FEDERAL UNIVERSITY OF MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Christian Reis Fagundes Gomes

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

Belo Horizonte
2021

Christian Reis Fagundes Gomes

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

Versão final

Dissertação apresentada ao Programa de Pós-Graduação em Computer Science da Federal University of Minas Gerais, como requisito parcial à obtenção do título de Mestre em Computer Science.

Orientador: Marcos André Gonçalves
Coorientador: Leonardo Chaves Dutra da Rocha

Belo Horizonte
2021

Christian Reis Fagundes Gomes

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

Final version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Marcos André Gonçalves
Co-Advisor: Leonardo Chaves Dutra da Rocha

Belo Horizonte
2021

Gomes, Christian Reis Fagundes

G633e On the Cost-Effectiveness of Stacking of Neural and Non-Neural Methods for Text Classification: Scenarios and Performance Prediction [manuscrito] / Christian Reis Fagundes Gomes. — 2021.
xiv, 60 f.

Orientador(a): Marcos André Gonçalves
Dissertação (mestrado) — Federal University of Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação
Referências: f. 55-60.;

1. Computação — Teses.

CDU 0

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

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

CHRISTIAN REIS FAGUNDES GOMES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. MARCOS ANDRÉ GONÇALVES - Orientador
Departamento de Ciência da Computação - UFMG

PROF. LEONARDO CHAVES DUTRA DA ROCHA - Coorientador
Departamento de Ciência da Computação - UFSJ

PROF. RODRYGO LUIS TEODORO SANTOS
Departamento de Ciência da Computação - UFMG

PROF. ALEXANDRE PLASTINO DE CARVALHO
Instituto de Computação - UFF

Belo Horizonte, 6 de Outubro de 2021.

I'd like to dedicate my work to everyone who helped me along this journey: my girlfriend, my family, my advisor/co-advisor and my friends.

Acknowledgments

First and foremost, I would like to thank my girlfriend who help me in all my journey with love, support and affection, being the most important person to this work and my life.

To my parents, Andrea and Carlos, and sister, Carol, who are the pillar of support and affection for me to seek and achieve all my goals in life.

To my advisor and co-advisor, Marcos and Leonardo, for their knowledge and support, which were extremely important for this work.

Also I would like to thank my friends for their support. Without that support I couldn't have succeeded in completing this work.

At last but not in least, I would like to thank everyone who helped and motivated me in this work.

"Human felicity is produced not so much by great pieces of good fortune that seldom happen, as by little advantages that occur every day."

(Benjamin Franklin)

Resumo

Algoritmos de redes neurais, como aqueles baseados em *transformers* e modelos de atenção, têm se destacado em tarefas de Classificação Automática de Texto (ATC). No entanto, essa melhora de desempenho tem altos custos computacionais. Conjuntos de classificadores mais simples (ou seja, *Stacking*) que exploram complementaridades dos algoritmos e representações textuais também mostraram produzir desempenho de alto nível em ATC, desfrutando de alta eficácia e custos computacionais potencialmente mais baixos.

Nesse contexto, apresentamos o primeiro e maior estudo comparativo para explorar a relação custo-benefício do *stacking* de classificadores ATC, composto por *transformers* e algoritmos que não utilizam redes neurais. Em particular, estamos interessados em responder a perguntas de pesquisa tais como: (1) É possível obter uma combinação de classificadores eficaz com custo computacional significativamente menor do que o melhor modelo de aprendizado para um determinado conjunto de dados? (2) Desconsiderando o custo computacional, existe uma combinação de classificadores que pode melhorar a eficácia do melhor modelo de aprendizagem? Além de responder a tais questões, outra contribuição principal dessa dissertação é a proposta de um método baseado em oráculos de baixo custo que pode prever o melhor ensemble em cada cenário (com e sem limitações de custo computacional) usando apenas uma fração dos dados de treinamento disponíveis.

Palavras-chave: Processamento de Linguagem Natural, Classificação Automática de Texto, Aprendizado Ensemble, Stacking Blending.

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 dissertation 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.

Keywords: Natural Language Preprocessing, Automatic Text Classification, Ensemble Learning, Stacking Blending.

List of Figures

List of Tables

TABLE OF CONTENTS

Chapter 1

Introduction

Natural Language Processing (NLP), Machine Learning and Data Mining all work together to automate the Automatic Text Classification (ATC) task. ATC is one of the most fundamental tasks involving texts. It aims to automatically associate documents with one or more pre-defined categories, thus providing the organization of information to allow a better understanding and interpretation of textual data. Often the ATC process can be decomposed into four main steps sequentially: Text Pre-Processing (e.g., tokenization, removal of stop words), Attribute Extraction (e.g., TF, TFIDF, Word Embeddings), Dimensionality Reduction (e.g., PCA, LSI) and Application of Classification (Machine Learning) Techniques (e.g., Naive Bayes, SVM, Neural Networks) [Kadhim, 2019; Dalal and Zaveri, 2011] to create a model.

Currently, algorithms based on Neural Networks (e.g., BERT [Devlin et al., 2018], XLNet [Yang et al., 2019]) have become prominent in the ATC area, where they are used both to learn textual representations and as classification algorithms. Undoubtedly, algorithms exploiting Deep Learning have proven to achieve significant gains in several areas of NLP. However, such algorithms have some disadvantages, such as: (i) the need for lots of training data to obtain good performance and (ii) the high computational cost for execution (i.e., the time needed for training the neural network) [Sun et al., 2019; Cunha et al., 2021].

Ensemble methods are strategies capable of combining multiple models to improve the prediction generalization in a given task. Such methods have shown promise in the area of ATC [Džeroski and Ženko, 2004; Ding and Wu, 2020]. Among the possible ensemble strategies, Stacking has the characteristic of using a meta-model (i.e., meta-layer) capable of combining the prediction outputs of different heterogeneous individual models. The basic premise of Stacking is that different learning models, or different textual representations, can complement each other. The meta-model can reveal information intrinsic to the data by combining different models, potentially improving the results (i.e., effectiveness) of a given task. [Rokach, 2009; Wolpert, 1992]. Early work on Stacking in ATC [Larkey and Croft, 1996] aimed to show combinations of different classification algorithms capable of producing better effectiveness results than any individual classifier.

Despite the advantages of using Stacking in ATC, the benefits of ensemble

techniques over a single strong classifier (i.e., a classifier with high generalization) are not always clear [Dong and Han, 2004]. In fact, previous work on Stacking in ATC has mainly focused on improving the overall 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 cost or which combination of efficient algorithms can bring effective results at a lower cost.

1.1 Objectives

Thus, in this dissertation, the main objective is to investigate the cost-effectiveness tradeoff that has been vastly ignored up to today in the literature on Stacking techniques in ATC. To this goal, an extensive set of experiments involving supervised text classification algorithms considered state-of-the-art in the field (neural and traditional network methods) is carried out to evaluate the cost-effectiveness tradeoffs. In addition, we propose a method based on a greedy strategy capable of identifying in short time (i.e., without having to train a classifier with all available training data) the Stacking combinations that potentially will produce the best effectiveness results. The proposed methods – **called Oracle** – manages to produce highly effective Stacking combinations using a fraction of the training data.

1.2 Contributions

The **first contribution of this dissertation** is a thorough study of the cost-effectiveness of Stacking techniques for text classification tasks. Instead of just evaluating the effectiveness of a combination of several current and effective methods, including those based on transformers models, we carry out a study of Stacking combinations capable of achieving a better compromise between low cost (i.e., high efficiency) and high effectiveness when compared to a single individual model (i.e., the single most effective classifier in a given dataset). We conduct a wide range of comparative experiments with combinations of Stacking and classification algorithms considered state-of-the-art in 8 datasets widely used in ATC. This dissertation seeks answers based on empirical evidence for the following Research Questions (RQ), considering the best learning model for each dataset:

- RQ1: Is it possible to obtain an effective ensemble with less computational time than the best individual learning model?
- RQ2: Is it possible to improve the effectiveness of the best learning model using an ensemble without increasing computational time?
- RQ3: Disregarding computational time, is there an ensemble that can improve effectiveness when compared to the best learning model?

As far as we know, this is the first work to investigate the Stacking cost-effectiveness [Cunha et al., 2021] of text classifiers based on neural networks and traditional strategies from the perspectives described above.

The **second main contribution** of the dissertation is the proposal of a method based on low-cost oracle that can predict the best stacking ensemble in a given scenario (with and without computational cost limitations) using only a fraction of the available training data. The oracle method first estimates the best unique algorithm (which can be seen as a baseline for effectiveness) to perform an efficient greedy search of ensembles guided by its effectiveness and efficiency relative to the best individual algorithm. The Oracle method predicts efficient ensembles successively, including algorithms that improve its efficiency using a Average Meta-layer strategy. Furthermore, the new method avoids the inclusion of computationally expensive algorithms (relative to the best individual algorithm) to guarantee the efficiency of the ensemble. This new proposed algorithm is the first known strategy to efficiently predict effective ensembles capable of dealing with practical efficiency issues related to our research questions. In more detail, this proposal aims to predict three ensembles corresponding to the time constraints of RQ1, RQ2 and RQ3, respectively, avoiding the potential high computational cost of evaluating expensive base models and their ensembles, especially on large data sets. Oracle's specific research questions are as follows:

- ORQ1: It is possible to predict, using a fraction of the training data, an effective ensemble that will tie or surpass the best learning model when trained with the entire training set available, at a lower cost than the of the best model?
- ORQ2: Is it feasible to make a prediction similar to ORQ1, but now with a cost less than or at most equal to the best model when trained with all training data?
- ORQ3: Without time constraints, is it possible to predict a combination that will be better than the best learning algorithm in a dataset?

The experimental results carried out in this dissertation show affirmative answers to the six research questions in most of the experiments. In most datasets, it is possible to obtain an ensemble of algorithms as good or better than the best individual algorithm

at a lower cost. In seven out of the eight experimented datasets, it is possible to obtain an ensemble with statistically significant gains concerning the best algorithm without increasing cost. Likewise, in seven of the eight datasets, the Oracle method provides results as good or (statistically significant) better than the best individual algorithm without increasing computational cost, providing empirical evidence for the practical benefits of the proposed Oracle.

The results of this dissertation generated a publication [Gomes et al., 2021] in The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021) in the Findings category. ACL is the most important worldwide conference on Natural Language Processing and Computational Linguistics (h5-index: 157).

1.3 Organization

This dissertation is organized as follows. Chapter 2 provides background and discusses some recent work on Stacking. Chapter 3 presents the methodology used to carry out the experiments performed to answer the six posed research questions. Chapter 4 describes the configuration for the execution of the experiments and discusses the respective results, which positively answer the research questions. Finally, Chapter 5 concludes the dissertation, highlighting the contributions and indicating possible paths for future studies.

Chapter 2

Related Work

Several types of textual representations and classification algorithms can be used to create a prediction model in Automatic Text Classification. In Section 2.1 we show which representations exist in the literature and which ones we use in this work, highlighting the textual information that each type of representation seeks to extract from the documents. In Section 2.2 we present the classification algorithms that we choose to use in this work, presenting the characteristics of each one to carry out the classification task. Finally, we describe in Section 2.3 the Stacking studies present in the literature, with focus on works that use Stacking in ATC.

2.1 Representations of Textual Data

Given a textual dataset defined by $\mathbb{D} = \{d_1, ..., d_n\}$, where each element of the set is a document and considering that the vocabulary of words in this *corpus* is defined by $\mathbb{V} = \{w_1, ..., w_m\}$, each document in $\mathbb{D}$ is composed by a set of words from $\mathbb{V}$, usually $\mathbb{V}_d \ll \mathbb{V}$. One of the main preprocessing steps consists of defining how to represent each word $w \in \mathbb{V}$ in each document $d \in \mathbb{D}$, on which it is necessary to extract useful information from each word.

Traditionally, texts have been represented with fixed-length vector representations (Vector Space Models), referred to in the literature as the bag-of-words (BoW). In this representation, the length of each vector is equal to the size of the collection's vocabulary and each element of the vector has a value corresponding to the weight of the term in the document that the vector represents. There are several robust strategies to weight the importance of terms, e.g., Term Frequency-Inverse Document Frequency (TFIDF) [Salton and Buckley, 1988]. The TFIDF representation generates a sparse vector ($|\mathbb{V}|$-dimensional) where each word has the TFIDF value represented in each document (with non-zero entries corresponding to the set of words $\mathbb{V}_d \subset \mathbb{V}$ observed in the documents).

Although TFIDF is by far the most used strategy, it does not consider the impor-

tance of co-occurrence of words in a document. In specific contexts, the co-occurrence of words can be pretty significant for the effectiveness of a learning model. A simple strategy to overcome this is to use the n-grams method [Cavnar et al., 1994]. The n-grams method take into account a sequence of n words that co-occur in a window (with size n); for a word w, the method considers n words before and after to represent a new word. Thus, the same TFIDF value is used, but this time applied to the n-grams. The use of n-grams has already demonstrated significant improvements in several domains of NLP (e.g., ATC, Natural Language Comprehension, etc), despite that the technique limits the capture of contextual information observed in non-sequential patterns.

Recently, much has been developed in terms of improving textual representation. Nowadays, the technique with the most promising results is the word embedding, such as Word2Vec [Mikolov et al., 2013], GloVe [Pennington et al., 2014], PTE [Tang et al., 2015] and FastText [Mikolov et al., 2017] which, based on word co-occurrence statistics in text datasets, represent words as vectors such that their similarities correlate with the semantic relationship (e.g., adjacent words of a target word). As shown in [Baroni et al., 2014], prediction models consistently outperform counting models in various tasks such as concept categorization, synonym detection, and semantic relationship tasks, providing strong evidence in favor of the supposed superiority of word embedding models.

In order to obtain a richer representation of the data, Mikolov et al. [2013] proposed the model Word2Vec. This unsupervised strategy aims to learn high-quality vectors for each word in the when training the model. This is achieved by a neural network trained with sequences of words that co-occur within a fixed-size window to predict the nth word, given the words $[w_1, ..., w_{n-1}]$ or the opposite. Each word is used as input to a log-linear classifier with a continuous projection layer, where the model predicts words within a certain distance (before or after) of the current word. Since more distant words are generally less related to the current word than those close to it, a lesser weight is placed on distant words, analyzing these words with less importance in the training step. The output is an array of word vectors represented in a vector space. The method can capture, efficiently and effectively, the semantic relationships between words given a large set of textual data and the model also allows arithmetic operations between those words. Unlike other distribution models, both Word2Vec and GloVe are predictive models in the sense that they aim to predict the occurrence of words rather than relying only on counting co-occurrence patterns. These models often bring richer representations that ultimately improve the learning capabilities of the supervised classification algorithms.

One of the main problems with Word2Vec and Glove is that they create the representations of words in an unsupervised way. Despite the generalist results of previous word embeddings, the approach makes poor representations for a specific task. Thus, Tang et al. [2015] created the algorithm called PTE, which aims to incorporate both unsupervised and supervised data to learn the embeddings of words. PTE works by

transforming a text dataset into three networks (called by the author the heterogeneous text network): a word-word network, a word-document network, and a third word-label network. With the heterogeneous text network, the authors perform predictive text embedding to learn low dimensional representations of each word in the corpus. The main advantage of PTE is that textual representations become optimized for specific tasks.

FastText [Mikolov et al., 2017] is another algorithm based on word embeddings, that aims to learn vectors for the sub-words found within each word, as well as the complete word. In each training step in FastText, the target word average and sub-word vectors are used for training. The fit calculated from the error is then used uniformly to update each of the vectors that have been combined to form the target word. This process adds many computing costs to the training stage. The resulting trade-off is a set of word arrays that contain embedded sub-word information. The authors claim that the potential benefits of FastText are: (i) the creation of better word incorporation for rare words; (ii) the use of character embedding for downstream tasks, resulting in a performance improvement when compared to word embedding such as Word2Vec and GloVe.

A recent approach that has gained considerable prominence as a textual representation due to its high efficiency at a low cost is the meta-features approach. Meta-features are algorithms that work at data engineering level to increase the information captured by standard representations, such as BoW. Current works in this area are on distance-based meta-features, that are approaches for extract information from the initial text representation input and capture distance relationship information between documents. To make the meta-features creation process, algorithms such as clustering, knn, or category centroid are used [Canuto et al., 2018, 2019].

The Text Representation sub-area can be considered an field for itself due to the high number of studies in the literature and the great opportunities for improvement. Our dissertation focuses not on improving textual representation but on using existing representations to answer our research questions. Therefore, we selected both the traditional representation (TFIDF) and state-of-the-art representations (FastText, PTE and Meta-features) to use as input for the supervised classifiers.

2.2 Supervised Classification Algorithms

Early efforts in ATC focused on improve machine learning algorithms using simple representation techniques such as bag-of-words (e.g., TFIDF Weighted). We have as examples the following supervised classification algorithms: Naïve Bayes, K-Nearest Neighbors (kNN), Support Vector Machine (SVM), Random Forest and Logistic Regres-

sion [Li et al., 2020; Kadhim, 2019]. Even with such simple document representation, the use of methods such as LinearSVM [Fan et al., 2008] and XGBoost [Chen and Guestrin, 2016] produces good results with efficient convergence for large datasets [Fan et al., 2008].

The Naïve Bayes algorithm [Zhang, 2004] is based on the Bayes Theorem, where the main characteristic of the algorithm is the independence between all features given the value of the class attribute. The independence assumption is not true in real-world problems, but this is used because the amount of combinations without this assumptions makes the algorithm impossible to train. Despite the weakness due to the independence between the features, Naïve Bayes achieves competitive effectiveness results compared to other simple supervised classification algorithms.

Another traditional and popular algorithm, especially for binary problems is Logistic Regression [Kleinbaum et al., 2002]. Despite the name, Logistic Regression is a supervised classification algorithm that tries to solve a problem by fitting it into a Logistic Function (i.e., S-curve). The logistic function is defined by $f(z) = \frac{1}{1+e^{(\alpha+\sum \beta_i X_i)}}$, where X_i is each dependent variable of the problem and α and β_i are constants. Therefore, given a sample in the form of z, $f(z)$ returns whether this sample is of class 0 or class 1. It is a straightforward but powerful algorithm, very useful for binary problems precisely because of the characteristics of the logistic function that intrinsic tries to better distinguish two distinct classes by adapting the S curve to a specific problem.

The k Nearest Neighbor technique (i.e., kNN) [Bhatia et al., 2010] is a algorithm that identifies the category of unknown samples according to the class of the k near neighbors to those samples. It is a simple algorithm that performs well in supervised classification problems, including the ATC task. However, it is interesting to note that, due to its simplicity, the traditional kNN algorithm has a high demand for memory and computional power. There is a wide field of study of variations of the kNN to address such limitations [Rocha et al., 2015].

Cortes and Vapnik [1995] proposed the algorithm that became state-of-the-art in text classification, being used until today in the NLP area with significant gains in different tasks: the Support Vector Machine (SVM) algorithm. It is an algorithm for binary classification, but it can be extended to several classes as a problem of multiple binary classifications. The main objective of the SVM is to find an optimal hyperplane that separates the classes of the binary problem using the multidimensional features available. It is important to emphasize that the choice of the kernel for SVM is essential for the algorithm to be robust to non-linear problems.

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 parallel benefits of transformers, better exemplified by Bidirectional Encoder Representations from Transformers (i.e., BERT) [Devlin et al., 2018]. Following

BERT's success, the recent XLNet neural 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.

2.3 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. We can make predictions in a different validation set with the trained models, which were not used for training. With the saved models and the predictions in the validation set, a *meta-layer* (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, 2021; Onan et al., 2016] and multi-label classification [Xia et al., 2021; Weng et al., 2019]. Particularly, stacking produced 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, 2021; Pelle et al., 2018; Onan et al., 2016].

Campos et al. [2017] created a new classification algorithm based on a traditional Random Forest variation (Extremely Randomized Trees) that exploits the use of boosting techniques to improve performance in topic categorization and sentiment analysis tasks. The algorithm was named as Boosting Extremely Randomized Trees (a.k.a., BERT). Besides the creation of a new algorithm, the work also carried out experiments based on the use of Stacking in these different tasks. Stacking was performed by analysing the diversity divergence of 9 base classifiers (SVM, Naive Bayes, KNN, etc). The authors obtained improvements of up to 20% in some cases, showing the great benefits of using Stacking in the ATC scenario. The authors also present the execution time results when using Stacking, emphasizing how the performance issue can be problematic in this scenario.

Stacking of a few traditional base algorithms reported highly effective results on tackling the very complex morphology of Arabic languages [Abuhaiba and Dawoud, 2017]. In that work, the authors explore the use of four types of ensemble for improving the task of Arabic Text Documents Classification: Fixed Combining Rules, Stacking, AdaBoost

and Bagging. Only the TFIDF text representation was used for all chosen base classifiers. All the ensemble techniques improved the accuracy up to 99%, where the Stacking was the most time consuming solution because of the implicit characteristic of training in two phases (training of both base classifiers and meta-layer).s Stacking of classifiers trained on different document representations (including pre-trained word embeddings-based features) also reported high effectiveness in multiple works [Carvalho and Plastino, 2021; Pelle et al., 2018; Onan et al., 2016]. For example, Carvalho and Plastino [2021] exploit the use of n-grams, meta-features and word embeddings for improving the task of Tweets Sentiment Analysis in 22 datasets. The authors tested the use of these different textual representations in two distinct ways: concatenation of the multiple feature spaces and the ensemble Stacking technique. In the experiments, Stacking with Logistic Regression as meta-layer outperformed the simple feature concatenation in 12 out of the 22 tweet datasets.

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. In [Reid and Grudic, 2009], the authors demonstrated the occurrence of overfitting in Stacking even when using simple linear algorithms. They also present an experimental study on how Ridge Regression, Lasso Regression and Elastic Net Regression can be used as regularization techniques in the meta-layer to deal with the overfitting problem.

Ledezma et al. [2010] propose a Genetic Algorithm to find the best Stacking configuration (GA-Stacking). The GA-Stacking selects which base algorithms should be include in the combination, the parameters of these algorithms and the meta-layer algorithm. They show how GA-Stacking overfitted due the high search space used in the genetic algorithm as well as a solution to avoid that – they exploited a validation set to calculate the fitness value of the genetic algorithm in each iteration. Gupta and Thakkar [2014] performed a summary study of the meta-heuristics techniques employed to solve the Stacking configuration issue. The authors highlight the advantages and drawbacks of each existing technique present in the literature, concluding the work with tips for future work in this area.

Most of the works that exploits Stacking to optimize the choice of a subset of base algorithms focused only on maximizing the ensemble effectiveness with no concern for efficiency (i.e., execution time). Also, the majority of existent works assume a pre-defined set of base algorithms to be combined in the Stacking ensemble. In this dissertation, we provide a thorough evaluation of the effectiveness and efficiency trade-offs of Stacking and 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.

Chapter 3

Methodology

This chapter describes our methodology and the inner workings of the new Oracle method. In Section 3.1, we present the Research Questions we aim to answer. The Stacking experiments were designed and run based on those questions. In Section 3.2, we present the implementation details of the Oracle algorithm whose goal is to reduce computational costs while maintaining or improving effectiveness. We also have research questions concerning the Oracle, based on the previous Stacking questions.

3.1 Time-Constrained Stacking

Our goal when using Stacking is to explore different ensemble combinations with the goal of outperforming the best individual algorithms at reduced costs. In this context, we aim to answer the following Research Questions (RQ's):

- Research Question 1 (RQ1): Is it possible to obtain an effective ensemble with less computational time than the best individual learning model?
- Research Question 2 (RQ2): Is it possible to improve the effectiveness of the best learning model using an ensemble without increasing computational time?
- Research Question 3 (RQ3): Disregarding computational time, is there an ensemble that can improve the efficiency concerning the best learning model?

With RQ1, we aim to identify whether it is possible to obtain a stacking of base algorithms 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 essential to indicate 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

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 computational cost of the best base algorithm as an upper limit.

Finally, 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 stacking over the best base algorithm in exchange for the additional computational cost. This scenario is where most current studies work. It would be the use of Stacking to find the best ensemble combination regardless of the computational cost necessary to obtain it.

It is extremely important to point out that we assume that the base algorithms can be executed in parallel on different machines (both in Stacking and in the Oracle experiments). as such, each base algorithm has training execution independent of the others, and Stacking is only responsible for combining them in a meta-layer. We can make this restriction since if each algorithm of a Stacking ensemble were required to execute sequentially, this would only worsen Stacking time and the Oracle would have a large margin of advantage concerning time, since the Oracle uses only a fraction of the training set.

3.2 Oracle-Based Prediction of Stacking Performance

Choosing the best possible ensemble configuration is a difficult problem since it is impossible to run the experiments with all possible combinations depending on the number of base algorithms. Thus, in addition to the Stacking experiments, we propose in this dissertation to solve the problem of choosing the best ensemble among the possible combinations with the base algorithms. For this, we propose the strategy which we call the Oracle. Oracle allows a simple way to select competitive ensemble sets at a time (computation cost) strictly smaller than the best base algorithm.

We implement the proposed strategy as follows: (i) each base algorithm is trained with a reduced amount of the training set (e.g., 10%, 15%, 30%, etc.); (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 returns the best algorithm based on the validation set. For each iteration, the following best algorithm, as estimated in a validation set, is added and we verify whether the combined result presents a statistically significant improvement ($\alpha = 0.05$) concerning 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 (i.e., validation set) that is not contained in the minor part used in training. Besides, as a meta-layer, we use a simple average, i.e., we 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 $\mathbb{E} \subset \mathbb{A}$ is the best set of algorithms based on the current combination. As it is a simple meta-layer and not a learning algorithm, the cost can be considered insignificant in the choice process. This choice of the simple meta-layer is essential because the meta-layer algorithm can become more computationally expensive than training the base algorithms.

Algorithm 1 Oracle Algorithm

```
procedure ORACLE(𝔸)
    ℂ ← Best(𝔸)
    𝕊 ← 𝔸 − ℂ
    while 𝕊 ≠ ∅ do
        X ← Best(𝕊)
        𝔼 ← ℂ ∪ X
        if Avg(𝔼) > Avg(ℂ) then
            ℂ ← 𝔼
            𝕊 ← 𝔸 − ℂ
        else
            𝕊 ← 𝕊 − X
    return ℂ                                  ▷ Best combination
```

With the definition of how the Oracle algorithm works, we state the three Research Questions for the Oracle experiments:

- Oracle Research Question 1 (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?
- Oracle Research Question 1 (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?
- Oracle Research Question 1 (ORQ3): With no time constraints, can we predict a combination that will be better than the best learning algorithm in a dataset using a smaller piece of training?

Answering the Oracle research questions, we will be able to determine whether our method can achieve the main objective for which the method was created: discover efficient and competitive ensembles in a shorter computational time than the best base algorithm.

Chapter 4

Experiments and Results

In this chapter, we introduce the experimental settings and discuss the results. We start by presenting in Section 4.1.4 the configurations used to run the Classification, Stacking and Oracle experiments. We present the datasets chosen for the experiments, the Textual Representations used, the Supervised Classification Algorithms exploited in the ATC tasks, as well as other configurations to allow the replication of the experiments performed in this dissertation.

After defining the settings, we initially present the ATC results of the individual base algorithms in Section 4.2. Although the main focus of the dissertation is not to assess the classification effectiveness of individual algorithms but rather that of the ensemble (Stacking), we highlight in this section the results of each individual base algorithm and the conclusions we can draw from using them. This analysis of individual algorithms is important since we always compare each dataset's best individual algorithm with the Stacking and Oracle´s experimental results.

Next, we present the results and analysis of the Stacking experiments in Section 4.3 with the comparisons of the Stacking results versus individual base algorithms. We also discuss the answers for research questions RQ1, RQ2 and RQ3 defined in Chapter 3. In Section 4.4 we present and discuss the results of the Oracle method and the answers for the corresponding research questions ORQ1, ORQ2 and ORQ3. Finally, in Section 4.5 we have a summary of the results achieved in our vast experimental environment.

Dataset	# Doc.	# Feat.	# Classes	Class Distribution				Avg Doc.	Skewness
				Minor	Median	Mean	Major		
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
AGNews	127,600	39,837	4	31,900	31,900	31,900	31,900	37	Balanced
IMdB	348,415	115,831	10	12,836	31,551	34,841	63,233	326	Imbalanced
Reut	13,327	27,302	90	2	29	148	3964	171	Imbalanced
Sogou	510,000	98,974	5	102,000	102,000	102,000	102,000	588	Balanced
WebKB	8,199	23,047	7	137	926	1,171	3705	209	Imbalanced
Yelp	700,000	115,371	5	140,000	140,000	140,000	140,000	136	Balanced

Table 4.1. Statistics of Automatic Textual Classification (ATC) datasets.

4.1 Execution Configurations

4.1.1 Textual Datasets

We consider the effectiveness and efficiency of the models on eight datasets very known by the ATC community. Of those eight datasets, four are large-scale datasets (more than 100,000 documents) [Zhang et al., 2015; Diao et al., 2014] – AGNews, IMdB Reviews, Sogou and Yelp – and four are mid-sized datasets [Canuto et al., 2014, 2018] – 20NG, ACM, Reut and WebKB.

In Table 4.1 we have a summary of each of the datasets. We present the number of documents (# Docs), how extensive the vocabulary is (# Feat.), how many classes (# Classes), information on the distribution of documents between classes (Class Distribution) and the average number of words per document. In Class Distributions, we can see the smallest amount of documents in a class (Minor), median amount (Median), average amount (Mean) and the largest amount (Major). It is important to highlight all this information since the different characteristics of these datasets lead to different challenges for the ATC algorithms. Below we have the information of the textual data and classes for each dataset:

- 20 Newsgroups (**20NG**): The 20 Newsgroup dataset is a set of 18,846 documents divided into 20 different newsgroups, with each group having a specific topic (e.g., politics, religion, sport, etc.). It is a popular dataset in the ATC area, as it has the characteristic of having documents with approximately equal amounts among the different 20 classes (i.e., balanced dataset).

- ACM-DL (**ACM**): It is a dataset composed of 24,897 documents from the ACM Digital Library. All articles are from the Computer Science field, and the objective

is to classify each article in 11 classes of the first level of the taxonomy adopted by the ACM.

- AG's News (**AGNews**): It is a dataset of news articles collected from the web[1], containing 127,600 documents and 4 classes. The documents come from 496,835 categorized news articles and over 2,000 news sources. As for the class, they were created according to the title and description of each article. This dataset has an equal balance for all classes, each one having the same amount of documents.

- IMdb Reviews (**IMdB**): This dataset was created from the movie reviews site IMdB[2], where the creators of the dataset selected 50,000 random movies and crawled all of their reviews. This dataset has ten classes, which refer to user ratings (scaled from 0 to 10) in the selected movies. The resulting dataset has 348,415 documents and a high unbalance of documents in the ten classes.

- Reuters (**Reut**): The Reuters dataset is a famous benchmark dataset for ATC created by Carnegie Group, Inc. and Reuters, Ltd. This dataset is composed of news articles from the Reuters[3] website, where there are 13,327 documents distributed among 90 categories. It is a dataset well known for its high degree of unbalance between classes, where there are classes with only two documents.

- Sogou News (**Sogou**): Created from the Chinese search engine Sogou, the dataset was generated from a total of 2,909,551 news articles from different topics. The resulting dataset has 510,000 documents and five balanced classes (sports, finance, entertainment, automobile and technology). As it is a Chinese dataset, it was necessary to do a specific preprocessing to transform the dataset to produce Pinyin – the phonetic romanization of Chinese. Preprocessing was done with the *pypinyin* with *jieba* Chinese segmentation system package, as indicated by the authors in [Zhang et al., 2015].

- The 4 Universities Data Set (**WebKB**): This dataset was collected from the Computer Science departments of several universities in January 1997 by the World Wide Knowledge Base (WebKB) project of the Carnegie Mellon University (CMU) group. There are a total of 8,199 web pages that have been collected and that have been separated into the following categories: student, faculty, staff, department, course, project and other.

- Yelp Review Full (**Yelp**): The Yelp Reviews Full is a very popular ATC dataset that was created for the Yelp Dataset Challenge in 2015. The Yelp website is a tool for

[1] `http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`
[2] `https://www.imdb.com`
[3] `https://www.reuters.com`

users to rate establishments and can also generate their reviews about places. This dataset has 700,000 documents and five classes, where each document is a user's review and each class is the number of stars of the respective review. This dataset is balanced, where each class has the same amount of documents.

4.1.2 Textual Representations and Supervised Classification Algorithms

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., 2015], 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., 2019, 2018, 2016; Cunha et al., 2020a, 2021]. In Table 4.2 we have all the settings we used for each chosen textual representation. Before creating each representation, we performed simple preprocessing steps: transformation to lowercase and removal of special characters and accents.

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 4.2. Parameter settings for textual representations.

In terms of supervised classification algorithms (i.e., base algorithms), we consider the LinearSVM [Fan et al., 2008], kNN [Altman, 1992], LogisticRegression [Fan et al., 2008], XGBoost [Chen and Guestrin, 2016], XLNet [Yang et al., 2019] and BERT [Devlin et al., 2018]. The parameters tuned for the non-neural algorithms are shown in Table 4.3

while the parameters of neural-based algorithms area presented in Table 4.4. The implementations of LinearSVM, kNN and LogisticRegression are from scikit-learn[4] [Pedregosa et al., 2011] and XGBoost [Chen and Guestrin, 2016] is from the respective authors implementation-based package[5]. Omitted parameters are the libraries default. The Table 4.3 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.

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 4.3. Parameters tuned for each non-neural supervised classification algorithms. Remaining parameters are the official libraries default.

Table 4.5 has a summary of the base algorithms and respective representations used in each of them. We also create a unique identifier with letters of the alphabet for each algorithm/representation configuration. We create these identifiers to provide cleaner graphics and analysis, as there may be stacking ensembles putting all of these algorithms together, which can make visualizing/reading the results a difficult task.

[4] `https://scikit-learn.org/stable/index.html`
[5] `https://xgboost.readthedocs.io/en/latest`

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 4.4. Neural networks parameters and pretrained models.

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

Table 4.5. All base algorithms with their repespective representations. Each configuration of individual classifiers has a unique ID.

4.1.3 Stacking

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 valid and it is necessary to execute the base algorithms and combinations on one single machine, this would only 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.

To train the Meta-Layer, responsible for learning to combine the outputs of the different individual classifiers, we use the following algorithms: Majority Vote (hard/soft), Mean, Median and LinearSVM. Majority Vote Hard chooses the class that most classifiers have output and does not take probabilities into account. Majority Vote Soft works the same as Hard but using probabilities instead of predicted class labels. Mean and Median apply their respective operations on the output probabilities to combine the base classifiers. Finally, LinearSVM simply uses probabilities to train the Meta-Layer, allowing the Meta-Layer to intelligently and automatically learn how to combine the different classifiers using a supervised classification algorithm.

For didactic purposes, we will demonstrate how training and stacking prediction works when using one meta layer. We will see how the training set is used to train the Meta-Layer, where the Meta-Layer learns how to put together the probabilities of different classifiers without overfitting, and how the meta-layer is used to predict the test set.

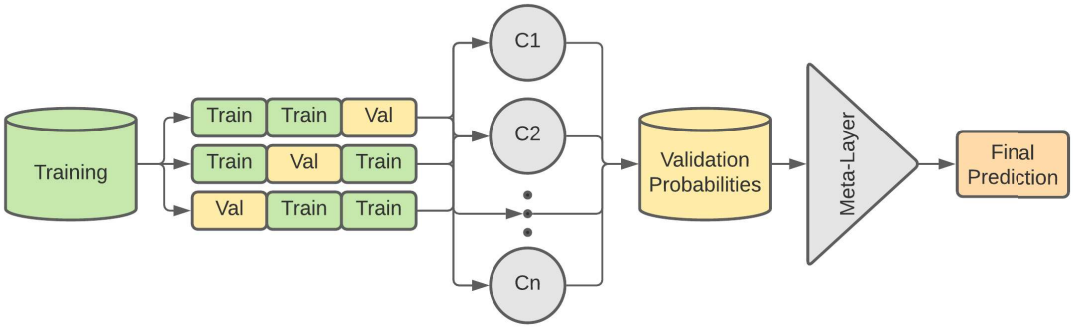

Figure 4.1. Stacking Meta-Layer training process.

In Figure 4.1 we have the representation of the Stacking Meta-Layer training process. We exemplify the training process using 3-fold cross validation and n classifiers (represented by C1, C2, ..., Cn). The process starts by separating the training set into 3 folds, where each classifier will be trained with 2 folds and the remaining fold is used to extract the probabilities. By repeating this process 3 times, we get a set of probabilities validation, which is used to train the Meta-Layer. It is worth to mention that the number of records in the probabilities validation set is the same as in the original training set. We

could also simply train with every workout and use training odds to train the Meta-Layer. However, this strategy tends to overfit to the training set distribution [Tang et al., 2014].

With the Meta-Layer trained, it is now enough to perform the test prediction. The prediction process is much simpler than training, as shown in Figure 4.2. First it is necessary to train the individual classifiers with the complete train set, exactly as is normally done in the supervised classification task. With the individual algorithms trained, we predict the test set in each classifier, thus building the input to predict the test set in the Meta-Layer. Meta-Layer then informs which class is predicted for each record according to the input probabilities.

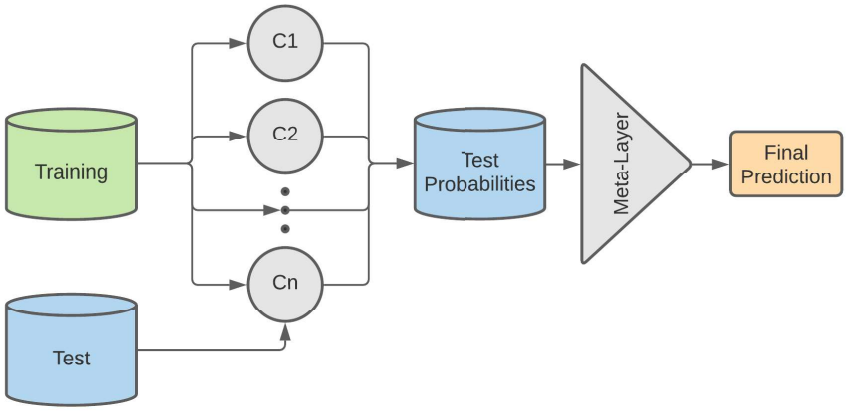

Figure 4.2. Stacking Meta-Layer predict process.

4.1.4 Statistical Techniques, Supervised Classification Metrics and Machines

The experiments in the smaller datasets were executed using a 10-fold cross-validation procedure, while in the larger we used 5-fold due to the computational cost. The algorithms parameters were tuned using the Bayesian Optimization [Bergstra et al., 2015] approach with ten iterations, with the 5-fold stratified strategy and the training set (nested cross-validation). In the Table 4.3, we have the values of each parameter that we optimize in the non-neural base algorithms. 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]. The parameters of the neural-based algorithms are present in Table 4.4.

To generate the predicted classes and probabilities used as input to the Meta-Layer,

we used another internal 5-fold stratified cross-validation to separate only the training set into training/validation. It is important to run the cross-validation again as if we use the probabilities of the entire train set since there may be overfitting issues [Wolpert, 1992; Tang et al., 2014]. The optimization of hyperparameters for this new cross-validation uses the same scheme as the individual algorithms with every training.

We evaluate all methods, combined with different representations, concerning 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 overall decisions, MacroF1 measures each class's classification effectiveness, averaging them, crucial for skewed datasets. In addition to effectiveness, we also assess the cost of each method in terms of 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].

In our experiments, we adopt the cloud environment Amazon Web Services Elastic Computing (i.e., AWS EC2)[6] 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.

4.2 Supervised Classification Results

We can see in Figures 4.3 and Figure 4.4 the results of the individual classifiers for the mid-sized and larges datasets, respectively. In each figure, we have the 18 combinations of classifiers/representations chosen and their respective results for the MicroF1 and MacroF1 metrics. In each figure, we can also see the statistical draws for both metrics marked with * in the classifiers IDs.

Exploring the results in Figure 4.3, we can see the performance for the 20NG, ACM, Reut and WebKB datasets. In the 20NG dataset, the IDs of the individual classifiers that have the best results are DHL – these classifiers use the Metafeatures as a textual representation as input. Among these top classifiers, the highest mean algorithm

[6] `https://aws.amazon.com/pt/ec2`

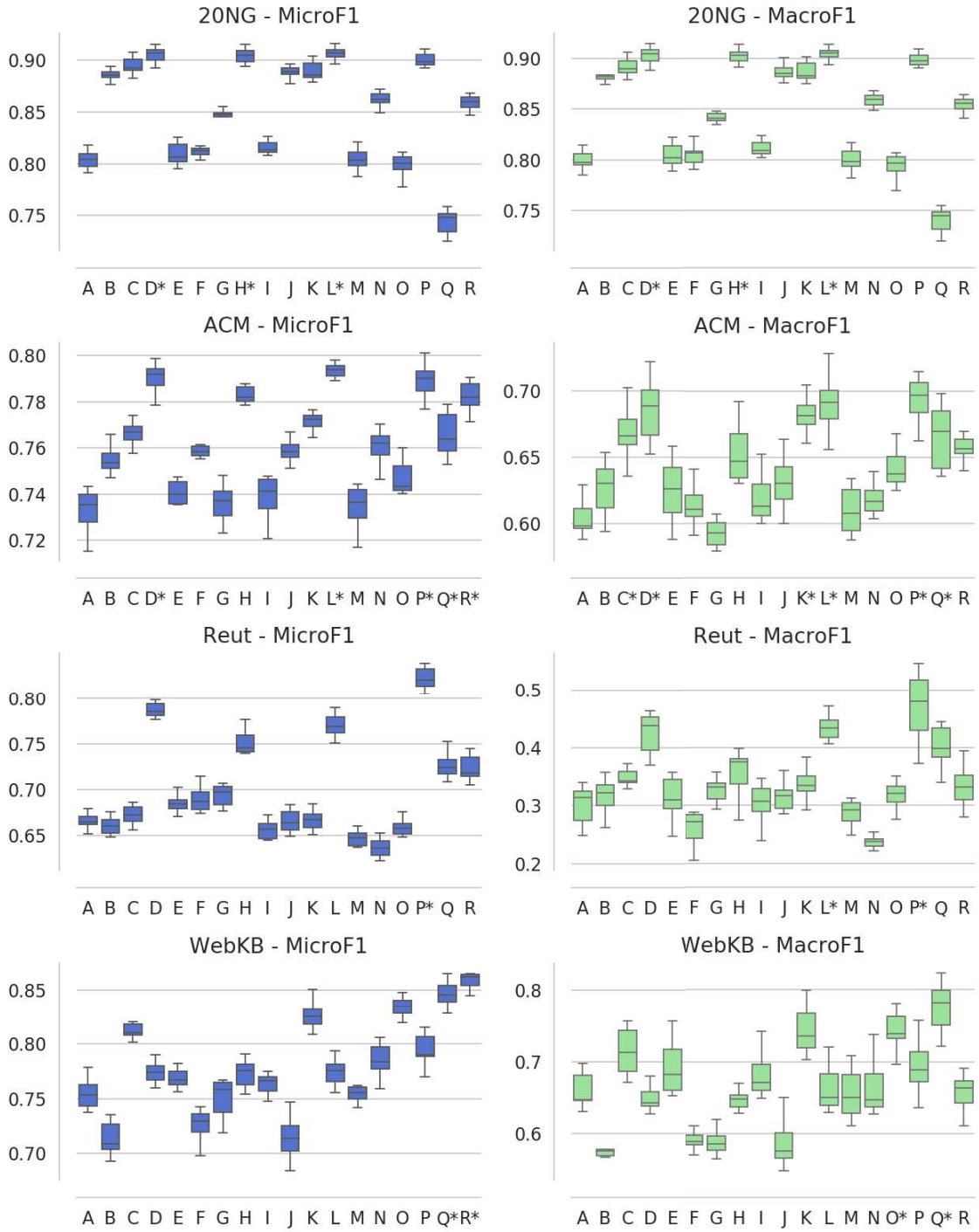

Figure 4.3. Classification results of each algorithm/representation for mid-size datasets: 20NG, ACM, Reut and WebKB. In blue we have the MicroF1 metric, while in green we have the MacroF1 metric.

for MicroF1/MacroF1 is L (Logistic Regression + Metafeatures), where the mean of MicroF1 is 0.907 ± 0.008 and MacroF1 is 0.905 ± 0.008.

In the ACM dataset, the algorithms with statistical ties for MicroF1 are DLPQR and for MacroF1, CDKLPQ. Again, as in 20NG, we highlight the algorithms that use Metafeatures along with algorithms that exploit TFIDF and those based on neural networks (XLNet and BERT). The algorithm with the highest MicroF1 average absolute value among the ties is L (Logistic Regression + Metafeatures) with a value of 0.791 ± 0.007, while for MacroF1 is P (XGBoost + Metafeatures) with a result of 0.693 ± 0.017.

In Reuters, one of the most challenging datasets due to class imbalance, the best classifier in terms of MicroF1 is P, while for MacroF1 both, L and P, tie. Again these are classifiers that use Metafeatures as input, where the classifier with the highest absolute average value for both metrics is P (XGBoost + Metafeatures), with a value of 0.821 ± 0.012 for MicroF1 and 0.473 ± 0.0569 for MacroF1.

The last mid-sized dataset is WebKB, where we see QR classifiers for MicroF1 and OQ for MacroF1 among the best results. We can see the presence again of neural networks in the ties for both metrics. The classifier with the highest average value among the ties for MicroF1 is R (XLNet) with a value of 0.860 ± 0.010, while for the MacroF1, it is Q (XGBoost + TFIDF) with a value of 0.777 ± 0.033.

For these mid-sized dataset results, it is interesting to highlight the high presence of classifiers that exploit Meta-features among the best. We can see also that neural networks are among the best results, but there are less complex and less computationally costly algorithms that tie or beat the neural-based classifiers in the scenario.

We now look at the results for the large datasets (AGNews, IMdB, Sogou and Yelp) in Figure 4.4. In these larger datasets, we can see that neural-based classifiers stand out among the ties/wins and that the most basics non-neural classifiers become less present among the best results.

Analyzing the results of the AGNews dataset, we see that the two neural networks Q (BERT), R (XLNet) are among the best results. The best average result for the metrics is the R classifier, with MicroF1 equal to 0.937 ± 0.002 and MacroF1 equal to 0.937 ± 0.002. In IMdB and Yelp datasets, neural networks are also highlighted. In IMdB, classifier R (XLNet) obtains an effectiveness of 0.389 ± 0.002 for MicroF1 and 0.3408 ± 0.002 for MacroF1. We also note a tie in the MicroF1 metric with the non-neural classifier J (Logistic Regression + PTE). In the Yelp dataset, the best classifier is also XLNet, with a value of 0.650 ± 0.001 for MicroF1 and MacroF1. In the Sogou dataset, we have different behavior from the previously mentioned large datasets, where the non-neural classifier O (XGBoost + TFIDF) produces the best results for both metrics. This classifier has a value of 0.957 ± 0.0009 for both metrics. It is interesting to note that even in a large dataset, Metafeatures are also effective.

To summarize the number of ties/wins of each classifier, we created Table 4.6, where we can see the classifiers that obtained at least one tie/win for each metric. The idea is simple – if a classifier gets a tie or a win in any dataset or metric, one point

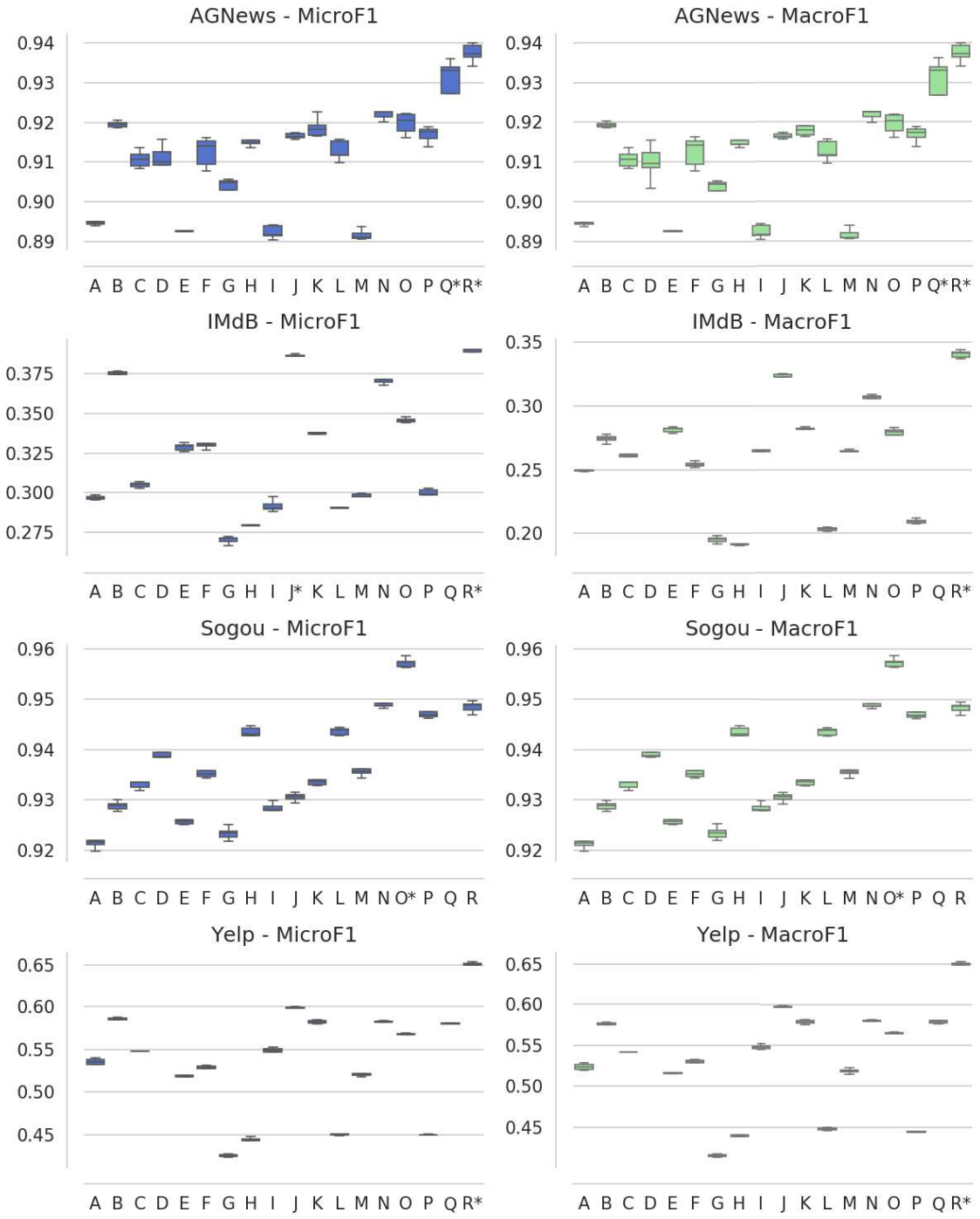

Figure 4.4. Classification results of each algorithm/representation for large datasets: AGNews, IMdB, Sogou and Yelp. In blue, we have the MicroF1 metric, while in green, we have the MacroF1 metric. Classifiers D and Q appear with some null values in the figure as some folds have been overfitted. Since these classifiers underperformed due to overfitting, we omit them to improve the visual quality of the figure.

Metric	Algorithm	Representation	ID	Wins/Ties
MicroF1	BERT	Raw Documents	R	5
	XLNet	Raw Documents	Q	3
	LinearSVM	Metafeatures	D	2
	Logistic Regression	Metafeatures	L	2
	XGBoost	Metafeatures	P	2
	XGBoost	TFIDF	O	1
	kNN	Metafeatures	H	1
	LogisticRegression	PTE	J	1
MacroF1	BERT	Raw Documents	R	3
	XLNet	Raw Documents	Q	3
	Logistic Regression	Metafeatures	L	3
	XGBoost	TFIDF	O	2
	XGBoost	Metafeatures	P	2
	LinearSVM	Metafeatures	D	2
	LinearSVM	TFIDF	C	1
	kNN	Metafeatures	H	1
	LogisticRegression	TFIDF	K	1

Table 4.6. Wins/Ties in classification for each effectiveness metrics.

is added to the Wins/Ties count. This table shows how neural networks stand out for both metrics due to their performance in large datasets. We can also see non-neural algorithms are not behind in terms of effectiveness, presenting satisfactory Wins/Ties and competitive results with neural networks in smaller datasets. As highlighted in the analyses, it is interesting to note the high presence of Metafeatures representations among the Wins/Ties of non-neural classifiers.

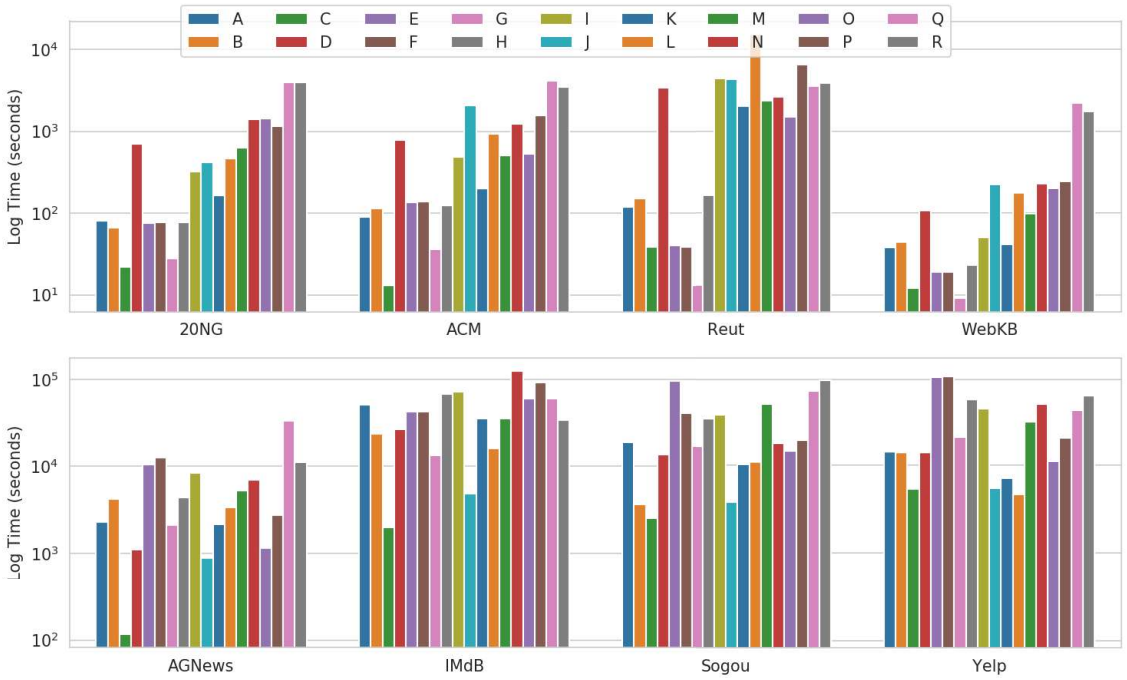

Figure 4.5. Average execution time, in logarithmic scale, for each base classifier in each chosen dataset.

Finally, we can see in Figure 4.5 the average execution times (in logarithmic scale) of each base classifier in the mid-sized and large datasets. We can see that neural-based classifiers are more computational costly than some more basic classifiers in mid-sized datasets – in some cases, this time is twice as long. In Reuters, there is an interesting case, where even some basic algorithms (e.g., E and P) take longer to execute than neural networks due to the complexity posed by class imbalance in this dataset. In large datasets, times increase a lot for non-neural classifiers, precisely because of the extra amount of documents and, consequently, the number of features (i.e., words) that these datasets possess. Looking closely at times, we can see that non-neural classifiers have a significantly lower computational time than neural-based classifiers.

The effectiveness and time results from the base classifiers presented in this section are used as baselines for the subsequent analyses. In next section, we will answer the posed Stacking and Oracle research questions, taking into account the best base classifiers in each dataset.

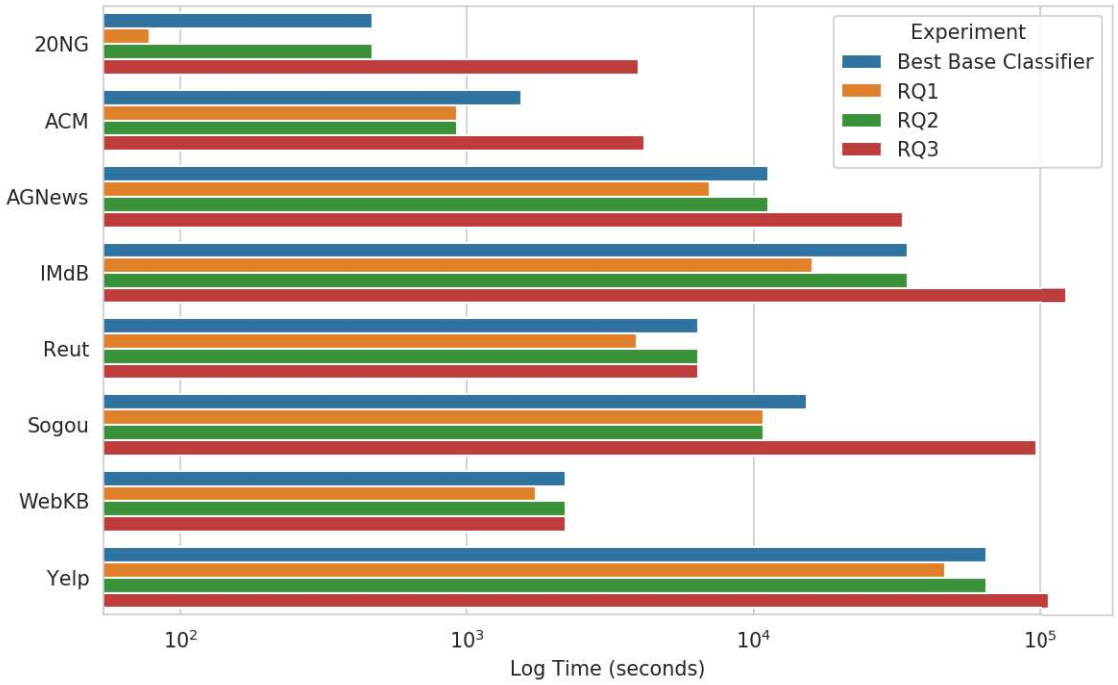

Figure 4.6. Average time (in log scale) of the Best Base Individual Classifier and Stacking Research Questions for each dataset.

4.3 Stacking Results

Results for RQ1, RQ2 and RQ3, in terms of MacroF1 for each dataset are shown in Tables 4.7, 4.8, and 4.9, respectively, while Figure 4.6, shows the analysis of the computational cost. 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 4.5), 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 since it is harder to improve them in the highly skewed scenario in most experimented datasets, as we saw in the classification section. However, we also consider MicroF1, whose results are summarized in Table 4.10.

In Table 4.7, which focuses on RQ1 that has a strong constraint in terms of execution time, we can see that in 5 out of 8 datasets, it is possible to obtain a combination of classifiers (stacking) that is 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. The gains in terms of cost (time) are very significant (see Figure 4.6), ranging

from 1.87x speedup improvement (in Reuters) to 7.16x (in WebKB). Speedups in 20NG, ACM and Sogou were 6x, 6.6x and 1.4x, respectively. Even if we consider the three cases in which there were some minimum effectiveness losses (0.39% in AGNews, 0.91% in IMdB and 4.67% in Yelp), there are some significant speedups: 1.6x in AGNews, 2.15x in IMdB and 1.47x in Yelp. Some chosen stacked combinations are interesting, such as EFGH in 20NG that contains all versions of kNN, and IKL in ACM, containing three versions of Logistic Regression. Both combinations contain classifiers with Metafeatures.

Dataset	Experiment	MacroF1	Combination	Most Costly
20NG	Best Base	90.58	L (Log. Reg. + MetaFeat)	
	RQ1	● 91.02	EFGH	H
ACM	Best Base	69.32	P (XGBoost + MetaFeat)	
	RQ1	▲ 71.50	IKL	L
Reut	Best Base	47.37	P (XGBoost + MetaFeat)	
	RQ1	● 46.98	JK	J
WebKB	Best Base	77.76	Q (XLNet)	
	RQ1	● 79.61	HLPR	R
AGNews	Best Base	93.74	R (BERT)	
	RQ1	▼ 93.37	MNOP	N
IMdB	Best Base	34.09	R (BERT)	
	RQ1	▼ 33.18	JL	L
Sogou	Best Base	95.73	O (XGBoost + TFIDF)	
	RQ1	● 95.70	JKL	L
Yelp	Best Base	65.07	R (BERT)	
	RQ1	▼ 60.43	IJK	I

Table 4.7. Stacking results for RQ1.

Results for RQ2 (Table 4.8) are also very interesting. In 6 out of eight 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[7], 3.1% in ACM, 5.4% in IMdB, 9% in WebKB and 1.52% in Yelp. Reuters is only considered a tie because of the high variability of the results across folds in this dataset (due to a 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 confirm statistically significant gains in Reuters (See Table 4.10). Finally, in the Sogou dataset, there was a tie when using Stacking.

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. In ACM and Sogou, the base algorithm is not part of the combination.

[7]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	L (Log. Reg. + MetaFeat)	
	RQ2	▲ 91.63	IJKL	L
ACM	Best Base	69.32	P (XGBoost + MetaFeat)	
	RQ2	▲ 71.50	IKL	L
Reut	Best Base	47.37	P (XGBoost + MetaFeat)	
	RQ2	● 51.99	NOP	P
WebKB	Best Base	77.76	Q (XLNet)	
	RQ2	▲ 84.07	All	Q
AGNews	Best Base	93.74	R (BERT)	
	RQ2	▲ 94.12	CHQR	R
IMdB	Best Base	34.09	R (BERT)	
	RQ2	▲ 35.95	LR	R
Sogou	Best Base	95.73	O (XGBoost + TFIDF)	
	RQ2	● 95.70	JKL	L
Yelp	Best Base	65.02	R (BERT)	
	RQ2	▲ 66.01	CHLPQR	R

Table 4.8. Stacking results for RQ2.

Notice also that the gains are somewhat limited due to the restricted number of classifiers that can be combined due to the time constraints, impacting 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 by including more costly classifiers. There are further gains in AGNews (0.94%), 20NG (2.06%), IMdB (5.8%), ACM (6.32%), Sogou (1.49%) and Yelp (2.59%). Notice that there is a tendency in this scenario to include most algorithms in the combinations, like in ACM, WebKB, AGNews, Sogou and Yelp, to obtain further improvements, meaning 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 was chosen in scenario RQ2. This combination exploits the most effective and complementary algorithms and may not even include the base classifier. In other cases, such as IMdB, a combination of a few of the most effective (and costly) algorithms suffice to obtain larger gains, meaning that the meta-layer is 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 4.6 for the cases of 20NG, ACM and AGNews. In those datasets, the costs have tripled (AGNews), quadrupled (ACM and IMdB) or become 8x more expensive. It is up to the application designer to decide whether this

cost-effectiveness trade-off is worth it.

Dataset	Experiment	MacroF1	Combination	Most Costly
20NG	Best Base	90.58	L (Log. Reg. + MetaFeat)	
	RQ3	▲ 92.45	CHQR	Q
ACM	Best Base	69.32	P (XGBoost + MetaFeat)	
	RQ3	▲ 73.70	All	R
Reut	Best Base	47.37	P (XGBoost + MetaFeat)	
	RQ3	● 51.99	NOP	P
WebKB	Best Base	77.76	Q (XLNet)	
	RQ3	▲ 84.07	All	R
AGNews	Best Base	93.74	R (BERT)	
	RQ3	▲ 94.63	All	Q
IMdB	Best Base	34.09	R (BERT)	
	RQ3	▲ 36.06	PR	P
Sogou	Best Base	95.73	O (XGBoost + TFIDF)	
	RQ3	▲ 97.16	All	R
Yelp	Best Base	65.02	R (BERT)	
	RQ3	▲ 66.71	All	F

Table 4.9. Stacking results for RQ3.

Table 4.10 summarizes the effectiveness results. For RQ1, there are ten win/ties out of 16 possibilities (8 datasets, two metrics). Remind that in this scenario, ties are considered a good result due to the reduction in costs. We also have 13 wins for RQ2 and 15 wins for RQ3, only ties in Reut and Sogou with no loss at all. In terms of cost (Figure 4.6), significant reductions in scenario 1 (RQ1) can be obtained in all eight 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 compared to the cost of the base classifier. Furthermore, 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	3	3	1	4	3
RQ2	7	1	0	6	2	0
RQ3	8	0	0	7	1	0

Table 4.10. Win/Tie/Loss summary for Stacking research questions.

4.4 Oracle Results

This section presents the results related to research questions for the Oracle method proposed in this dissertation. The MacroF1 results of the Greedy Oracle predictor are shown in Tables 4.11, 4.12 and 4.13. These results correspond to an Oracle that uses the base algorithms trained with 30% of the training data and predicts in a different training data portion in a folded cross-validation procedure, using an simple Average meta-layer. The mean runtime for each dataset, comparing the oracle results against the best base classifiers, are shown in Figure 4.7.

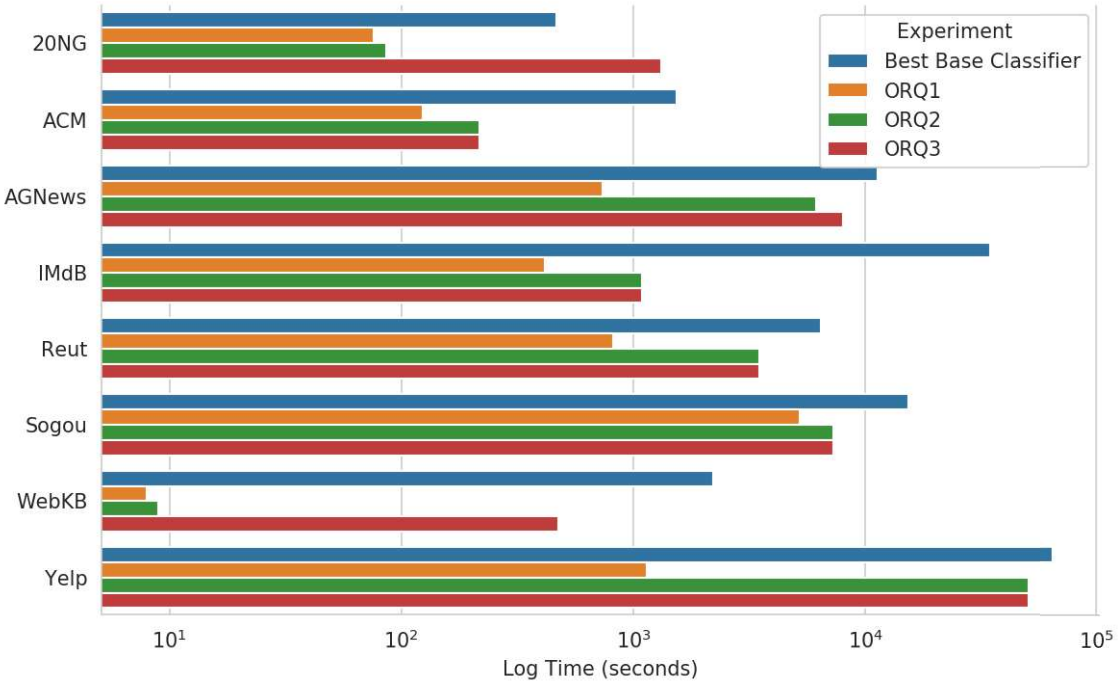

Figure 4.7. Average time (in log scale) of the Best Base Individual Classifier and Oracle Research Questions for each dataset.

We will start by analyzing the ORQ1 research question, which limits the inclusion of classifiers in the oracle stacking ensemble by the time of the best base algorithm with 30% of training, that is, only classifiers with time strictly shorter than the best base algorithm can be part of the stacking ensemble in this research question. In Table 4.11 we have the results for ORQ1, where we can see that Oracle can achieve good predictions in 3 of the 8 datasets in this scenario. In the table we show the best base classifiers with 100%, as they are the baseline we want to compare and prove that we can maintain/improve effectiveness while reducing computational cost (i.e., efficiency). We

can see in the table that we achieved competitive ensemble using only 30% in 3 datasets, against the best classifiers trained with all training data.

In the 20NG and ACM datasets we get better results than the best base algorithm with 100%, which is a great improvement since we are not including the best base algorithm and we are significantly reducing the execution time. In the WebKB we can tie even while reducing the computational cost, another very relevant result since we are reducing the time without losses in the F1Macro metric. However, even in the case where there were losses, some were minimal, like in AGNews with a loss of only 0.62% with potential gains in training time. In the Sogou dataset, there was a slight reduction of effectiveness of approximately 1%, while there was a time reduction. Only in Reuters, IMdB and Sogou were significant MacroF1 losses, mainly due to the failure to predict which would be the best base algorithm and the impossibility of including the predicted best base algorithm in the combination. For Reuters and Sogou, the best base algorithms are respectively XGBoost + MetaFeat (P) and XGBoost + TFIDF (O), while BERT (R) is the best base algorithm for IMdB. We can see in these datasets that there really was a reduction in computational cost, but none of these algorithms with 30% of training achieved results close to those of the best base algorithms.

The table shows that in large datasets the strategy did not achieve a satisfactory result for ORQ1 since most of these datasets have a neural network as the best based algorithm and the amount of data limits the inclusion of networks in the combination. As the time of neural networks is longer due to computational complexity, this allows the inclusion of more classifiers in the scenario shown in RQ1 in the Stacking section (Table 4.7). It is essential to highlight that we do not know what will be the best algorithm when using all the training data or its effectiveness in an actual real execution scenario. Indeed, with more data, there is a tendency for some algorithms, such as the transformers, to improve their effectiveness, but their sound performance may not be predicted with few training data. Remind also that this is a stringent 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.

When we are allowed to include the best-predicted algorithm in the stacking scenario for ORQ2, the results show a drastic improvement – we can make a reasonable prediction in 7 out 8 cases (3 wins and 4 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 low cost (Figure 4.7), as an excellent result. In this scenario 20NG, ACM and Yelp datasets have the bests results, where we can improve the MacroF1 metric at the same time we can predict the best base algorithm with 100% of training. However, even when we cannot predict the best algorithm, it is possible to find a combination of less expensive algorithms that can tie with the best base algorithm. This is the case, for

Dataset	Experiment	MacroF1	Combination	Most Cost
20NG	Best Base	90.58	L (Log. Reg. + MetaFeat)	
	ORQ1	▲ 91.62	ABCDHK	H
ACM	Best Base	69.32	P (XGBoost + MetaFeat)	
	ORQ1	▲ 71.85	ABCDHIJKL	C
Reut	Best Base	47.37	P (XGBoost + MetaFeat)	
	ORQ1	▼ 39.60	ABDPQ	P
WebKB	Best Base	77.76	Q (XLNet)	
	ORQ1	● 75.79	BCE	E
AGNews	Best Base	93.74	R (BERT)	
	ORQ1	▼ 93.16	BDFJNP	B
IMdB	Best Base	34.09	R (BERT)	
	ORQ1	▼ 27.79	K	K
Sogou	Best Base	95.73	O (XGBoost + TFIDF)	
	ORQ1	▼ 94.73	P	P
Yelp	Best Base	65.07	R (BERT)	
	ORQ1	▼ 59.74	J	J

Table 4.11. Oracle results for ORQ1. Output ensemble by the Oracle using 30% of training data *versus* the best base classifier using all training data. In this scenario the ensemble generated by the Oracle has to be strictly less expensive than the best base algorithm with 30% training.

example, of the WebKB and AGNews datasets, where in both the best base algorithm with 100% training were, respectively, the neural transformers XLNet and BERT.

In the Reut dataset we were able to discover an ensemble, in less time, that tie with the best base classifier. No classifier was added to the stacking combination in the Sogou and IMdB datasets, although it was possible to find the best base algorithm with less training in the case of Sogou (XGBoost + TFIDF (O)). IMdB was the only case in which we could not make a good prediction precisely by the failure in predicting, with 30% of training, that BERT would be the best algorithm when all the training data is used.

The best results with regard to MacroF1 are up to 3.23% in the case of the ACM and by approximately 1% in the 20NG and Yelp. In these datasets, we have 20.20x, 7.11x, and 1.28x runtime speedups for 20NG, ACM, and Yelp, respectively. In the Reut, WebKB, AGNews and Sogou datasets, the ensemble found by the oracle tied with the best base algorithm while reducing the execution time. There was an incredible speedup of 245.55x for WebKB, 2.09x for Sogou and 1.84x for Reut/AGNews. It is important to highlight that the computational cost reduction was high at WebKB since it is a small dataset, the smallest dataset among the ones we choose for the experiments.

Finally, when no time constraints are imposed, the oracle's prediction results are excellent: 6 wins, 1 tie and only one loss. The most relevant gains were obtained in the ACM and WebKB datasets, with a respective increase of 6.89% and 6% for MacroF1.

Dataset	Experiment	MacroF1	Combination	Most Cost
20NG	Best Base	90.58	L (Log. Reg. + MetaFeat)	
	ORQ2	▲ 91.52	ABCDHKL	D
ACM	Best Base	69.32	P (XGBoost + MetaFeat)	
	ORQ2	▲ 71.56	ABDLP	P
Reut	Best Base	47.37	P (XGBoost + MetaFeat)	
	ORQ2	● 47.37	DLP	L
WebKB	Best Base	77.76	Q (XLNet)	
	ORQ2	● 78.22	BCEFHI	I
AGNews	Best Base	93.74	R (BERT)	
	ORQ2	● 94.10	BCDFHJKLNOPQ	Q
IMdB	Best Base	34.09	R (BERT)	
	ORQ2	▼ 33.73	J	J
Sogou	Best Base	95.73	O (XGBoost + TFIDF)	
	ORQ2	● 95.73	O	O
Yelp	Best Base	65.02	R (BERT)	
	ORQ2	▲ 65.77	CJR	R

Table 4.12. Oracle results for ORQ2. Output ensemble by the Oracle using 30% of training data *versus* the best base classifier using all training data. In this scenario the ensemble generated by the Oracle it has to have a less than or equal runtime than the best base algorithm with 30% training.

It is interesting to note that, although there is no time constraints, as we are using only 30% of the data for training, it was possible to find these results at a lower computational cost than the best base algorithms in both datasets. There is a speedup of 7.11x for ACM and 4.63x for WebKB, as can be seen Figure 4.7. Oracle also achieved a MacroF1 statistical increase of 1.76% on 20NG, 1.37% for Sogou and 0.95% in AGNews. Although it seems like a relatively small gain, we must emphasize that these datasets already have high results for the metric, which makes a small statistical improvement a difficult task. In the datasets AGNews and Sogou there was a reduction, respectively, of 1.39x and 2.09x in the execution time. Only in the 20NG dataset that there was an increase in execution time, due to the inclusion of a neural network in the ensemble chosen by oracle. The last dataset in which oracle discovered an ensemble with significant gain was Yelp, with 1.15% metric increase and 1.28x of speedup.

In the IMdB dataset the oracle discovers an ensemble that presents a loss for the MacroF1, that it is explained by the same reasons as in the previous scenario: the failure to predict BERT as the future best algorithm. However, even in this case, the prediction for using algorithm J (LogisticRegression + PTE) as the sole combination (an unusual prediction) produced minimal losses: only 1.05% at a cost much smaller than using BERT. Furthermore, in Reuters, we obtain an absolute increase in MacroF1 values (6% increase), though not statistically significant due to the high variability.

Dataset	Experiment	MacroF1	Combination	Most Cost
20NG	Best Base	90.58	L (Log. Reg. + MetaFeat)	
	ORQ3	▲ 92.17	ABCDHKLR	R
ACM	Best Base	69.32	P (XGBoost + MetaFeat)	
	ORQ3	▲ 73.48	ABDLP	P
Reut	Best Base	47.37	P (XGBoost + MetaFeat)	
	ORQ3	● 50.18	DLP	l
WebKB	Best Base	77.76	Q (XLNet)	
	ORQ3	▲ 83.12	ABCEIKLMNOPQ	Q
AGNews	Best Base	93.74	R (BERT)	
	ORQ3	▲ 94.63	All	E
IMdB	Best Base	34.09	R (BERT)	
	ORQ3	▼ 33.73	J	J
Sogou	Best Base	95.73	O (XGBoost + TFIDF)	
	ORQ3	▲ 97.04	CLMOPR	R
Yelp	Best Base	65.02	R (BERT)	
	ORQ3	▲ 65.77	CJR	R

Table 4.13. Oracle results for ORQ3. Output ensemble by the Oracle using 30% of training data *versus* the best base classifier using all training data. In this scenario the ensemble generated by the Oracle has no time restrictions.

Looking at Figure 4.7, where the oracle times are shown for each scenario (ORQ1, ORQ2, ORQ3) and each dataset, we can see that in most cases the oracle predictions have strictly lower runtimes when compared to the best base algorithms with 100% of the training data. It was only in the 20NG dataset for the ORQ3 scenario that there was an increase in computational cost, which as explained above, involves the addition of a neural network in the ensemble. 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 to produce effectiveness gains, the advantage's of running the oracle's predictions in terms of cost stand for themselves.

In Table 4.14 we have the summary of results in terms of MicroF1 and MacroF1: considering all 48 results (3 research questions, 8 datasets, 2 metrics), the oracle method achieves 24 wins, 10 ties (most of them in scenarios ORQ1 and ORQ2, which can be considered good results) and only 13 losses, six of them in the IMdB dataset for the simple reason that we failed in predicting a neural network winner with fewer data. This fail is undoubtedly a point to be improved in our methodology. One idea is to look at the absolute effectiveness values with a single training point (30%) and the tendency to grow considering several points (5%, 10%, ...).

RQ	MicroF1			MacroF1		
	Win	Tie	Loss	Win	Tie	Loss
ORQ1	2	0	6	2	1	5
ORQ2	3	5	0	3	4	1
ORQ3	8	0	0	6	1	1

Table 4.14. Win/Tie/Loss summary for Oracle research questions.

4.5 Summary

This chapter answered the six research questions posed in this dissertation concerning Stacking (RQ1, RQ2, RQ3) and the Oracle (ORQ1, ORQ2, ORQ3). We started by presenting the base classifiers' classification results, highlighting the best classifiers in each dataset and respective execution times. We observed how some basic algorithms beat neural networks, especially in the mid-sized datasets. We also saw that neural networks are among the classifiers with the most wins/draws, but this comes at a high computational cost.

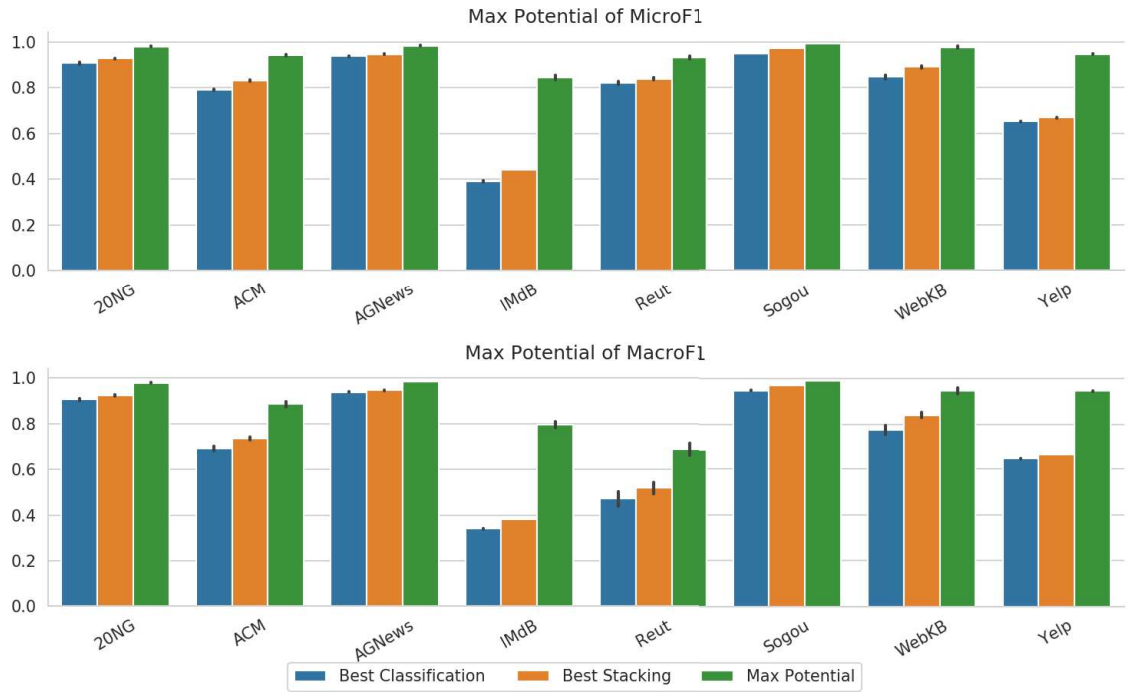

Figure 4.8. Maximum potential gain considering the 18 algorithms *versus* the results of the best base classifier (Best Classification) and the best stacking ensemble (Best Stacking) for each dataset.

We saw in the Stacking results that there are less computational costly ensembles that beat/tie the best base classifiers (RQ1) in 5 datasets when there are time constraints, how it is possible to improve effectiveness without increasing cost in almost all datasets (RQ2) and how improvements can be obtained when we completely eliminate cost constraints in the majority of the cases (7 out of the 8 datasets) (RQ3). In all the answers, we have satisfactory results that argue in favour of exploiting stacking in order to improve state-of-art results in ATC.

We finish our experimental study by analysing the performance of the Oracle, aimed at reducing computational time while keeping good effectiveness for the stacking results. We saw that it is possible to obtain less costly ensembles and have the same effectiveness of traditional stacking. Our Oracle method could produce satisfying stacking ensembles using only 30% of the data in the training phase. Our algorithm managed to deliver these results in half of the cases, ultimately demonstrating that is is possible to propose a cheap and effective way to find ensembles that beat the best base classifiers.

Finally, in our last set of experiments, our goal is to assess how far the stacking strategies presented in this work are from the effectiveness upper bound (maximal potential) considering the 18 algorithms adopted in all the above evaluations. To measure the maximal potential, for each test instance of each dataset, if one of the 18 algorithms correctly predicts its class, we assume that the ensemble strategy also correctly predicts its correct class. In that case, we are considering that the meta-layer always makes the best choices. We observe these results in Figure 4.8, for both MicroF1 and MacroF1. We also present the results achieved by the best stacking and the best individual algorithm. For all evaluated collections, we can observe that there is still large potential to be explored - in some collections (i.e., IMdB and Reut) more than others (i.e., Sogou). Despite being unrealistic to assume that meta-layer will always make the best choice, comparing all these results show us how much we can still engage efforts in this line of research, trying to improve the results obtained in this dissertation.

Chapter 5

Conclusion and Future Work

In this dissertation we present two important contributions to the application of Stacking in Automatic Text Classification (ATC): a thorough study of cost-effectiveness trade-offs of stacking 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, 8 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 before. 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 4.7–4.9). 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 the Attention-based methods BERT and XLNet for the classification task. Therefore, combinations including both of these recent and distinct paradigms (Meta-Features and Attention) for stacking were able to produce very effective results on most datasets (as shown in Table 4.13). We suggest that stacking methods should start by exploiting these two paradigms in conjunction. 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.

As future work, we will propose and evaluate different extensions for our Stacking strategy. First, we plan to add new and more recent classification algorithms proposed in the literature, such as different variations of the BERT algorithm (e.g., RoBERTa and DistilBERT Liu et al. [2019] to the Stacking. For each base algorithm considered in our stacking, we will apply recent models of classification interpretability [Lundberg

and Lee, 2017] to extract explanations related to its predictions, evaluating how much each classifier contributes to the final prediction in the meta-layer. We did not find studies in literature that explore interpretability models for stacking strategies, despite the potential of this combination to improve meta-layer decisions and, consequently, the global effectiveness of a stacking strategy. We also aim to explore multi-objective feature selection [Viegas et al., 2018] in the stacking meta-layer in order to optimize both effectiveness and computational cost. We will also exploit recent advances in the pre-processing pipeline for text classification in the meta-layer as well as other types of constraints (e.g., labeling effort) [Cunha et al., 2020b].

There is also room for improvement regarding the Oracle. For example, we can apply selective sampling [Silva et al., 2016] strategies to reduce the total training data used for the base algorithms, decreasing Oracle's computational training cost and improving the total time cost. These strategies make it possible to explore different granularities of training data reduction, according to dataset characteristics. It is important to mention that our Oracle proposal and its possible improvements have applicability in other tasks, such as Recommender Systems based on ensemble approaches [Fortes et al., 2018], a scenario we also intend to explore.

Bibliography

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

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

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238--247.

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., and Cox, D. D. (2015). Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008.

Bhatia, N. et al. (2010). Survey of nearest neighbor techniques. *arXiv preprint arXiv:1007.0085*.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135--146.

Campos, R., Canuto, S., Salles, T., de Sá, C. C., and Gonçalves, M. A. (2017). Stacking bagged and boosted forests for effective automated classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, SIGIR '17, pages 105--114, New York, NY, USA. Association for Computing Machinery.

Canuto, S., Gonçalves, M. A., and Benevenuto, F. (2016). Exploiting new sentiment-based meta-level features for effective sentiment analysis. In *Proceedings of the ninth ACM international conference on web search and data mining*, pages 53--62. ACM.

Canuto, S., Salles, T., Gonçalves, M. A., Rocha, L., Ramos, G., Gonçalves, L., Rosa, T., and Martins, W. (2014). On efficient meta-level features for effective text classification. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1709--1718. ACM.

Canuto, S., Salles, T., Rosa, T. C., and Gonçalves, M. A. (2019). 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.

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

Carvalho, J. and Plastino, A. (2021). On the evaluation and combination of state-of-the-art features in twitter sentiment analysis. *Artificial Intelligence Review*, 54(3):1887--1936.

Cavnar, W. B., Trenkle, J. M., et al. (1994). N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175. Citeseer.

Chen, T. and Guestrin, C. (2016). 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, New York, NY, USA. Association for Computing Machinery.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273--297.

Cunha, W., Canuto, S., Viegas, F., Salles, T., Gomes, C., Mangaravite, V., Resende, E., Rosa, T., Gonçalves, M. A., and Rocha, L. (2020a). Extended pre-processing pipeline for text classification: On the role of meta-feature representations, sparsification and selective sampling. *Information Processing & Management*, 57(4):102263.

Cunha, W., Canuto, S. D., Viegas, F., Salles, T., Gomes, C., Mangaravite, V., Resende, E., Rosa, T., Gonçalves, M. A., and da Rocha, L. C. (2020b). Extended pre-processing pipeline for text classification: On the role of meta-feature representations, sparsification and selective sampling. *Inf. Process. Manag.*, 57(4):102263.

Cunha, W., Mangaravite, V., Gomes, C., Canuto, S., Resende, E., Nascimento, C., Viegas, F., França, C., Martins, W. S., Almeida, J. M., et al. (2021). On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study. *Information Processing & Management*, 58(3):102481.

Dalal, M. K. and Zaveri, M. A. (2011). Automatic text classification: a technical review. *International Journal of Computer Applications*, 28(2):37--40.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Diao, Q., Qiu, M., Wu, C.-Y., Smola, A. J., Jiang, J., and Wang, C. (2014). Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14, page 193–202, New York, NY, USA. Association for Computing Machinery.

Ding, W. and Wu, S. (2020). A cross-entropy based stacking method in ensemble learning. *Journal of Intelligent & Fuzzy Systems*, pages 1--12.

Dong, Y.-S. and Han, K.-S. (2004). A comparison of several ensemble methods for text categorization. In *IEEE International Conference onServices Computing, 2004. (SCC 2004). Proceedings. 2004*, pages 419--422.

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

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871--1874.

Fortes, R. S., Lacerda, A., de Freitas, A. R. R., Bruckner, C., Coelho, D. G., and Gonçalves, M. A. (2018). User-oriented objective prioritization for meta-featured multi-objective recommender systems. In Mitrovic, T., Zhang, J., Chen, L., and Chin, D., editors, *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018*, pages 311--316. ACM.

Gomes, C., Gonçalves, M. A., Rocha, L., and Canuto, S. (2021). On the cost-effectiveness of stacking of neural and nonneural methods for text classification: Scenarios and performance prediction. *Association for Computational Linguistics, Bangkok, Thailand*.

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

Howard, J. and Ruder, S. (2018). Fine-tuned language models for text classification. *arXiv preprint arXiv:1801.06146*, pages 1--7.

Hull, D. (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.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification.

Kadhim, A. I. (2019). Survey on supervised machine learning techniques for automatic text classification. *Artificial Intelligence Review*, 52(1):273--292.

Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., and Klein, M. (2002). *Logistic regression*. Springer.

Kokkinos, F. and Potamianos, A. (2017). Structural attention neural networks for improved sentiment analysis.

Larkey, L. S. and Croft, W. B. (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, New York, NY, USA. Association for Computing Machinery.

Ledezma, A., Aler, R., Sanchis, A., and Borrajo, D. (2010). Ga-stacking: Evolutionary stacked generalization. *Intelligent Data Analysis*, 14(1):89--119.

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

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4765--4774. Curran Associates, Inc.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2017). Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825--2830.

Pelle, R., Alcântara, C., and Moreira, V. P. (2018). A classifier ensemble for offensive text detection. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, pages 237--243.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532--1543.

Reid, S. and Grudic, G. (2009). Regularized linear models in stacked generalization. In *International Workshop on Multiple Classifier Systems*, pages 112--121. Springer.

Rocha, L., Ramos, G., Chaves, R., Sachetto, R., Madeira, D., Viegas, F., Andrade, G., Daniel, S., Gonçalves, M., and Ferreira, R. (2015). G-knn: An efficient document classification algorithm for sparse datasets on gpus using knn. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, page 1335–1338, New York, NY, USA. Association for Computing Machinery.

Rokach, L. (2009). Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis*, 53(12):4046--4072.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513--523.

Silva, R. M., Gomes, G. C., Alvim, M. S., and Gonçalves, M. A. (2016). Compression-based selective sampling for learning to rank. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 247--256, New York, NY, USA. ACM.

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

Sun, C., Qiu, X., Xu, Y., and Huang, X. (2019). How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194--206. Springer.

Tang, J., Alelyani, S., and Liu, H. (2014). Data classification: algorithms and applications. *Data Mining and Knowledge Discovery Series*, pages 37--64.

Tang, J., Qu, M., and Mei, Q. (2015). *PTE: Predictive Text Embedding through Large-Scale Heterogeneous Text Networks*, page 1165–1174. Association for Computing Machinery, New York, NY, USA.

Urbano, J., Lima, H., and Hanjalic, A. (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.

Viegas, F., Rocha, L., Gonçalves, M., Mourão, F., Sá, G., Salles, T., Andrade, G., and Sandin, I. (2018). A genetic programming approach for feature selection in highly dimensional skewed data. *Neurocomputing*, 273:554--569.

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

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241--259.

Xia, Y., Chen, K., and Yang, Y. (2021). Multi-label classification with weighted classifier selection and stacked ensemble. *Information Sciences*, 557:421--442.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753--5763.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (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.

Zhang, H. (2004). The optimality of naive bayes. *AA*, 1(2):3.

Zhang, X., Zhao, J. J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626.