

Table 5: RQ2.

in ACM, the base algorithm is not part of the combination. Notice also that, due to the time constraints, the gains are somewhat limited due to the restricted number of classifiers that can be combined. This has some impacts on the results, for instance, in IMdB only two algorithms belong to the best combination while the combination in ACM has only three classifiers. Only in WebKB the combination includes all 18 classifiers as the base algorithm is also the most expensive one. Another interesting aspect of the combinations is that in all datasets, a classifier using Metafeatures was included (e.g., M and Q)

Finally, in the scenario with no time constraint (RQ3), further gains can be obtained with the inclusion of more costly classifiers. There are further gains in AGNews (0.94%), 20NG (2.06%), IMdB (5.8%) and ACM (6.32%). Notice that in this scenario, there is a tendency to include most algorithms in the combinations, like in ACM, WebKB and AGNews, to obtain further improvements. This means that most algorithms have complementary information that tends to contribute to the final results. Another interesting aspect to notice is that in some cases, such as in 20NG, a completely different combination than that chosen in scenario RQ2, was picked. This combination exploits the most effective and complementary algorithms, and may not even include the base classifier. In other cases, such as in IMdB, a combination of a few of the most effective (and costly) algorithms suffices to obtain larger gains. This means that the meta-layer is really doing a good job in learning about the individual performance of the algorithms and their complementarity. Finally, these additional effectiveness gains come with potential high increases in cost, clearly seen in Figure 1 for the cases of 20NG, ACM and AGNews. In those datasets, the costs have tripled (AGNews), quadrupled (ACM and IMdB) or become 8x more cost expensive. It is up to the application designer to decide whether this cost-effectiveness tradeoff is worth it.

Dataset	Experiment	MacroF1	Combination	Most Costly
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	RQ3	▲ 92.45	DIRS	R
ACM	Best Base	69.32	Q (XGBoost + MetaFeat)	
	RQ3	▲ 73.70	All	R
Reut	Best Base	47.37	Q (XGBoost + MetaFeat)	
	RQ3	● 51.99	OPQ	Q
WebKB	Best Base	77.76	R (XLNet)	
	RQ3	▲ 84.07	All	R
AGNews	Best Base	93.74	S (BERT)	
	RQ3	▲ 94.63	All	R
IMdB	Best Base	34.09	S (BERT)	
Reviews	RQ3	▲ 36.06	QS	Q

Table 6: RQ3.

Table 7 summarizes the effectiveness results. For RQ1, there are 8 win/ties out of 12 possibilities (6 datasets, two metrics). Remind that in this scenario ties are considered a good result due to the reduction in costs. For RQ2 and RQ3, there are 11 wins, only 1 tie (in Macro in Reuters) no losses at all. In terms of cost (Figure 1), significant reductions in scenario 1 (RQ1) can be obtained in all 6 datasets, with almost no loss (or minimal losses) in terms of effectiveness. For scenario 2 (RQ2), effectiveness gains can be obtained in almost all cases with no additional cost when compared to the cost of the base classifier. And for scenario 3 (RQ3) additional effectiveness gains can be obtained, but sometimes with a very high increase in cost.

RQ	MicroF1			MacroF1		
	Win	Tie	Loss	Win	Tie	Loss
RQ1	2	2	2	1	3	2
RQ2	6	0	0	5	1	0
RQ3	6	0	0	5	1	0

Table 7: Win/Tie/Loss Summary.

4.2.2 Oracle Results

MacroF1 results of the Greedy Oracle predictor are shown in Tables 8, 9 and 10. These results correspond to an Oracle that uses the results of the base algorithms trained with 30% of the training data and predicting in a different training data portion in a nested folded cross-validation procedure.

We start by answering ORQ1. Table 8 shows that in half of the cases we can perform a good prediction, i.e., one that predicts a combination of methods that will tie or outperform the best base algorithm when trained with all the available training data (100%). It is very important to stress that in a real situation we do not really know what will be the best algorithm when using all the training data nor its effectiveness. Indeed, with more data, there is a tendency for some algorithms, such as the transformers, to improve their effectiveness, but their good performance may not be predicted with few training data. Remind also that this is a very strict scenario: even if we can predict which will be the best base algorithm, we cannot use it in the combination given the time constraints of ORQ1.

Dataset	Experiment	MacroF1	Combination	Most Cost
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	ORQ1	▲ 91.62	ABCDIL	I
ACM	Best Base	69.32	Q (XGBoost + MetaFeat)	
	ORQ1	▲ 71.85	ABCDIJKLM	D
Reut	Best Base	47.37	Q (XGBoost + MetaFeat)	
	ORQ1	▼ 32.55	ABDQR	Q
WebKB	Best Base	77.76	R (XLNet)	
	ORQ1	● 75.79	BCF	F
AGNews	Best Base	93.74	S (BERT)	
	ORQ1	▼ 93.16	BDGKOQ	B
IMdB	Best Base	34.09	S (BERT)	
Reviews	ORQ1	▼ 27.79	L	L

Table 8: Oracle ORQ1.

Given all these limitations, mainly that only the algorithms with a cost lower than the best base algorithm (with 30% of training) can be considered, it is impressive that we can make a prediction that will surpass in effectiveness the best base algorithms using 100% of training in 20NG and ACM and tie with it, being cheaper, in WebKB. But even in the case in which there were losses, some were minimal, like in AGNews with a loss of only 2.5% with potential gains in training time. Only in Reuters and IMdB there were significant MacroF1 losses, mainly due to the failure of predicting which would be the best base algorithm[3] and the impossibility of including the predicted best base algorithm in the combination.

When we are allowed to include the best-predicted algorithm in the stacking (scenario for ORQ2) results are even better – we can make a good prediction in 5 out 6 cases (2 wins and 3 ties). Notice that in this scenario we consider a tie as a good result. We interpret that being able to predict a combination that will tie with the best algorithm with 100% of training in a dataset, without knowing which one will this best, at a very lost cost

[3]In case of Reuters, a XGBoost with Metafeatures (Q) and in IMdB, BERT (S).

Dataset	Experiment	MacroF1	Combination	Most Cost
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	ORQ2	▲ 91.52	ABCDILM	M
ACM	Best Base	69.32	Q (XGBoost + MetaFeat)	
	ORQ2	▲ 71.56	ABDMQ	Q
Reut	Best Base	47.37	Q (XGBoost + MetaFeat)	
	ORQ2	● 47.37	DMQ	M
WebKB	Best Base	77.76	R (XNet)	
	ORQ2	● 78.22	BCFGIJ	J
AGNews	Best Base	93.74	S (BERT)	
	ORQ2	● 94.10	BCDGIKLMOPQR	R
IMdB	Best Base	34.09	S (BERT)	
Reviews	ORQ2	▼ 31.11	K	K

Table 9: Oracle ORQ2.

(Figure 2), as an excellent result. Notice that the best results in this scenario (i.e., 20NG and ACM) are obtained when we can in fact predict what will be the best base algorithm with 100% of training. But even when we cannot predict, as in the case of WebKB and AGNews[4], we can find a combination of simpler (and potentially less expensive) algorithms that can tie with the best. Again, IMdB was the only case in which we could not make a good prediction exactly by the failure in predicting, with 30% of training, that BERT would be the best algorithm when all the training data is used.

Finally, when no time constraints are imposed the oracle's prediction results are excellent: 4 wins, 1 tie and only one loss (in IMdB). This last loss is explained by the same reasons as in the previous scenario: the failure of predicting BERT as the future best algorithm. But even in this case, the prediction for using algorithm K: LogisticRegression with PTE as the sole combination (an unusual prediction) produced minimal losses: only 1.05% at a cost much smaller than using BERT. And in the case of Reuters, we obtain an absolute increase in MacroF1 values (6% increase), though not statistically significant due to the high variability.

When looking at the costs of making the predictions in each scenario (ORQ1, ORQ2, and ORQ3), shown in Figure 2, we can see that in all cases (but 20NG for ORQ3), the oracle's predictions times are much smaller, in many cases negligible[5], when compared to the time to run the base algorithm with 100% of training. Given the time constraints imposed by ORQ1 and ORQ2 and the fact even in the scenario for ORQ3, only a portion of the 18 available algorithms needed to be stacked (in most cases) to produce effectiveness gains, the advan-

[4]In both the best base algorithms with 100% were the neural transformers XLNet and BERT

[5]Some differences are in the orders of magnitude.

Dataset	Experiment	MacroF1	Combination	Most Cost
20NG	Best Base	90.58	M (Log. Reg. + MetaFeat)	
	ORQ3	▲ 92.17	ABCDILMS	M
ACM	Best Base	69.32	Q (XGBoost + MetaFeat)	
	ORQ3	▲ 73.48	ABDMQ	Q
Reut	Best Base	47.37	Q (XGBoost + MetaFeat)	
	ORQ3	● 50.18	DMQ	M
WebKB	Best Base	77.76	R (XLNet)	
	ORQ3	▲ 83.12	ABCFJLMNOPQR	J
AGNews	Best Base	93.74	S (BERT)	
	ORQ3	▲ 94.63	All	R
IMdB Reviews	Best Base	34.09	S (BERT)	
	ORQ3	▼ 33.73	K	K

Table 10: Oracle ORQ3.

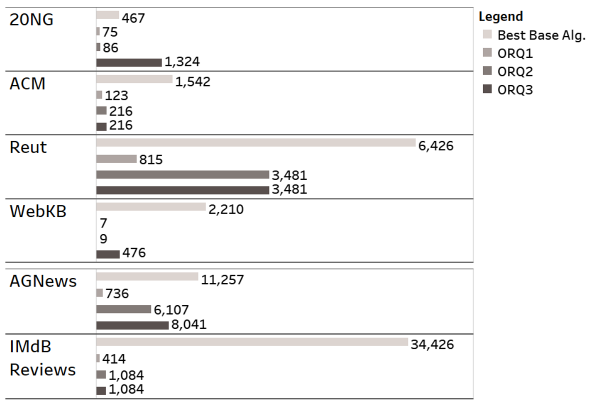

Figure 2: Oracle Times

tage's of running the oracle's predictions in terms of cost stand for themselves.

Table 11 summarizes the results in terms of Micro and MacroF1: considering all 36 results (three RQs, 6 datasets, 2 metrics) the oracle predicted 17 wins, 10 ties (most of them (8) in scenarios ORQ1 and ORQ2, which can be considered good results) and only 9 losses, six of them in a single dataset (IMdB) for the simple reason that we failed in predicting a neural network winner with fewer data. This is certainly a point to be improved in our methodology. One idea is to look not only at the absolute effectiveness values with a single training point (30%) but look also at the tendency of growing considering several points (5%, 10%, ..).

RQ	MicroF1			MacroF1		
	Win	Tie	Loss	Win	Tie	Loss
ORQ1	2	0	4	2	1	3
ORQ2	2	4	0	2	3	1
ORQ3	5	1	0	5	1	0

Table 11: Oracle Win/Tie/Loss Summary

5 Conclusion and Future Work

We presented two important contributions to the application of Stacking in ATC: a thorough study of cost-effectiveness trade-offs and the proposal of a new oracle method to predict the best ensemble combination for a dataset at a low cost. Our extensive experiments, composed of 4 textual representation methods, 6 datasets, 4 non-neural based algorithms and 2 neural-based algorithms, provided us with answers to questions that had not yet been explored in the literature. By performing stacking with different time constraints, we showed that it was possible to obtain combinations that positively answered the posed questions regarding the time-constrained stacking and the oracle predictions in terms of both, effectiveness and efficiency.

We highlight general and practical guidelines based on our extensive experiments. First, we notice the consistent appearance of recent meta-features on the best combinations of base learners obtained for each evaluated research question (Tables 4–6). In fact, due to the focus of meta-features on summarizing relevant distance-based information from the original features, we strongly suggest their exploitation in ensemble combinations. Moreover, the largest datasets benefit from additional data to fine tune BERT for the classification task. Therefore, combinations including both of these recent and distinct paradigms (meta-features and BERT) for stacking were able to produce very effective results on most datasets (as shown in Table 10). We suggest that stacking methods should start by exploiting these two paradigms in conjunction. Finally, our experiments show the need of specific stacking solutions for different scenarios/datasets. The application of our proposed Oracle efficiently predicts effective best base models on time-constrained scenarios, allowing adaptable solutions that automatically optimize the choice of base learners for each specific dataset. We suggest to exploit the Oracle in all these situations.

In the future, we will explore different Oracle configurations, explore multi-objective feature selection in the stacking meta-layer (Viegas et al., 2018), study other types of constraints (e.g., labeling effort) and apply the Oracle in fields such as recommender systems.

Acknowledgments

This work was supported by CNPq, CAPES, FAPEMIG, AWS, NVIDIA and Google (Awards).

References

Ibrahim S. I. Abuhaiba and Hassan M. Dawoud. 2017. Combining different approaches to improve arabic text documents classification. *International Journal of Intelligent Systems and Applications(IJISA)*, 9(4):39 – 52.

Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. 2015. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Raphael Campos, Sérgio Canuto, Thiago Salles, Clebson C.A. de Sá, and Marcos André Gonçalves. 2017. Stacking bagged and boosted forests for effective automated classification. In *Proceedings of the 40th ACM SIGIR Conference on Research and Development in Information Retrieval*, page 105–114.

S. Canuto, D. X. Sousa, M. A. Gonçalves, and T. C. Rosa. 2018. A thorough evaluation of distance-based meta-features for automated text classification. *IEEE Transactions on Knowledge and Data Eng. (TKDE)*, 30(12):2242–2256.

Sérgio Canuto, Marcos André Gonçalves, and Fabrício Benevenuto. 2016. Exploiting new sentiment-based meta-level features for effective sentiment analysis. In *Proc. of the ninth ACM international conference on web search and data mining*, pages 53–62.

Sergio Canuto, Thiago Salles, Thierson C Rosa, and Marcos A Gonçalves. 2019a. Similarity-based synthetic document representations for meta-feature generation in text classification. In *Proc. of the 42nd ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 355–364.

Sergio Canuto, Thiago Salles, Thierson C. Rosa, and Marcos A. Gonçalves. 2019b. Similarity-based synthetic document representations for meta-feature generation in text classification. In *Proc. of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 355–364.

Sergio Canuto, Daniel Xavier Sousa, Marcos Andre Goncalves, and Thierson Couto Rosa. 2018. A thorough evaluation of distance-based meta-features for automated text classification. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2242–2256.

Jonnathan Carvalho and Alexandre Plastino. 2020. On the evaluation and combination of state-of-the-art features in twitter sentiment analysis. *Artificial Intelligence Review*.

Tianqi Chen and Carlos Guestrin. 2016a. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794.

Tianqi Chen and Carlos Guestrin. 2016b. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Washington Cunha, Sérgio D. Canuto, Felipe Viegas, Thiago Salles, Christian Gomes, Vítor Mangaravite, Elaine Resende, Thierson Rosa, Marcos André Gonçalves, and Leonardo C. da Rocha. 2020. Extended pre-processing pipeline for text classification: On the role of meta-feature representations, sparsification and selective sampling. *Inf. Process. Manag.*, 57(4):102263.

Washington Cunha, Vítor Mangaravite, Christian Gomes, Sérgio D. Canuto, Elaine Resende, Cecilia Nascimento, Felipe Viegas, Celso França, Wellington Santos Martins, Jussara M. Almeida, Thierson Rosa, Leonardo C. da Rocha, and Marcos André Gonçalves. 2021. On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study. *Inf. Process. Manag.*, 58(3):102481.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Weimin Ding and Shengli Wu. 2020. A cross-entropy based stacking method in ensemble learning. *Journal of Intelligent & Fuzzy Systems*, pages 1–12.

Saso Džeroski and Bernard Ženko. 2004. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008a. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(61):1871–1874.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008b. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871–1874.

A. Gupta and A. R. Thakkar. 2014. Optimization of stacking ensemble configuration based on various metahueristic algorithms. In *2014 IEEE International Advance Computing Conference (IACC)*, pages 444–451.

Zhihao Hou, Kun Ma, Yufeng Wang, Jia Yu, Ke Ji, Zhenxiang Chen, and Ajith Abraham. 2021. Attention-based learning of self-media data for marketing intention detection. *Eng. Appl. Artif. Intell.*, 98:104118.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Jeremy Howard and Sebastian Ruder. 2020. A survey on text classification: From shallow to deep learning. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 31.

David Hull. 1993. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–338.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 427–431.

Filippos Kokkinos and Alexandros Potamianos. 2017. Structural attention neural networks for improved sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 586–591.

Leah S. Larkey and W. Bruce Croft. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, page 289–297.

Agapito Ledezma, Ricardo Aler, Araceli Sanchis, and Daniel Borrajo. 2010. Ga-stacking: Evolutionary stacked generalization. *Intelligent Data Analysis*, 14:89–119. 1.

Aytug Onan, Serdar Korukoglu, and Hasan Bulut. 2016. Lda-based topic modelling in text sentiment classification: An empirical analysis. *Int. J. Comput. Linguistics Appl.*, 7(1):101–119.

Rogers Pelle, Cleber Alcântara, and Viviane P. Moreira. 2018. A classifier ensemble for offensive text detection. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, page 237–243.

Sam Reid and Greg Grudic. 2009. Regularized linear models in stacked generalization. In *Multiple Classifier Systems*, pages 112–121, Berlin, Heidelberg. Springer Berlin Heidelberg.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015a. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1165–1174, New York, NY, USA. Association for Computing Machinery.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015b. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1165–1174.

Julián Urbano, Harlley Lima, and Alan Hanjalic. 2019. Statistical significance testing in information retrieval: an empirical analysis of type i, type ii and type iii errors. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514.

Felipe Viegas, Leonardo C. da Rocha, Marcos André Gonçalves, Fernando Mourão, Giovanni Sá, Thiago Salles, Guilherme Andrade, and Isac Sandin. 2018. A genetic programming approach for feature selection in highly dimensional skewed data. *Neurocomputing*, 273:554–569.

W. Weng, C. Chen, S. Wu, Y. Li, and J. Wen. 2019. An efficient stacking model of multi-label classification based on pareto optimum. *IEEE Access*, 7:127427–127437.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241 – 259.

Yuelong Xia, Ke Chen, and Yun Yang. 2020. Multi-label classification with weighted classifier selection and stacked ensemble. *Information Sciences*.

Yan-Shi Dong and Ke-Song Han. 2004. A comparison of several ensemble methods for text categorization. In *IEEE International Conference onServices Computing, 2004. (SCC 2004). Proceedings. 2004*, pages 419–422.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626.

Appendix

Algorithm	Parameters	Value
XLNet	Pretrained Model	XLNet-Base
	batch_size	32
	epochs	5
	max_len	64
	learning_rate	5e-5
	max_grad_norm	1.0
	weight_decay_rate	0.01
BERT	Pretrained Model	BERT-Base
	batch_size	32
	patience	5
	max_len	150
	initial_learning_rate	5e-5

Table 12: Neural networks parameters and pretrained models.

Method	Parameters	Value
TFIDF	Normalization	L2
	Stopwords	NLTK, English
	Max Features	Small Datasets: ∞
		Large Datasets: 50k
	MinDF	2
	Sublinear	TF
PTE	Window	5
	MinDF	2
	Dimensions	300
FastText	Window	5
	Epochs	500
	Model	Skipgram
	Dimensions	300
MetaFeatures	k	[10, 15, 20, 30, 35, 40, 45, 50]

Table 13: Text representations.

Algorithm	Parameters Tunned	Range Values
Linear SVM	C	uniform(0, 20)
	penalty	[l1, l2]
kNN	n_neighbors	range(1, 100, 1)
	metrics	[cosine, l1, l2, minkowski, euclidean]
	weights	[uniform, distance]
Logistic Regression	C	uniform(0, 20)
	penalty	[l2, None]
	solver	[newton-cg, lbfgs, sag, saga]
	class_weight	[None, balanced]
XGBoost	n_estimators	range(100, 1000, 50)
	learning_rate	quniform(0.01, 0.5, 0.01)
	eta	quniform(0.025, 0.5, 0.025)
	max_depth	range(1, 14, 1)
	min_child_weight	quniform(1, 6, 1)
	subsample	quniform(0.5, 1.0, 0.05)
	gamma	quniform(0.0, 1.0, 0.05)
	colsample_bytree	quniform(0.5, 1.0, 0.05)

Table 14: Algorithms and parameters. The implementations of LinearSVM, kNN and LogisticRegression are from scikit-learn and XGBoost is from the respective authors implementation-based package. Omitted parameters are the libraries default. This table has the **range** functions and the **uniform** and **quniform** distributions functions, which are used to define the search space of some algorithms. The range(low, high, step) function returns a number between [**low**, **high**) in a **step** interval. The uniform(low, high) function returns a value uniformly between **low** and **high**. The quniform(low, high, q) function returns a value like round(uniform (low, high) / q) * q and differs from the uniform by a smooth factor.