EMPILHAMENTO DE FLORESTAS PARA CLASSIFICAÇÃO DE DADOS RUIDOSOS E DE ALTA DIMENSIONALIDADE

RAPHAEL RODRIGUES CAMPOS

EMPILHAMENTO DE FLORESTAS PARA CLASSIFICAÇÃO DE DADOS RUIDOSOS E DE ALTA DIMENSIONALIDADE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Marcos André Gonçalves

Belo Horizonte, Minas Gerais

21 de julho de 2017

RAPHAEL RODRIGUES CAMPOS

STACKING BAGGED AND BOOSTED FORESTS FOR CLASSIFICATION OF NOISY AND HIGH-DIMENSIONAL DATA

Dissertation presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

ADVISOR: MARCOS ANDRÉ GONÇALVES

Belo Horizonte, Minas Gerais

July 21, 2017

Campos, Raphael Rodrigues

C198s Stacking Bagged and Boosted Forests for Classification of Noisy and High-Dimensional Data / Raphael Rodrigues Campos. — Belo Horizonte, Minas Gerais, 2017
xxvi, 66 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Marcos André Gonçalves

1. Computação - Teses. 2. Floresta Aleatória. 3. Aprendizado de Máquina. 4. Mineração de Dados (Computação). I. Orientador. II. Título.

CDU 519.6*82 (043)

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

FOLHA DE APROVAÇÃO

Stacking bagged and boosted forests for classification of noisy and high-dimensional data

RAPHAEL RODRIGUES CAMPOS

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
Departamento de Ciência da Computação - UFSJ

PROF. MARCO ANTÔNIO PINHEIRO DE CRISTO
Departamento de Ciência da Computação - UFAM

PROF. PEDRO OLMO STANCIOLI VAZ DE MELO
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 21 de julho de 2017.

I dedicate this dissertation to my parents, whose support and partnership have been fundamental in order for me to achieve my dreams.

Acknowledgments

Firstly, I would like to thank my advisor, Marcos André Gonçalves, for giving me the opportunity and trusting in my work, for his guidance and for the freedom I was granted throughout these years.

In alphabetical order, I would also like to express my most sincere gratitude to all my colleagues who somehow contributed to this work: Adriano Lages, Armando Honório, Bruno Laporais, Clebson Sá, Daniel Xavier, Felipe Moraes, Giuseppe Guilherme, Guilherme Gomes, Gustavo Penha, Jordan Silva, Keiller Nogueira, Luís Henrique Bicalho, Luiz Felipe, Raul Sanchez, Raquel Aoki, Sérgio Canuto, Sérgio Nunes, Thiago Salles, Vaux Gomes and Wellington Dores. We spent a long time having technical, non-technical conversations and sharing moments of relaxation, which I will certainly carry with me. Special thanks go to Sérgio Canuto and Thiago Salles who helped directly for the enrichment of this work.

I also thank all employees of DCC-PPGCC, for their dedication and competence. As well as, to CAPES for their financial support.

Even if I haven't succeeded to fully explain my research, I would like to warmly thank my parents, Rita and Mário, and my siblings, Charles, Érika, and Márcia, for their unconditional support and comfort during these years of study. My special thanks go to my mother for being a model for me of how knowledge and education can provoke a revolution on someone's fate, and my father for somehow his influence on my passion for math.

Last but not least, Marcela, I am forever grateful for your comprehension, support, and love.

"Essentially, all models are wrong, but some are useful"
(George E. P. Box)

Resumo

Floresta Aleatória (FA) é uma das estratégias mais bem-sucedidas para tarefas de classificação automática. Motivado por seu grande sucesso, recém-propostos métodos baseados em FA têm alavancado a ideia central da RF de agregar um grande conjunto de árvores de decisão com baixa correlação, que é inerentemente paralelizável e provê capacidade excepcional de generalização. Nesse contexto, esse trabalho provê várias novas contribuições para essa linha de pesquisa. Primeiramente, nós propomos uma nova estratégia baseada em FA (BERT) que aplica a técnica de *boosting* em árvores extremamente aleatórias com *bagging*. Segundo, nós demonstramos empiricamente que essa nova estratégia, assim como os recém-propostos classificadores BROOF e LazyNN_RF complementam uns aos outros, motivando-nos a empilhá-los a fim de produzir um método ainda mais eficaz. Até onde sabemos, esse é a primeira estratégia que efetivamente combina as três principais estratégias de comitê de classificadores: empilhamento, *bagging* (a base da FA) e *boosting*. Por último, nós exploramos as instâncias *out-of-bag (OOB)* para empilhar, eficientemente e sem viés, métodos baseados em *bagging*, desse modo diminuindo consideravelmente o custoso processo de treino do procedimento de empilhamento. Nossos experimentos cobrindo dois domínios ruidosos e com alta dimensionalidade - classificação de tópicos e sentimentos - provê forte evidência em favor dos benefícios de nossas soluções baseadas em FA. Nós mostramos que o BERT está dentre os classificadores de mais alta efetividade na vasta maioria dos casos analisados, mantendo os benefícios únicos da FA (interpretabilidade, paralelização, fácil parametrização, capacidade de lidar com dados heterogêneos e valores faltantes).

Palavras-chave: Empilhamento, Floresta Aleatória, Árvores Extremamente Aleatórias, Boosting, Classificação, Aprendizado Supervisionado, Aprendizado de Máquina.

Abstract

Random Forests (RF) are one of the most successful strategies for automated classification tasks. Motivated by the RF success, recently proposed RF-based classification approaches leverage the central RF idea of aggregating a large number of low-correlated trees, which are inherently parallelizable and provide exceptional generalization capabilities. In this context, this work brings several new contributions to this line of research. First, we propose a new RF-based strategy (BERT) that applies the boosting technique in bags of extremely randomized trees. Second, we empirically demonstrate that this new strategy, as well as the recently proposed BROOF and LazyNN_RF classifiers do complement each other, motivating us to stack them to produce an even more effective classifier. Up to our knowledge, this is the first strategy to effectively combine the three main ensemble strategies: stacking, bagging (the cornerstone of RFs) and boosting. Finally, we exploit the efficient and unbiased stacking strategy based on out-of-bag (OOB) samples to considerably speedup the very costly training process of the stacking procedure. Our experiments in several datasets covering two high-dimensional and noisy domains of topic and sentiment classification provide strong evidence in favor of the benefits of our RF-based solutions. We show that BERT is among the top performers in the vast majority of analyzed cases, while retaining the unique benefits of RF classifiers (explainability, parallelization, easiness of parameterization, heterogeneous data and missing value handling).

Keywords: Stacking, Random Forest, Extremely Randomized Trees, Boosting, Classification, Supervised Learning, Machine Learning.

List of Figures

List of Tables

List of Algorithms

Contents

Chapter 1

Introduction

In this chapter, we discuss the main motivations and arguments that support this dissertation. We also briefly describe our work and explicitly state our contributions.

1.1 Problem Statement

Since the advent of the Web, the amount of data available has grown unprecedentedly. Thus, organizing and extracting useful information from this enormous quantity of data has become an important (if not vital) task for industry and society. By using machine learning techniques to automatically associate documents with classes, Automatic Text Classification (ATC) provides means to organize information which allows better comprehension and interpretation of the data [Baeza-Yates and Ribeiro-Neto, 1999]. Many important applications, such as topic categorization, sentiment analysis, spam filtering, language identification, recommender systems, among others, can be effectively and efficiently solved by automatic textual classifiers. Despite the wide applicability of ATC, text classification brings its own challenges, such as **high dimensionality** and the presence of **noise** [Khan et al., 2010]. Properly handling these issues is of great importance to guarantee high classification effectiveness. This is the central topic of this work.

Several machine learning techniques aimed at tackling the challenging ATC problem have been proposed. In particular, ensembles of classifiers have been shown to excel in this situation [Salles et al., 2015; Danesh et al., 2007; Dong and Han, 2004], enjoying high effectiveness in this domain. Random Forests (RF) are one of the most successful classifier ensembles in a wide variety of classification tasks [Fernández-Delgado et al., 2014]. Despite being a classifier with great generalization power, it has been shown that RF models may suffer from overfitting issues [Segal, 2004], having its effective-

ness degraded in the presence of many irrelevant or noisy attributes—a characteristic of textual classification tasks. More precisely, it has been shown that RF classifiers whose decision trees are grown to their maximum depth are deemed to perform poorly in presence of noisy attributes. Optimistically speaking, these attributes are considerably correlated to one another or are weakly related to the outcome [Salles et al., 2017]. Particularly, when the number of attributes is large, but the fraction of relevant ones is small, random forest models tend to perform poorly. This has to do with the unnecessary variance incurred by the model, as discussed in [Hastie et al., 2009]: the bagged decision trees become plagued by irrelevant or noisy attributes, which introduces unnecessary model complexity (e.g., irrelevant classification paths). Recently, novel RF-based models were proposed to mitigate the overfitting issue faced by RF in presence of many noisy or irrelevant attributes by exploiting very distinct strategies, namely, a lazy RF version called LazyNN_RF [Salles et al., 2017], and a boosted RF strategy named BROOF [Salles et al., 2015]. Both methods learn classification models focusing on specific sub-regions of the input space, hoping to filter out irrelevant attributes and data—the primary factors that contribute to RF's tendency to overfit.

The LazyNN_RF is a lazy learner that utilizes the k nearest training instances to the test example as a projected training set, which is given to a Random Forest classifier for training. Subsequently, the trained RF classifier is used to predict the test example class. Salles et al. [2017] showed that projecting the training set of RF classifier to the neighborhood of test data can mitigate the overfitting problem suffered by random forests when dealing with data with many irrelevant or noisy attributes, and provides more generalization power to the RF classifier. The BROOF is a method based on the well known and successful technique called boosting. It focuses on hard-to-classify regions of the input space, by means of the combination of boosting technique and the exploitation of the out-of-bag (OOB) error, promptly generated by RF classifier at training time. Salles et al. [2015] showed that restricting the influence of training data to the hard-to-classify sub-regions of the input space, by means of a smooth combination of random forests and boosting, can mitigate the overfitting issue and leverage classification effectiveness on new unseen data.

1.2 Our Proposal

In this work, we advance the state-of-the-art in text classification by proposing a novel derivation of the RF classifier. More specifically, we propose a new boosted version of the RF classifier, based on some ideas explored by the BROOF classifier: the so-

called Boosted Extremely Randomized Trees (BERT) classifier. While BROOF is able to mitigate the overfitting issue faced by RF classifier when applied to high-dimensional noisy data, by avoiding the generation of overly complex trees, it offers limited capability of bias reduction (through the so-called selective out-of-bag based weight update strategy) as shown by Salles et al. [2017]. Thus, bias may still pose as an important factor to contribute to the error rate. To tackle this potential issue, we here propose to introduce another source of randomization in the boosted strategy proposed by Salles et al. [2015] in order to achieve a better bias-variance tradeoff, by building tree in a more extreme fashion as proposed by Geurts et al. [2006]. This novel strategy has the following motivations: (*i*) we expect to avoid overly complex models (and thus mitigate overfitting) through the application of the BROOF-like strategies; (*ii*) to provide more control over the learner's bias through the additional randomization offered by building extremely randomized trees [Geurts et al., 2006]; and, finally, (*iii*) to exploit the fact that the extremely randomized trees have shown to be more robust to noise than the RF classifier.

Moreover, motivated by the fact that distinct learning methods may complement each other, uncovering specific structures that underlie the input/output relationship of the data at hand, in this work we also propose to exploit the complementary characteristics of the recently proposed RF-based approaches and ours, by stacking them in order to learn an even more effective meta-classifier. As we shall see later, their level of disagreement is high, which motivates our idea. Up to our knowledge, this is the first attempt to combine the three main ensemble strategies: bagging, boosting and stacking.

Finally, when stacking classifiers, one usually relies on k fold cross-validation procedures to estimate the *a posteriori* class probabilities for each example, to serve as input for the meta-classifier. Based on these predicted *a posteriori* class distribution estimates, the meta-classifier induces a relationship between these predictions and the true class. However, such estimation strategy may be very costly and sometimes ineffective, since it depends on learning k different models to estimate the probability distributions that serve as input for the stacking procedure. In order to cope with this problem, we here propose to exploit the efficient and unbiased out-of-bag (OOB) error estimate, an out-of-the-box estimate naturally produced by the bootstrap procedure used in each RF-based learner. Thus, we avoid additional computation efforts to learn a stacked classifier.

The main research questions answered in this dissertation are:

- **RQ1**: Can we further improve the generalization capabilities of BROOF algo-

rithm when applied to high-dimensional data by means of additional source of randomness?

- **RQ2** Can we effectively and efficiently combine Random Forests based methods, such as LazyNN_RF, BROOF and BERT, in order to leverage their predictive performance in high-dimensional data with many noise/irrelevant attributes?

1.3 Main Contributions

In summary, the main contributions of this work are:

1. The proposal of a novel RF-based classifier, named BERT, that is able to outperform state-of-the-art classifiers;

2. The proposal of a new stacking classifier that exploits the complementary characteristics of BROOF, LazyNN_RF and BERT that is able to outperform all analyzed classification algorithms, including a stacking of traditional methods, often by large margins;

3. The proposal of a new estimation strategy based on the use of OOB for generating the input for the stacked meta-classifier that substantially reduces the computational effort/runtime of the stacking strategy while retaining its predictive power;

4. A parallel version of LazyNN_RF algorithm which exploits the massively parallel power of Graphical Processing Units (GPUs);

5. Extensive experimentation with 15 datasets in two domains – topic categorization and sentiment analysis; – against several baselines including traditional classifiers (to compare with BERT), several stacking combinations (to compare with the stacking of Forests) and several state-of-the-art stackers (to compare with our OOB-based approach).

1.4 Publications

The contributions obtained with the development of this master's dissertation were published by means of publication in national and international conferences and proceedings. Next, it is presented a complete list of the published papers:

- Raphael Campos, Sérgio Canuto, Thiago Salles, Clebson C. A. de Sá and Marcos André Gonçalves. 2017. Stacking Bagged and Boosted Forests for Effective Automated Classification. In Proceedings of SIGIR '17, Shinjuku, Tokyo,Japan, August 07-11, 2017, 10 pages - [Campos et al., 2017].

- Campos, R. R. and Gonçalves, M. A. (2016). Bert: Melhorando classicação de texto com árvores extremamente aleatórias, bagging e boosting. In 31st of the Brazilian Symposium on Databases - [Campos and Gonçalves, 2016] (***Honorable Mention for Best Short Paper***).

- R R. Campos, M A. Gonçalves and T. Salles (2016). Quando a Amazônia Encontra a Mata Atlântica: Empilhamento de Florestas para Classificação Efetiva de Texto. 4th Symposium on Knowledge Discovery, Mining and Learning - [Campos et al., 2016].

1.5 Outline

The remainder of this work is described as follows:

- In Chapter 2, we review and highlight some relevant learning algorithms found in literature which are the base of our work.

- In Chapter 3, we detail the experimental workload used in our analysis. We provide a description of the six reference datasets regarding topic categorization and the ten reference datasets regarding sentiment analysis. Then, we briefly describe the text classification algorithms and the metrics used to compare their effectiveness.

- In Chapter 4, we detail a lazy version of Random Forest classifier proposed by [Salles et al., 2017], and propose to scale the algorithm performance by taking advantage of the massively parallel power of Graphical Processing Units (GPUs) through the GTkNN algorithm [Canuto et al., 2015].

- In Chapter 5, we detail our proposed BERT classifier, a boosted version of bagged extremely randomized trees, providing its motivations, learning strategy details, and extensive experimental evaluation.

- In Chapter 6, we detail our proposed method to efficiently stack RF-based methods as well as its experimental evaluation and analysis.

- Finally, in Chapter 7, we conclude the our work, pointing out some possible directions for further investigation.

Chapter 2

Background

For the sake of this dissertation, we devote this chapter to recall important concepts and highlight algorithms which are the base of our work.

2.1 Supervised Learning

Automatic data classification is key to solve effectively a broad variety of practical problems, being of great value for industry and society, mainly when a traditional (purely specified by well defined algorithmic steps) solution is not viable. Automatic classifiers have become fundamental to enhance and support several distinct tasks, such as automatic text classification, organizing digital libraries, automatically tagging topics, filtering spam on email systems, identifying the writing style of textual data, supporting diagnosis in health care systems, recognizing handwriting input, recognizing objects in images, analysing micro-arrays, to name few. In all such cases, it is difficult to conceive a set of rules to effectively solve the problem under consideration without looking into previously observed data related to it. Furthermore, such set of rules generally does not cover all possible cases, limiting its discriminative power. Hence, a rule-based system more precise requires, due to the adaptive behavior of the data, that one continuously expresses more rules, which may rapidly leads to an overwhelm of exceptions. Worse still, such set of rules may not capture latent relationships and may change as time goes by, harming the predictive power of such systems. Evidently, such problems require more sophisticated solutions, capable of recognizing patterns from observed data to adequately categorize the unobserved one, which is what supervised learning methods do [Salles et al., 2017].

More formally, supervised learning entails learning a mapping between a set of input variables and an output variable and applying this mapping to predict the out-

puts for unseen data [Cord et al., 2008]. Given a set of N examples of the form $\{(X_1, y_1), ..., (X_N, y_N)\}$, known as training set, where X_i denotes the vector representation of the i-th instance and $y_i \in \mathbb{Y}$ is a categorical attribute indicating the class of the i-th instance. The main objective of a supervised learning algorithm is to learn an approximation of the unknown class *a posteriori* probability distribution $P(y_i|X_i)$, which underlies the relationship between data points and their associated classes, based on the observed data. There exist two main approaches to achieve that, one based on the direct estimation of $P(y_i|X_i)$, and another based on the indirect estimation of $P(y_i|X_i)$. The former approach, so called *discriminative* classifier, learns a direct map $f : \mathbb{R}^n \rightarrow \mathbb{Y}$, from the observed inputs X_i to the output class y_i, by minimizing an effectiveness metric (e.g., error rate), without making any assumption regarding the probability density function for each class. On the other hand, the latter, which is known as *generative* classifiers, learns the joint distribution $P(X_i, y_i)$, of the inputs X_i and the class y_i, and thus make their predictions by using Bayes rules to estimates the posterior distribution $P(y_i|X_i)$.

Finally, one can group supervised learners into two main categories, according to the way they yield the prediction model, namely, eager and lazy learners. Eager learners build a single model, given the entire training set, which is used to classify all unseen data presented to the classifier. In contrast, lazy learning methods simply store the training set and, thus, postpone the generation of the model until it is given a test example. Given a test instance x, they select training examples whose patterns are considered more appropriate to discriminate x's class, according to some distance or similarity function.

2.2 Decision Trees

Decision trees (DTs) are non-parametric supervised learning methods, very popular in a wide range of classification and regression tasks. The general idea of tree-based methods is to recursively partition the input space into a set of hyper-rectangles, in each of which is fitted a simple model, such as a constant. They are conceptually simple yet powerful [Hastie et al., 2009]. Their popularity is mainly due to its unique characteristics: (i) easy explainability and interpretability, since trees can be easily visualised; (ii) requires small effort on data preparation, tree-based methods usually do not require data normalisation and can handle missing values, and both categorical and numerical features; (iii) logarithmic time complexity to predict data; (iv) capable of handling multi-output problems.

There are several popular algorithms to build tree-based learning models, namely CART [Breiman and Olshen, 1984], ID3 [Quinlan, 1986] and C4.5 [Quinlan, 1996]. In this work, we focus on CART algorithm, since several freely available implementation of it can be found in machine learning libraries such as the well known *scikit-learn* library [Pedregosa et al., 2011] written in python, *rpart* [Therneau et al., 2015] library in R, and *Weka* in Java [Hall et al., 2009].

Algorithm 1 Decision Tree pseudo-code

1: **function** TRAIN($\mathbb{D}_{train} = \{(X_i, y_i)|_{i=1}^{N}\}$)
2: $n_{root} \leftarrow Node(\mathbb{D}_{train})$
3: $h_i \leftarrow BuildTree(n_{root}, \mathbb{F})$ ▷ $\mathbb{F}$ is the feature set
4: **end function**
5: **function** BUILDTREE($node$, $\mathbb{F}$)
6: **if not** $StoppingCriterion(node)$ **then**
7: $f_{max} \leftarrow \arg_f \max InformationGain(\mathbb{F})$
8: $cutpoint \leftarrow FindCutPoint(f_{max}, node)$
9: $node.left \leftarrow BuildTree(Node(\{(X_i, y_i)|X_i^{f_{max}} \leq cutpoint\}), \mathbb{F})$
10: $node.right \leftarrow BuildTree(Node(\{(X_i, y_i)|X_i^{f_{max}} > cutpoint\}), \mathbb{F})$
11: **end ifreturn** $node$
12: **end function**

Classification and regression tree (CART) recursively splits the input space by making orthogonal divisions considering an univariate splitting criteria of impurity [Breiman and Olshen, 1984]. Finding the best binary partition in terms of some impurity measure, such as Gini Index, Entropy or misclassication error, may be computationally infeasible. Hence, a greedy algorithm is usually performed in order to choose the best attribute to split and its cut-point, shown in the Algorithm 1 (lines 7-8). Afterwards, the data is divided into two disjointed regions accordingly to the obtained split variable and point. This process is repeated recursively on all resulting regions until a stop criterion is reached (i.e., tree depth, the total number of leaves) or all the leaves are pure, which implies in a tree grown to its maximum depth. A fully grown decision tree may overfit the data, thus degrading its predictive performance on new unseen data, which can be mitigate by pruning it, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree in order reduce the model complexity. Other drawbacks faced by decision trees are: (i) instability (high variance), small variations in the data may result in a very distinct tree being generated; and (ii) overly-biased trees in presence of heavily imbalance classes. All these issues (including overfitting) can be mitigate by training multiple trees in an ensemble learner, as we shall discuss in the subsequent section. For further information

on the subject, we refer the interested reader to [Breiman and Olshen, 1984; Quinlan, 1986, 1996; Louppe, 2014]

2.3 Ensemble Methods

Build a single model that covers the entire instance space may not be optimal since it may not be able to capture all nuances of the data. In fact, different models/algorithms may cover distinct regions and aspects of the input space complementing one another predictions [Kuncheva and Whitaker, 2003].

Boosting and Random Forests, two well-known and successful learning algorithms, are based on the idea of building a strong model by means of combining multiple decision trees, each of which is built by somehow disturbing the training set, and subsequently, averaging the predictions in order to come up with a final decision.

In the subsequent subsections we dive into these methods, discuss how they fix the aforementioned drawbacks of decision trees and outline their unique features.

2.3.1 Random Forests

Proposed by Breiman [2001], the Random Forests (RF) classifier has been one of the most successful classifiers in an enormous variety of automatic classification tasks [Fernández-Delgado et al., 2014] comparable, and sometimes superior, to Support Vector Machine (SVM). A fundamental aspect that guarantees the high effectiveness of RF classifier is the large set of low-correlated trees composing the forest, which is obtained by disturbing the data with series of random procedures, such as bagging of the training set and random attribute selection drawn from a randomly chosen subset of features. It also important to build trees with prediction capabilities better than random guessing, which is typically achieved by growing them to their maximum depth (low bias). By considering the average of several decorrelated trees as the model prediction, it can be shown that the RF classifier reduces variance while keeping its bias unchanged [Breiman, 2001]. It typically brigs up more stable models with higher generalization capabilities.

More specifically, each of decision tree is grown as follows: Firstly, the bagging (**b**ootstrap **agg**regat**ing**) procedure is performed (line 3). Introduced by Breiman [1996], bagging is an ensemble method which aims to control variance by creating several versions of the classifier and average them. It is specially suitable for estimators that has high-variance and low-bias (e.g., decision trees grown to its maximum depth). Thus, given a training set $\mathbb{D}_{train}$, the bootstrap procedure yields M training

Algorithm 2 Random Forest pseudo-code

1: **function** Train($\mathbb{D}_{train} = \{(X_i, y_i)|_{i=1}^{N}\}$, M)
2: **for** each $m \in \{1, .., M\}$ **do**
3: $\mathbb{D}_{train}^{m} \leftarrow Bootstrap(\mathbb{D}_{train})$ ▷ $\mathbb{D}_{train}^{m} \in \mathbb{D}_{train}$
4: $\mathbb{F}_m \leftarrow RandomSubspace(\mathbb{F})$ ▷ $\mathbb{F}$ is the feature set
5: $n_{root} \leftarrow Node(\mathbb{D}_{train}^{m})$
6: $h_m \leftarrow BuildTree(n_{root}, \mathbb{F}_m)$ ▷ Algorithm 1
7: **end for**
8: **end function**
9: **function** Classify(X, $h_m|_{m=1}^{M}$)
10: **return** $\arg_c \max \sum_{m=1}^{M} h_m(X)$
11: **end function**

sets $D_{train}^{i}|_{i=1}^{M}$ by sampling with replacement from the original data. Second, each of the M trees $h_i|_{i=1}^{M}$ is trained considering these newly created training sets and a feature subset $\mathbb{F}_i$ sampled from the original feature set $\mathbb{F}$ without replacement (line 4), where $|\mathbb{F}_i| << |\mathbb{F}|$. Finally, the final prediction is given by majority voting h_i, $1 \leq i \leq M$ (line 10). The Algorithm 2 summarizes the whole process.

In addition to its high generalization capability in several application domains, the Random Forests classifiers also has some interesting characteristics which may be very useful in determined situations. One of such interesting characteristics is the so-called Out-of-Bag estimate error. When building ensemble models based on bootstrapped samples, such as Bagging or Random Forests, it is possible to use the left-out samples $\mathbb{D}_{oob}^{m} = \mathbb{D}_{train} \setminus \mathbb{D}_{train}^{m}$ to estimate important statistics, such as the generalization error. More precisely, since each tree in the Random Forests is trained with bootstrapped samples, approximately $e^{-1} \approx 37\%$ of the original training set is left aside (out-of-bag) [Hastie et al., 2009] and may be used as an independent validation set in order to estimate its individual prediction [Wolpert and Macready, 1999]. The RF's Out-of-Bag error estimate is formally defined by:

$$Err^{OOB} = \frac{1}{N} \sum_{(x_i, y_i) \in \mathbb{D}_{train}} L(\arg_c \max \sum_{m=1}^{M} h_m(x_i) I[x_i \in \mathbb{D}_{oob}^{m}], y_i), \tag{2.1}$$

where I denotes an indicator function that returns 1 when the m-th tree did not use x_i as training instance, 0 otherwise, and L is a given loss function with the predicted class $\hat{y}_i$ and the actual one y_i as parameters. Hence, the Random Forests' out-of-bag prediction estimate $\hat{y}_i^{oob}$ at (x_i, y_i) is obtained by the majority vote of the prediction estimation of the trees which did not use (x_i, y_i) for training ($\arg_c \max \sum_{m=1}^{M} h_m(x_i) I[x_i \in \mathbb{D}_{oob}^{m}]$). Thus, the RF's Out-of-Bag error estimate

Err^{OOB} is defined by the average loss over all training examples, using $\hat{y}_i^{oob}$ as the prediction in order to estimate the loss for a given (x_i, y_i). The out-of-bag error estimate is an unbiased estimate of the generalization error of the ensemble model, providing statistics comparable, and sometimes more accurate, to K-fold cross-validation [Wolpert and Macready, 1999].

Another RF's interesting feature is the variable importances. In several tasks it is important to identify the input variable that are the most discriminative in order to have a deeper comprehension of the problem under consideration. Random forests provide means for assessing the importance of an input variable, and consequently boost the interpretability of the ensemble model [Louppe, 2014]. There are other characteristics, such as proximity measures, missing data handling, which also have useful applications. However, since they are not closely related to this dissertation, we refer the reader to [Breiman, 2001; Hastie et al., 2009; Louppe, 2014].

2.3.2 Boosting

Boosting is a sequential meta-algorithm which trains several "weak learners" (that is, classifiers capable of yielding predictions slightly better than random guessing) in order to generate a more precise model. For each iteration i of the boosting procedure, let Δ^i be a probability distribution over the training set of size N. In the first iteration $i = 1$, all instances have equal probability $\Delta^1(j) = \frac{1}{N}$, $\forall j|_{j=1}^{N}$. For each successive steps, the instances weights are revised and the learning algorithm is retrained on the weighted observations. At iteration $i > 1$, each example x_j in the training set, which were misclassified, has its weight incremented so that, in the succeeding iteration, an updated distribution Δ^{i+1} is considered, which emphasizes the misclassified instances (e.g, the hardest to classify ones). Each successive "weak learner" learns by focusing on those hard-to-classify regions of the input space. Thus as specialized classifiers are built upon these hard-to-classify regions, the bias tends to minimize.

The classical boosting algorithm, called AdaBoost [Freund and Schapire, 1997], is summarized in the Algorithm 3. The current model h_m is induced on the weighted observations (line 4). At line 5, weighted error rate is computed. Subsequently, the weight α_m is calculated at line 6 and the weights of each of the instances are updated for the following iteration (7). Instances incorrectly classified by h_m have their weights scaled by a factor e^{α_m}, increasing their relative influence for inducing the succeeding learner h_{m+1}. The ensemble prediction is assessed by means of weighted majority vote (line 11).

Despite being able to leverage dramatically the effectiveness of tree-base methods,

Algorithm 3 AdaBoost Pseudo Code

function Train($\mathbb{D}_{train} = \{(X_i, y_i)|_{i=1}^{N}\}$, M)
  $w_1 \leftarrow \frac{1}{|\mathbb{D}_{train}|}$
  for each $m \in \{1, .., M\}$ **do**
    $h_m \leftarrow C(\mathbb{D}_{train}, w_m)$
    $err_m \leftarrow \frac{\sum_{i=1}^{|\mathbb{D}_{train}|} w_m^i I[y_i \neq \hat{y}_i]}{\sum_{i=1}^{|\mathbb{D}_{train}|} w_m^i}$
    $\alpha_m \leftarrow \log(\frac{1-err_m}{err_m})$
    $w_{m+1}^i \leftarrow \frac{w_m^i e^{\alpha_m I[y_i \neq \hat{y}_i]}}{\mathbb{Z}}$, where $\mathbb{Z}$ is a normalizing constant
  end for
end function
function Classify(X, $(\alpha_m, h_m)|_{m=1}^{M}$)
  return $\arg_c \max \sum_{m=1}^{M} \alpha_m h_m(X)$
end function

and considered one of the most successful and powerful learning algorithms, boosting may have its predictive performance hampered when in presence of noisy data since later iterations may over-emphasize instances which are noise (hence yielding extremely ineffective classifiers)[Freund and Schapire, 1996]. This specially pertinent to Boosting since its probability update rule tends to lead the "weak-learners" to hard-to-classify regions that may be plagued by noise instances, leading to sub-optimal decision boundaries. We revisit this subject in Chapter 5. For further information on Boosting, we refer the interested reader to [Freund and Schapire, 1996, 1997; Hastie et al., 2009; Salles et al., 2015, 2017]

Chapter 3

Experimental Workload

In this chapter, we present the experimental workload used in our analysis. We cover two important text classification domains, namely, topic categorization and sentiment analysis. While the former deals with the task of assigning a label to textual documents in an automatic fashion, the later deals with the task of predicting the sentiment underlying a textual passage. We provide a description of the six reference datasets regarding topic categorization and the ten reference datasets regarding sentiment analysis in Section 3.1. Then, we briefly describe the analyzed text classification algorithms in Section 3.2. Finally, in Section 3.3 we describe the metrics used to compare the effectiveness of the baselines.

3.1 Datasets

One of the challenges when categorizing textual data into topics is that textual documents are usually represented by a great amount of features (high dimensionality) and most of them could be irrelevant or noisy [Khan et al., 2010]. However, despite being a challenging application domain, it is of great importance nowadays, due to its wide applicability and demand. Here, we considered both scientific and news articles, organized in the following datasets:

4 Universities (4UNI), a.k.a, WebKB contains Web pages collected from Computer Science departments of four universities by the Carnegie Mellon University (CMU) text learning group[1]. There is a total of $8,277$ web pages, classified in 7 categories (such as student, faculty, course and project web pages).

[1] `http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/`

20 Newsgroups (20NG) contains 18,805 newsgroup documents, partitioned almost evenly across 20 different newsgroups categories[2]. 20ng has become a popular dataset for experiments in text applications of machine learning techniques, such as text classification and text clustering.

ACM-DL (ACM) a subset of the ACM Digital Library with 24,897 documents containing articles related to Computer Science. We considered only the first level of the taxonomy adopted by ACM, whereas each document is assigned to one of 11 classes.

Reuters (REUT90) is a classical text dataset, composed by news articles collected and annotated by Carnegie Group, Inc. and Reuters, Ltd. We consider here a set of 13,327 articles, classified into 90 categories.

Spambase (SPAM) is a collection of spam e-mails collected by Hewlett-Packard Labs. It contains 1,813 e-mails labeled as spam and 2,788 non-spam personal e-mails.

MEDLINE (MED) a subset of the MedLine dataset, with 861,454 documents classified into 7 distinct classes related to Medicine. This dataset was obtained from Rocha et al. [2008]. In that work the authors considered the first level of the taxonomy so that each document article is classified under only one category, avoiding dealing with multilabel cases.

UniRCV1 The Reuters Corpus Volume 1 (RCV1) is a dataset with 804, 427 English language news stories. We considered the complete topics taxonomy comprised of 103 classes. However, the original RCV1 is a multi-label dataset with the multi-label cases needing special treatment, such as score thresholding, etc. (see [Lewis et al., 2004] for details). As our current focus is on unilabel tasks, to allow a fair comparison among the other datasets (which are also unilabel) and all baselines (which also focus on unilabel tasks), we decided to remove the documents assigned to more than one class from RCV1, deriving a new dataset which we call UniRCV1. This collection has 101 classes and about 20% less documents. Nevertheless, as we shall see, the effectiveness levels obtained by our method and the best baselines are still compatible with those of the original multi-label RCV1

The details regarding each topic categorization dataset (size, number of features and class distribution) can be found in Table 3.1.

[2] `http://qwone.com/~jason/20Newsgroups/`

			Class Distribution						
Dataset	Size	# Features	# Classes	Mean	Minor Class	1st Quartile	Median	3rd Quartile	Major Class
SPAM	4601	57	2	230	1813	1815	1818	1820	2788
4UNI	8274	40195	7	1182	137	343	929	1382	3757
20NG	18766	61050	20	938	627	952	978	988	998
ACM	24897	59990	11	2263	63	761	2041	3278	6562
REUT90	13327	19590	90	148	2	8	29	91	3964
MED	861454	268783	7	123065	1843	36196	44089	143568	455994
UniRCV1	652909	46120	101	6464	3	401	1646	6725	62943

Table 3.1: Text Categorization: Statistics Summary for each Dataset

Regarding sentiment analysis, we use a collection of ten publicly available sentiment benchmarks of message that alludes to how much a user-generated content express a positive or negative sentiment about a subjects. We consider messages from several domains, such as reviews, posts on social networks, user comments and snippets of opinion news.

Amazon consists of a set of product reviews form *amazon.com*.

BBC a set of messages from comments in the BBC and Runners World forum from SentiStrength research [Thelwall et al., 2012].

Debate consists of tweets about the 2008 U.S. Presidential debate.

Digg user provided comments on web content aggregated in *digg.com*.

MySpace a set of messages crawled from the Myspace network, used in SentiStrength research.

NYT includes sentence-level snippets from a set of New York Times opinion editorials.

Tweets a set of tweets from VADER work [Hutto and Gilbert, 2014] which were crawled from Twitter's public timeline (with varied times and days of posting).

Twitter this dataset consists of human labeled messages used in the SentiStrength research.

Yelp consists of a set of business and services reviews from the greater Phoenix, AZ metropolitan area.

Youtube a set of user provided comments on video content.

The details regarding each sentiment analysis dataset (size, number of features and class distribution) can be found in Table 3.2.

In all cases, we performed a traditional preprocessing task that consists of removing stopwords (using the standard SMART list) and applying a simple feature

			Class Distribution						
Dataset	Size	# Features	# Classes	Mean	Minor Class	1st Quartile	Median	3rd Quartile	Major Class
Amazon	3610	3678	2	1800	1482	1644	1805	1966	2128
BBC	752	5655	2	376	99	238	376	514	653
Debate	1979	3363	2	990	730	860	990	1119	1249
Digg	782	4015	2	391	210	300	391	482	572
NYT	4946	8756	2	2473	2204	2338	2473	2608	2742
Myspace	834	2639	2	417	132	274	417	560	702
Tweets	4196	7346	2	2098	1299	1698	2098	2498	2897
Twitter	2289	7777	2	1145	949	1047	1144	1242	1340
Yelp	5000	19398	2	2500	2500	2500	2500	2500	2500
Youtube	2432	6275	2	1216	767	992	1216	1440	1665

Table 3.2: Sentiment Analysis: Statistics Summary for each Dataset

selection procedure, removing terms with low "document frequency (DF)"[3]. Regarding term weighting, we tested TF, TFIDF and L2 normalization schemes, choosing the best strategy for each classification approach. Particularly, we use TF for all classifiers based on RF and Naive Bayes, and TFIDF with L2 normalization for both SVM and kNN.

3.2 Baselines

Next we provide a brief description of the learning methods used as baselines for our analysis.

Support Vector Machine (SVM) searches for the optimal hyperplane that splits the positive from the negative class, by maximizing the margin between the closest points from either class. SVM is inherently a binary classifier which implies in adopting approaches, such as one-versus-one or one-versus-all, in order to adapt binary SVM for multi-class classification tasks. Linear SVM is specially popular for text classification problems, since it is robust to high-dimensional data, being a top performer in such scenarios.

Naive Bayes (NB) is a generative learning strategy that applies the Bayes's theorem with the strong assumption that features are independent. Despite this assumption is rather optimistic and generally not true, NB classifiers usually outperform more sophisticated approaches in practice. Its feature independence assumption may be specially appropriate when applied to high-dimensional data since each distribution can be independently estimated as a one dimensional distribution. We adopt here the Multinomial Naïve Bayes approach, since it is well-accepted for text classification tasks.

k-Nearest Neighbors (kNN) is, perhaps, the most well-known and broadly

[3]We removed all terms that occur in less than six documents (i.e., DF<6).

used lazy algorithm. In this method, the majority class of the training instances composing the neighborhood of the test instance x is used to assess its class label. The neighborhood of the new unseen data x is defined by the set of k instances closest to x in the input space, where closeness implies a metric, such as Euclidean distance or cosine similarity. The rationale here is that, based on the contiguity hypothesis, we expect a test instance x to have the same classification as the training instances located in the neighborhood of x. The free parameter here is the number of neighbors, k.

Random Forests (RF) is an ensemble of low-correlated decision trees constructed by series of random procedures, such as bagging of the training set and random attribute selection drawn from a randomly chosen subset of features. These large set of trees with reduced correlation are one of the key aspects that guarantee the high effectiveness of RF classifier [Breiman, 2001]. Another fundamental aspect is the necessity of building trees with prediction capabilities better than random guessing, which is typically achieved by growing them to their maximum depth. By considering the average of several decorrelated trees as the model prediction, it can be shown that the RF classifier reduces variance while keeping its bias unchanged, which typically brings up better models, with higher generalization power [Salles et al., 2017]. The RF's free parameters are the number of sampled features and the number of composing trees.

Lazy Nearest Neighbor Random Forest (Lazy) attempts to mitigate the overfitting issue observed when RF is applied to problems with many noisy attributes, by restricting the training set to be composed by observations in the input space that exhibit higher similarity with the test instance. The projection of training set by means of k-nearest neighbor contributes to a more robust and precise classifier [Salles et al., 2017]. Such a localized training set, composed of instances which are similar to the test instance, filters out potentially noise/irrelevant data and reduces the impact of noisy features in the classification model. Moreover, such a projected space allows the model to remove unnecessary bias through a smaller and more localized training set, while also reducing the variance by means of the traditional RF's model averaging procedure [Salles et al., 2017].

Extremely Randomized Trees (Extra-Trees) are an ensemble of trees similar to the RF however, they have two key differences: (i) Extra-Trees do not apply the bagging procedure to construct a set of the training samples for each tree. Hence, the same input training set is used to train all trees; (ii) Extra-Trees pick a node split very extremely (both a variable index and variable splitting value are chosen randomly), whereas Random Forest finds the best split (optimal one by variable index and variable splitting value) among random subset of variables. They have same free parameters as RF.

BROOF combines boosting and bagging by exploiting RF as "weak learners" in a boosting framework. The boosting update rule in BROOF uses the out-of-bag (OOB) samples, promptly generated by the RF classifier at training time, as an unbiased error estimate to drive the boosting iterations and "smoothly" reweights just the OOB instances. Salles et al. [2015] show by an extensively experimental workload (contrasted to up to ten state-of-the-art classifiers, covering almost 500 results) that restricting the influence of training data to the hard-to-classify sub-regions of the input space, by means of a smooth combination of random forests and boosting, can mitigate the overfitting issue observed when learning decision trees composing the ensemble classifier, thus leveraging classification effectiveness on new unseen data.

We use the scikit-learn implementation[4] of linear SVM, k-Nearest Neighbors (kNN), Multinomial Naïve Bayes (NB), Random Forests (RF) and Extremely Randomized Trees (Extra-Trees). We use our own implementation of the RF-based methods, namely, BROOF, the lazy version of the RF classifier (LAZY) and the lazy version of extra-trees (LXT), since there is no freely available implementation for these classifiers.

The free parameters of these classifiers include the cost C (for SVM), neighborhood size k (for KNN and LAZY) and the number of features considered in the split of a node on the RF-based approaches. These free parameters were set using **5-fold cross-validation** within the training set. For the RF-based approaches, each tree is grown without pruning, as suggested by Hastie et al. [2009], and since the results obtained with 200, 300 and 500 trees are statistically tied (with 95% confidence), we adopted 200 trees due to the lower cost. Concerning the BROOF classifier, we use 8 weak learners, setting the maximum number of iterations to 200, as suggested by Salles et al. [2015]. We use the same parameters for the proposed BERT method.

We would like to point out that some of the results obtained in some datasets may differ from the ones reported in other works for the same datasets. Such discrepancies may be due to several factors such as differences in dataset preparation[5], the use of different splits of the datasets (e.g., some datasets have "default splits" such as REUT and 20NG[6]) We would like to stress that we ran all alternatives under the same conditions in all datasets, using the best traditional feature weighting scheme, using standardized and well-accepted cross-validation procedures that optimize parameters for each of alternatives, and applying the proper statistical tools for the analysis of the results.

[4]Available in `http://scikit-learn.org/`

[5]For instance, some works do exploit complex feature weighting schemes or feature selection mechanisms that do favor some algorithms in detriment to others.

[6]We believe that running experiments only in the default splits is not the best experimental procedure as it does not allow a proper statistical treatment of the results.

3.3 Evaluation Metrics

The explored classifiers were compared using two standard information retrieval measures: *micro averaged* F_1 (MicroF_1) and *macro averaged* F_1 (MacroF_1) [Lewis, 1995]. Let $C = \{c_1, c_2, ..., c_k\}$ be the set of class of a given collection. For each class c_i, we can define the following values:

- True positives for c_i (TP_i): the number of test documents that the true class is c_i and that were classified as c_i
- True negatives for c_i (TN_i): the number of test documents that the true class is not c_i and that were not classified as c_i
- False positives for c_i (FP_i): the number of test documents that the true class is not c_i and that were classified as c_i
- False positives for c_i (FN_i): the number of test documents that the true class is c_i and that were not classified as c_i

Recall and precision can be assessed per class or globally. The recall $r(c_i)$ for a given class c_i is defined as $r(c_i) = \frac{TP_i}{TP_i+FN_i}$, which means the fraction of test documents of class c_i that were correctly classified. The precision $p(c_i)$ for a given class c_i is given as $p(c_i) = \frac{TP_i}{TP_i+FP_i}$, which means the fraction of documents correctly classified from all documents attributed to the class c_i. The global recall and precision are respectively defined as $r(C) = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k}(TP_i+FN_i)}$ and $p(C) = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k}(TP_i+TN_i)}$. The F_1 measure combines the recall and precision metric by means of their harmonic mean. While the MicroF_1 measures the classification effectiveness overall decisions (i.e., the pooled contingency tables of all classes), which is defined as:

$$MicroF_1 = 2\frac{p(C)r(C)}{p(C)r(C)} \tag{3.1}$$

The MacroF_1 measures the classification effectiveness for each individual class and averages them, as follows:

$$MicroF_1 = \frac{\sum_{i=1}^{k} 2\frac{p(c_i)r(c_i)}{p(c_i)r(c_i)}}{k} \tag{3.2}$$

MicroF_1 tends to be dominated by the classifier's performance on more frequent classes and the MacroF_1 is more influenced by the performance on rare ones.

To compare the average results of our 5-fold cross-validation experiments, we assess their statistical significance by applying a paired t-test with 95% confidence and

Bonferroni correction to account for multiple comparisons. This test assures that the best results, marked in **bold**, are statistically superior to others (up to the chosen confidence level).

Chapter 4

Lazy Random Forests

In this chapter, we detail the lazy version of Random Forest proposed by Salles et al. [2017], the so-called Lazy Nearest Neighbor Random Forest (LazyNN_RF) Classifier, which aims at mitigating the overfitting issue faced by RF in high-dimensional data with many irrelevant or noise attributes, by means of exploiting the neighborhood of test data in order to filter out noise data and, hopefully, learn a more accurate model. Despite being very successful at leveraging the RF generalization capabilities in such challenging scenarios, the LazyNN_RF has a high computational cost at test time due to its lazy nature, which may be prohibitive in real-world tasks. Thus, in order to stack the LazyNN_RF with other RF-based methods (such as BERT and BROOF), we need to compute it faster. We propose to exploit the massively parallel power of Graphical Processing Units (GPUs) and the highly parallelizable characteristic of the kNN algorithm. In order to do so, the algorithm GTkNN proposed by Canuto et al. [2015] will be adopted as a component of the proposed algorithm.

4.1 LazyNN_RF

Lazy learners have their own singular characteristics. Firstly, they are more "localized" than eager learners, since they postpone training until the test sample is given they are able to capture the nuances of the neighborhood of the test data, thus, yielding highly effective prediction models. In addition, they do not assume that training and test data follow the same distribution. Consequently, they are less restrict than eager approaches in terms of data distribution assumptions being able to easily adapt to evolving datasets, without the need for periodical training [Salles et al., 2017].

k-Nearest Neighbor (kNN) classifier is, perhaps, the most well-known and broadly used lazy algorithm. In this method, the majority class of the training instances

composing the neighborhood of the test instance x is used to assess its class label. The neighborhood of the new unseen data x is defined by the set of k instances closest to x in the input space, where closeness implies a metric, such as Euclidean distance or cosine similarity. The rationale here is that, based on the contiguity hypothesis, we expect a test instance x to have the same classification as the training instances located in the neighborhood of x [Manning et al., 2008].

Based upon the fact that traditional Random Forests face some drawbacks, specially when applied to high-dimensional and noisy classification problems, such as textual classification, Salles et al. [2017] proposed the so-called LazyNN_RF algorithm which mitigates the overfitting issue observed when RF is applied to problems with many noisy attributes, by restricting the training set to be composed by observations in the input space that exhibit higher similarity with the test instance, thus being good positive examples (in highly homogeneous regions of the input space) or near-positive negative examples (which lay down on hard-to-classify regions). The projection of training set by means of k-nearest neighbor contributes to a more robust and precise classifier.

Such a localized training set, composed of instances which are similar to the test instance, filters out potentially noise/irrelevant data and reduces the impact of noisy features in the classification model. Moreover, such a projected space allows the model to remove unnecessary bias through a smaller and more localized training set, while also reducing the variance by means of the traditional RF's model averaging procedure.

Nevertheless, LazyNN_RF shares some issues with other lazy methods when compared to eager ones, the most concerning being its computational cost. As briefly mentioned, all computations to learn the model are postponed to classification time, incurring a potentially high computational cost for each test instance. On the other hand, eager learners incur in computational cost only once as they attempt to generalize the observations before receiving test data. Therefore, it is fundamentally important to tackle this issue in order to apply this family of methods in large-scale datasets.

4.2 GTkNN

There exist efficient and exact algorithms to compute the k-Nearest Neighbors for small dimensionality, one of such is kd-trees. However, such methods are not suitable for solving this problem efficiently in high dimensional data. In such challenging scenario, it is usually employed approximate nearest neighbor search algorithms (i.e., ball-trees, locally sensitive hashing). Nevertheless, the impact of such approximation must be

considered.

In order to avoid the potentially side-effects of approximate algorithms, we turn our attention to algorithms that take advantage of the unique characteristics of textual data, such as high sparsity, in order to resolve such problem optimally and efficiently. Supported by Zipf's law, which states that in a textual corpus, few terms are common, while many of them are rather rare, Canuto et al. [2015] proposed a highly efficient and exact algorithm for computing the k-Nearest Neighbors, known as GTkNN, which exploits the Zipf's law by means of combining inverted indexing and the massively parallel power of the Graphical Processing Units (GPUs).

The inverted index serves two purposes. First, it allows the proposed solution to save a lot of memory since the inverted index corresponds to a sparse representation of the data. Second, in the search time, the index is used to quickly find the instances sharing features with the test instance. Thus, restricting the calculation of this distance to smaller amount of instances. In the distance calculation step, a smart load balancing is applied among the GPU threads in order to maintain full occupancy of the GPU cores (increasing the parallelism). Finally, in the sorting phase, a efficient GPU-based partial sorting algorithm, called bitonic sort, is performed in order to avoid sorting all computed distances.

This highly parallel algorithm allows us to generate the projected training set for the LazyNN_RF procedure for large-scale real-world datasets in viable time. We, thus, can step towards our objective which is combining LazyNN_RF and other RF-based algorithms in order to further leverage their generalization power.

In the following section, we show the improvement achieved in runtime performance by adopting the GTkNN algorithm as component of the LazyNN_RF.

4.3 Experimental Evaluation

In order to evaluate the performance of the proposed method in high-dimensional data, we consider six real-world textual datasets, namely, 20 Newsgroups (20NG), Four Universities (4UNI), Reuters 90 (REUT90), ACM Digital Library (ACM), MEDLINE (MED) and RCV1 datasets (one can find a more detailed description in Section 3.1). For all datasets, we performed a traditional preprocessing task: we removed stopwords, using the standard SMART list, and applied a simple feature selection by removing terms with low "document frequency (DF)"[1]. Regarding term weighting, we used TFIDF for all classifiers. All datasets are single-label. In particular, in the case

[1]We removed all terms that occur in less than six documents (i.e., DF<6).

of RCV1, the original dataset is multi-label with the multi-label cases needing special treatment, such as score thresholding, etc. As our current focus is on single-label tasks, to allow a fair comparison among the other datasets (which are also single-label) and all baselines (which also focus on single-label tasks), we decided to transform all multi-label cases into single-label ones. In order to do this fairly, we randomly selected, among all documents with more than one label, a single label to be attached to that document. This procedure was applied in about 20% of the documents of RCV1 which happened to be multi-label.

All experiments were run on a Intel® Xeon, running at 2.2GHz, with 16Gb RAM. The GPU experiments were run on a Nvidia GeForce GTX TITAN Black, with 6Gb RAM.

In order to consider the costs of all data transfers in our efficiency experiments, we report the wall times on a dedicated machine so as to rule out external factors, like high load caused by other processes. To compare the average results of our cross-validation experiments, we assess the statistical significance of our results with a paired t-test with 95% confidence and Bonferroni correction to account for multiple tests. This test assures that the best results, marked in **bold**, are statistically superior to others.

We compare the computation time to automatic classify text using four different algorithms: (1) **g**puLazyNN_RF, our GPU-based implementation of LazyNN_RF (based on GTkNN algorithm); (2) **c**puLazyNN_RF, a parallel C++ implementation of LazyNN_RF kindly made available for us by the author of [Salles et al., 2017]; (3) **S**cikit RandomForestClassifier(RF) , one of the most optimized RF implementation free available [2]; and (4) **G**TkNN , a GPU-based implementation of kNN for text classification[3]. For the aforementioned experiment, the regularization parameter was chosen by using 5-fold cross-validation in the training set for RF. For the size of the neighborhood used for all kNN-based classifiers, we adopted k = 30 in all experiments, since it was empirically demonstrated as the best parameter for text classification [Hao Chen, 1999; Joachims, 1998; Yang, 1999]. In addition, we also fixed the maximum number of trees to 200 for LazyNN_RF as in [Salles et al., 2017].

We would like to stress that we ran all alternatives under the same conditions in all datasets, using the best traditional feature weighting scheme, using standardized and well-accepted cross-validation procedures that optimize parameters for each of alternatives, and applying the proper statistical tools for the analysis of the results.

[2] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

[3] `http://sourceforge.net/projects/gtknn/`

4.3.1 Runtime Analysis

Since lazy learners tend to be computational costly - as previously discussed - we focus our analysis on the performance issue. In table 4.1 we report the time (i.e, overall time to classify all test instances) spent to learn and classify 20% of each dataset in average (considering the remaining 80% as the training set - 5 fold cross validation). All RF-based algorithms presented in table 4.1 construct their trees in parallel (it was used 8 cores to build the trees in parallel). Recall that gpuLazyNN_RF utilizes GTkNN as a component and, thus, both use GPUs parallel power to accelerate their execution. Moreover, the algorithm cpuLazyNN_RF also executes its kNN part in parallel. It can be noticed that GTkNN runs much faster than gpuLazyNN_RF, what was expected since GTkNN is a sub-routine of gpuLazyNN_RF, thefore serves as lower bound.

	Runtime (seconds)			
Dataset	RandomForest(RF)	cpuLazyNN_RF	gpuLazyNN_RF	GTkNN
4UNI	147.97 ± 8.34	567 ± 20.00	**85.48 ± 0.31**	3.58
20NG	140.89 ± 1.27	1112 ± 27.00	**100.66 ± 7.29**	9.10
ACM	1030.25 ± 271.20	432.00 ± 31.00	**118.18 ± 8.27**	5.77
MED	75142.52 ± 293.12	13809.00 ± 206.00	**5604.30 ± 170.96**	990.00
RCV1	19685*	10542.89 ± 307.35	**2764.46 ± 221.31**	1130.00

Table 4.1: Runtime Analysis.(*) Value reported in [Salles et al., 2017].

As one can note, the GPU-based approach of LazyNN_RF outperformed its counterparts RF and cpuLazyNN_RF in all datasets, table 4.1. In MEDLINE, the largest dataset, for instance, the GPU approach reduced the LazyNN_RF's execution time from 3.84 hours to 1.55 hours, which is an impressive result. The most striking result is that gpuLazyNN_RF is in fact even faster than the traditional RF in all datasets. In [Salles et al., 2017] the author hypothesize that LazyNN_RF's runtime is directly related to both the training set size and how well the KNN projection is restricted to pure regions of data. Recall that, we fixed the neighborhood size to 30 examples. At this point, the KNN projection plays a key role to reduce runtime, by restricting the candidate features for splitting to those observed in the examples in the neighborhood of the test data. More importantly, if such a projection is (near-) pure, then tree generation becomes trivial. This explains why the approaches cpuLazyNN_RF and gpuLazyNN_RF become faster, compared to RF, for datasets with more documents and attributes such as MEDLINE and RCV1. Nevertheless, it can be noticed that the negative influence of the kNN over the algorithm performance has vanished, when using the GTkNN as sub-routine of gpuLazyNN_RF.

With those promising results, we can step towards our main objective which is

to combine LazyNN_RF and other RF-based approaches in order to further leverage their effectiveness on new unseen data.

Chapter 5

BERT Classifier

In this chapter, we detail BERT, a boosted version of the RF classifier that aims at taking the advantages of BROOF while avoiding its potential bias tendency. As we detail in the following, BERT introduces an additional source of randomization to reduce bias, while, at the same time, exploiting the advantages of the boosting strategy, in order to achieve superior generalization. We start our discussion with a brief overview of the BROOF strategy. Then, we describe the key building block for the BERT strategy, namely, the Extremely Randomized Trees. Finally, we describe BERT.

5.1 BROOF

Proposed by Salles et al. [2015], the BROOF classifier combines boosting and bagging by exploiting RF as "weak learners" in a boosting framework. Boosting is a sequential meta-algorithm which trains several "weak learners" (that is, classifiers capable of yielding predictions better than random guessing) in order to generate a more precise model. For each iteration i of the boosting algorithm, let Δ^i be a probability distribution over the training set of size M. When $i = 0$, $\Delta^0(j) = \frac{1}{M}$, $\forall j|_{j=1}^{M}$. At iteration $i > 0$, for all examples in the training set x_j, if x_j is misclassified, its weight is incremented so that, in the succeeding iteration, an updated distribution Δ^{i+1} is considered, which emphasizes the misclassified instances (e.g, the hardest to classify ones). In contrast to AdaBoost Freund and Schapire [1997], the update rule in BROOF uses the out-of-bag (OOB) samples, promptly generated by the RF classifier at training time, as an unbiased error estimate to drive the boosting iterations and to "smoothly" reweights just the OOB instances. Salles et al. [2015] show by an extensively experimental workload that restricting the influence of training data to the hard-to-classify

sub-regions of the input space, by means of a smooth combination of random forests and boosting, can mitigate the overfitting issue observed when learning decision trees composing the ensemble classifier, thus leveraging classification effectiveness on new unseen data. Also, the selective weight update strategy slows down boosting's tendency to focus on just a few hard-to-classify samples, thus offering some bias reduction capabilities. These ideias were even extended and generalized to the realm of learning-to-rank (L2R) tasks [de Sá et al., 2016]. However, although BROOF was shown to provide competitive results in text classification tasks, this can be mainly attributed to variance reduction, since its limited bias reduction capability [Salles et al., 2017] may not be enough to fully mitigate its tendency to overly emphasize hard-to-classify examples, mainly in noisy environments. This has to do with the underlying boosting strategy adopted by BROOF and may compromise classification effectiveness. We here propose to tackle the potentially high bias faced by BROOF by means of an additional source of randomization, through the use of the so-called extremely randomized trees, described in the following.

5.2 Extremely Randomized Trees

Extremely Randomized Trees (a.k.a., Extra-Trees) [Geurts et al., 2006] is an ensemble of trees. Unlike RFs, Extra-Trees do not bootstrap the training data when learning its composing trees. Instead, it uses the entire training set for doing so, relying on another more aggressive source of randomization to learn decorrelated trees by combining random cut-point choice and random attribute selection drawn from a randomly chosen subset of features while building the trees, thus guaranteeing reduced tree correlation in the ensemble. Furthermore, Geurts [2002] shows that cut-point variance seems to be responsible for a significant part of the generalization error of decision trees, thus, from a bias-variance tradeoff perspective, Extra-Trees can be seen as a manner to transfer cut-point variance (through additional randomness while finding the split) to the one due to random effects in the training set, which is reducible by averaging[Louppe, 2014].

Algorithm 4 Extra-Tree cut-point selection pseudo-code

1: **function** FINDCUTPOINT($f_j, node$)
2: $min_j \leftarrow \min(\{x_{i,j} | (X_i, y_i) \in node\})$
3: $max_j \leftarrow \max(\{x_{i,j} | (X_i, y_i) \in node\})$
4: Draw a cut-point v uniformly from $[\min_j, \max_j[$
5: **end function**

The Extra-Trees classifier has some benefits when compared to RF classifiers. Besides yielding better classification effectiveness, it has been shown that Extra-Trees is robust to noise data and outliers. Moreover, extra-trees carries over many common features from Random Forest, such as variable importances and proximity measure. Another interesting feature of Extra-Trees is the ability of learning models in an unsupervised fashion. When the size of the subset of randomly chosen features is set to 1 and the cut-point is selected in a random fashion, the structere of the tree can be learned independently of the output variable $\mathbb{Y}$. However, in noisy scenarios this way of growing trees can incur in highly biased models, thus, degrading Extra-Trees predictive performance.

In this work, we take advantage of BROOF-like techniques and Extra-Trees, by proposing what we call BERT, a smooth combination of both, as detailed next.

5.3 Boosted Extremely Randomized Trees

Recall that boosting algorithms tend to overly emphasize hard-to-classify examples, mainly when applied to noisy data. Therefore, the undesired bias towards these hard-to-classify examples is minimized by BROOF by updating only the probabilities Δ^i related to out-of-bag samples as proposed by Salles et al. [2015], hoping to decrease the misclassification rate when the "weak learner" is focused on hard-to-classify regions. Hence, by using a "weak learner" more robust to noisy data than Random Forests such as Extra-Trees [Geurts et al., 2006], it is expected to achieve better performance when focused on hard-to-classify regions, thus, producing a model with higher generalization power. In fact, as we shall see, BERT is among the top performer classifiers in all tested datasets, outperforming the original BROOF in several cases.

BERT combines boosting, bagging and Extra-Trees, by exploiting the following BROOF-like strategies: (i) to use the out-of-bag (OOB) error estimate as a less biased error estimation to drive the boosting algorithm; and (ii) to only update the weights of the out-of-bag instances during the boosting iterations. In order to smoothly combine these ideas to the Extremely Randomized Trees framework, we propose to introduce the bagging procedure into its training procedure, in order to allow proper OOB error estimation. During the Bootstrap procedure employed to generate sub-training sets in order to train each tree in the ensemble, each tree is trained with approximated $1-\frac{1}{\epsilon} \approx 63\%$ of the original training set[Hastie et al., 2009]. Similarly to cross-validation procedure, the examples left out (Out-of-Bag) of the trees training can be used to estimate an unbiased expected error rate. Hastie et al. [2009] showed that the OOB

error is less biased than the training one, which is frequently adopted to weight the weak-learners votes in boosting methods.

We argue that exploring these strategies through this novel classification framework brigs two benefits: it enables us to minimize variance (mitigating the overfitting problem faced by the trees composing the ensemble) and also provide us means to minimize bias, through the additional randomization source, leveraging the framework ability to avoid being stuck on a few hard-to-classify examples.

Algorithm 5 BERT Pseudo Code

1: **function** TRAIN($\mathbb{D}_{train} = \{(X_i, y_i)|_{i=1}^{N}\}$, M, n_{trees})
2: $L \leftarrow \emptyset$;
3: $w_1 \leftarrow \frac{1}{|D_{train}|}$
4: **for** each $m \in \{1, .., M\}$ **do**
5: $(h_m^{XT}, (x, y, \hat{y})_i^{oob}) \leftarrow BaggedExtraTrees(\mathbb{D}_{train}, n_{trees}, w_m)$
6: $OOB_{err}^{w} \leftarrow \frac{\sum_{i \in \mathbb{O}} w_m^i I[y \neq \hat{y}]}{\sum_{i \in \mathbb{O}} w_m^i}$, where $\mathbb{O} = (x, y, \hat{y})_i^{oob}$
7: $\alpha_m \leftarrow \log(\frac{1-OOB_{err}^{w}}{OOB_{err}^{w}})$
8: $w_{m+1}^i \leftarrow \frac{w_m^i \epsilon^{\alpha_m I[y \neq \hat{y}]}}{\mathbb{Z}}$, where $\mathbb{Z}$ is a normalizing constant
9: $L \leftarrow L \cup \{(h_m^{XT}, \alpha_m)\}$
10: **end for**
11: **return** L
12: **end function**

We summarize the proposed method in Algorithm 5. Given a training set $\mathbb{D}_{train}$, the number of boosting iterations M and the number of trees built per iteration n_{trees}. Initially, all training instances have equal probability mass $w_1 = \frac{1}{|D_{train}|}$. For each iteration $m \in M$ an Extra-Trees (with bagging) classifier is learned considering instances weight w_m in the sampling step of the bagging procedure as a probability distribution. The weights drive the ensemble towards hard to classify regions of the input space. Thus, if the ensemble focus on such regions, its capability of accurately covering such complex regions increases[Salles et al., 2015]. We do so by re-weighting the out-of-bag instances which were misclassified by $h_m^{ExtraTrees}$. Let $t_j \in h_m^{ExtraTrees}$ be an Extra-Tree of the ensemble learned in the m-th iteration, the predicted class $\hat{y}$ for the i-th OOB instance x can be computed as follows: $\hat{y} = \arg_c \max \sum_{j=1}^{n_{trees}} t_j(x) I[x \in oob_j]$, where I is an indicator function and oob_j is the set of out-of-bag instances of the j-th tree. Thus, the weighted out-of-bag error estimation OOB_{err}^{w} can be obtained as shown in the line 6 and used to update the weights of each OOB instance (lines 7-8).

5.4 Experimental Evaluation

We now report and discuss the conducted experimentally evaluation regarding the proposed BERT classifier considering a set of datasets regarding topic categorization and sentiment analysis. We first detail the explored datasets and the experimental setup. Then, we discuss the obtained experimental results, comparing our proposal to the other explored state-of-the-art classifiers.

5.4.1 Experimental Setup and Parameterization

5.4.1.1 Datasets

In order to evaluate the BERT classifier considering the textual classification domain, we consider five real-world topic categorization data sets as well as ten sentiment analysis ones, related to computer science articles (ACM), news (REUT), web pages (4UNI), medicine (MEDLINE), items reviews (Amazon), posts on social networks (Twitter, Debate), user comments (Youtube) and snippets of opinion news (NYT). One can find more detail on the datasest in Chapter 3. In all cases, we performed a traditional preprocessing task that consists of removing stopwords (using the standard SMART list) and applying a simple feature selection procedure, removing terms with low "document frequency (DF)"[1]. Regarding term weighting, we tested TF, TF-IDF and L2 normalization schemes, choosing the best strategy for each classification approach. Particularly, we use TF for all classifiers based on RF and Naive Bayes, and TF-IDF with L2 normalization for both SVM and kNN.

The explored classifiers were compared using two standard information retrieval measures: *micro averaged* F_1 (MicroF_1) and *macro averaged* F_1 (MacroF_1). While the MicroF_1 measures the classification effectiveness overall decisions (i.e., the pooled contingency tables of all classes), the MacroF_1 measures the classification effectiveness for each individual class and averages them (see more in section 3.3). To compare the average results of our **5-fold** cross-validation experiments, we assess their statistical significance by applying a paired two-tailed t-test with 95% confidence and Bonferroni correction to account for multiple comparisons. This test assures that the best results, marked in **bold**, are statistically superior to others (up to the chosen confidence level).

We evaluate ten distinct learning algorithms (a more detailed overview can be found in section 3.2). We use the scikit-learn implementation[2] of linear SVM, k-Nearest Neighbors (kNN), Multinomial Naïve Bayes (NB), Random Forests (RF) and

[1] We removed all terms that occur in less than six documents (i.e., DF<6).

[2] Available in `http://scikit-learn.org/`

Extremely Randomized Trees (Extra-Trees). We use our own implementation of the RF-based methods, namely, BROOF, the lazy version of the RF classifier (LAZY) and the lazy version of extra-trees (LXT), since there is no freely available implementation for these classifiers.

The free parameters of these classifiers include the cost C (for SVM), neighborhood size k (for KNN and LAZY) and the number of features considered in the split of a node on the RF-based approaches. These free parameters were set using 5-fold cross-validation within the training set. For the RF-based approaches, each tree is grown without pruning, as suggested in Hastie et al. [2009], and since the results obtained with 200, 300 and 500 trees are statistically tied (with 95% confidence), we adopted 200 trees due to the lower cost. Concerning the BROOF classifier, we use 8 weak learners, setting the maximum number of iterations to 200, as suggested in Salles et al. [2015]. We use the same parameters for the proposed BERT method.

We would like to point out that some of the results obtained in some datasets may differ from the ones reported in other works for the same datasets. Such discrepancies may be due to several factors such as differences in dataset preparation[3], the use of different splits of the datasets (e.g., some datasets have "default splits" such as REUT and 20NG[4]). We would like to stress that we ran all alternatives under the same conditions in all datasets, using the best traditional feature weighting scheme, using standardized and well-accepted cross-validation procedures that optimize parameters for each of alternatives, and applying the proper statistical tools for the analysis of the results. Our datasets are available (for result replication and testing of new configurations) under request.

5.4.2 Results and Discussions

We now turn our attention to the obtained results regarding the described classifiers. We start by considering the effectiveness of each classifier in the topic categorization task (see Table 5.1). In this case, BERT presents statistically tied results with SVM on most datasets despite their fundamentally different classification paradigms. While BERT and other RF-based approaches are based on extracting specific association rules that relate different features, SVM measures the complexity of hypotheses based on the margin with which they separate the data, which is independent of the number of features. This characteristic makes the SVM as one of the best-known classification

[3]For instance, some works do exploit complex feature weighting schemes or feature selection mechanisms that do favor some algorithms in detriment to others.

[4]We believe that running experiments only in the default splits is not the best experimental procedure as it does not allow a proper statistical treatment of the results.

		20NG	4UNI	ACM
BERT	microF1	**89.45 ± 0.46**	**84.61 ± 0.98**	**74.8 ± 0.59**
	macroF1	**89.13 ± 0.58**	**73.61 ± 1.85**	**62.1 ± 0.99**
SVM	microF1	**90.06 ± 0.43**	**83.48 ± 1.08**	**75.4 ± 0.66**
	macroF1	**89.93 ± 0.43**	**73.39 ± 2.17**	**63.84 ± 0.55**
BROOF	microF1	87.96 ± 0.24	**84.41 ± 1.07**	73.35 ± 0.79
	macroF1	87.44 ± 0.28	**73.23 ± 1.10**	60.76 ± 0.82
LAZY	microF1	87.96 ± 0.37	**82.34 ± 0.61**	**74.02 ± 0.79**
	macroF1	87.39 ± 0.37	68.33 ± 1.6	59.46 ± 1.35
NB	microF1	**88.99 ± 0.54**	62.63 ± 1.7	73.54 ± 0.71
	macroF1	**88.68 ± 0.55**	51.38 ± 3.19	58.03 ± 0.85
KNN	microF1	87.53 ± 0.69	75.63 ± 0.94	70.99 ± 0.96
	macroF1	87.22 ± 0.66	60.34 ± 1.36	55.85 ± 0.97
Extra-Trees	microF1	87.03 ± 0.41	**82.87 ± 1.00**	73.08 ± 0.55
	macroF1	86.65 ± 0.56	68.54 ± 2.60	58.71 ± 0.89
LXT	microF1	88.39 ± 0.51	81.24 ± 0.71	69.63 ± 0.91
	macroF1	88.05 ± 0.44	66.89 ± 1.23	57.33 ± 1.48
RF	microF1	83.64 ± 0.29	81.52 ± 1	71.05 ± 0.31
	macroF1	83.08 ± 0.35	65.44 ± 1.91	56.56 ± 0.45

(a) Datasets: 20NG, 4UNI and ACM

		REUT90	SPAM	MED
BERT	microF1	**67.33 ± 0.72**	**96.11 ± 0.52**	83.68 ± 0.32
	macroF1	**89.13 ± 0.58**	**95.93 ± 0.55**	74.25 ± 0.37
SVM	microF1	**68.19 ± 1.15**	92.55 ± 0.8	**86.19 ± 0.05**
	macroF1	**31.95 ± 2.59**	92.12 ± 0.87	**78.46 ± 0.42**
BROOF	microF1	**66.79 ± 0.97**	**96.09 ± 0.84**	83.05 ± 0.05
	macroF1	**28.48 ± 2.17**	**95.9 ± 0.88**	73.25 ± 0.42
LAZY	microF1	**66.3 ± 1.07**	92.91 ± 0.68	84.88 ± 0.08
	macroF1	26.61 ± 2.12	92.54 ± 0.71	72.90 ± 0.08
NB	microF1	65.32 ± 1.13	79.27 ± 0.81	82.92 ± 0.14
	macroF1	**27.86 ± 0.79**	78.18 ± 0.83	63.8 ± 0.43
KNN	microF1	**68.07 ± 1.07**	83.31 ± 0.98	82.16 ± 0.08
	macroF1	**29.93 ± 2.48**	82.91 ± 0.93	68.00 ± 0.34
Extra-Trees	microF1	64.87 ± 0.81	**95.74 ± 0.55**	82.49 ± 0.07
	macroF1	26.18 ± 2.55	**95.52 ± 0.58**	71.15 ± 0.31
LXT	microF1	65.92 ± 0.82	92.42 ± 0.78	83.84 ± 0.11
	macroF1	26.71 ± 2.53	92.05 ± 0.82	71.02 ± 0.23
RF	microF1	63.92 ± 0.81	**95.46 ± 0.74**	81.61 ± 0.05
	macroF1	24.36 ± 1.98	**95.22 ± 0.79**	70.41 ± 0.36

(b) Datasets: REUT90, SPAM and MED

Table 5.1: Topic categorization - Obtained results for base classifiers.

strategies to exploit discriminative evidence from high dimensional and sparse textual data. Even considering the specificities of each feature, BERT is capable of combining small pieces of evidence to build a general model with the same generalization power of SVM in scenarios which SVM traditionally works the best.

Despite the fact that SVM and BERT obtain statistically tied results in most

datasets, BERT shows significantly superior results to SVM in the SPAM dataset. For this particular dataset, the BERT capability of identifying specific non-trivial relationships between individual words to the spam category grants superior effectiveness for BERT. On the other hand, the same SVM mechanism that captures good general patterns limits such classifier in finding specific pieces of evidence that relate individual features in more complex ways.

BERT is also superior to other RF-based approaches in most datasets, which provides evidence towards the benefits of the proposed strategy in mitigating the RF overfitting problem. Specifically, BERT presented significant improvements over RF on all datasets, with gains up to 7% and 30% on MicroF$_1$ and MacroF$_1$, respectively.

Other strategies designed to overcome the RF limitations also presented inferior results to BERT. The LAZY method is always inferior to BERT in terms of MacroF$_1$, indicating that LAZY has a hard time on discriminating minor classes. We argue that by trying to find discriminative patterns based on the neighborhood of documents, LAZY can bias its model towards the larger classes, since their documents are most likely to appear in the neighborhood of an arbitrary test document. Another strategy aimed at improving RFs is the BROOF classifier. Our experimental results show that the proposed combination of boosting with extremly randomized trees can achieve better results than combining boosting with the original RFs. Although very competitive to BERT, BROOF is no match to our proposal in both the 20NG and ACM datasets.

Other classifiers, such as NB and KNN follow a completely different approach than SVM and the aforementioned RF-based methods. The simplicity of both strategies sometimes is not enough to provide effective classification from complex data distributions. However, when there is sufficiently discriminative evidence from individual words or similarity between documents, KNN and NB can achieve reasonable results, as the ones obained on news categorization datasets using these methods. However, even achieving some interesting results, NB and KNN can be substantially inferior to BERT, as noticed in 4UNI and ACM.

Now we turn our attention to the classification results of the sentiment analysis task, presented in Table 5.2. The experimental results show that, overall, BERT outperforms or ties most of its results with the best classification method for each dataset. Specifically, BERT and BROOF achieved the best (and statistically tied) results in almost all datasets, which provides additional evidence towards the benefits of reducing the overfitting issues of RF-based methods using boosting. In the sentiment analysis context, both methods take advantage of being able to identify the presence of individual words that are highly correlated to a positive or negative sentiment. Their generalization capabilities of identifying noisy or irrelevant information provide improvements

over the original RF, with 14% and 5% on MacroF$_1$ and MicroF$_1$, respectively.

		AMAZON	BBC	DEBATE	DIGG	MYSPACE
BERT	microF1	**75.76**	**86.04**	**79.38**	**75.7**	**86.46**
	macroF1	**74.16**	**53.78**	**77.24**	**63.03**	**67.35**
BROOF	microF1	74.52	**86.17**	**79.23**	**76.34**	**86.57**
	macroF1	**73.02**	**55.04**	**77.07**	**65.56**	**67.93**
NB	microF1	**74.68**	**86.84**	76	**76.09**	**85.5**
	macroF1	**73.32**	46.48	73.7	**64.38**	60.39
XT	microF1	73.88	**86.58**	**79.33**	**76.85**	**85.64**
	macroF1	72.1	**52.35**	**76.93**	60.84	**65.24**
SVM	microF1	74.24	86.43	**78.58**	**75.58**	**86.23**
	macroF1	72.84	50.34	**76.2**	60.96	**67.94**
RF	microF1	73.8	**86.84**	**78.53**	**75.31**	**85.74**
	macroF1	71.76	**52.4**	**75.79**	57.04	60.7
LXT	microF1	72.22	**87.11**	**78.07**	**74.93**	85.26
	macroF1	69.24	49.36	74.98	56.91	55.64
LAZY	microF1	72.3	**87.37**	76.35	**76.21**	85.14
	macroF1	69	50.19	72.89	58.9	55.55
KNN	microF1	69.86	86.44	74.03	74.8	**85.49**
	macroF1	67.64	46.36	72.25	52.77	56.07

(a) Datasets: AMAZON, BBC, DEBATE, DIGG and MYSPACE

		NYT	TWEETS	TWITTER	YELP	YOUTUBE
BERT	microF1	**68.14**	**88.3**	**76.15**	**94.26**	79.85
	macroF1	**67.01**	**85.49**	**74.71**	**94.26**	76.47
BROOF	microF1	**68.01**	**88.11**	**75.01**	**93.58**	79.69
	macroF1	**66.85**	**85.36**	**73.73**	**93.58**	76.18
NB	microF1	**67.1**	86.82	**74.88**	90.44	**83.43**
	macroF1	**66.06**	**84.4**	**73.89**	90.44	**80.3**
XT	microF1	**67.43**	86.46	**75.19**	91.74	79.65
	macroF1	**66.03**	83	**73.26**	91.73	76.22
SVM	microF1	66.34	86.87	**74.53**	92.94	**82.24**
	macroF1	65.48	**84.1**	**73.15**	92.94	77.99
RF	microF1	**67.43**	84.63	73.83	90.76	79.89
	macroF1	65.41	79.54	71.21	90.75	76.29
LXT	microF1	64.41	82.36	71.91	90.82	77.22
	macroF1	61.5	75.35	68.51	90.82	67.56
LAZY	microF1	64.64	80.86	70.86	90.18	76.48
	macroF1	61.61	73.53	67.49	90.17	66.4
KNN	microF1	60.61	77.34	67.02	74.5	75.74
	macroF1	54.92	68.14	64.08	73.83	67.27

(b) Datasets: NYT, TWEETS, TWITTER, YELP and YOUTUBE

Table 5.2: Sentiment analysis - Obtained results for base classifiers.

In the sentiment analysis scenario, SVM is not always among the best approaches, since it does not have the ability to discriminate the fine-grained discriminative evidence present in individual words and non-linear relationships between words. Moreover, there is no clear winner between SVM and the simple NB classifier. In this scenario, NB has the advantage of learning how each word relates to a positive or negative

sentiment, and they combine these pieces of evidence to determine the sentiment of a text document. However, unlike RF-based methods, there is no mechanism to identify noise or irrelevant words, which may affect its effectiveness results.

KNN, LAZY, and LXT are usually associated with the worse results for the sentiment analysis task. Instead of focusing on individual words, these methods exploit the distribution of training examples among neighbors of a test example. This analysis of the neighborhood can be a potential source of noise since similar text documents can be associated with different sentiment labels.

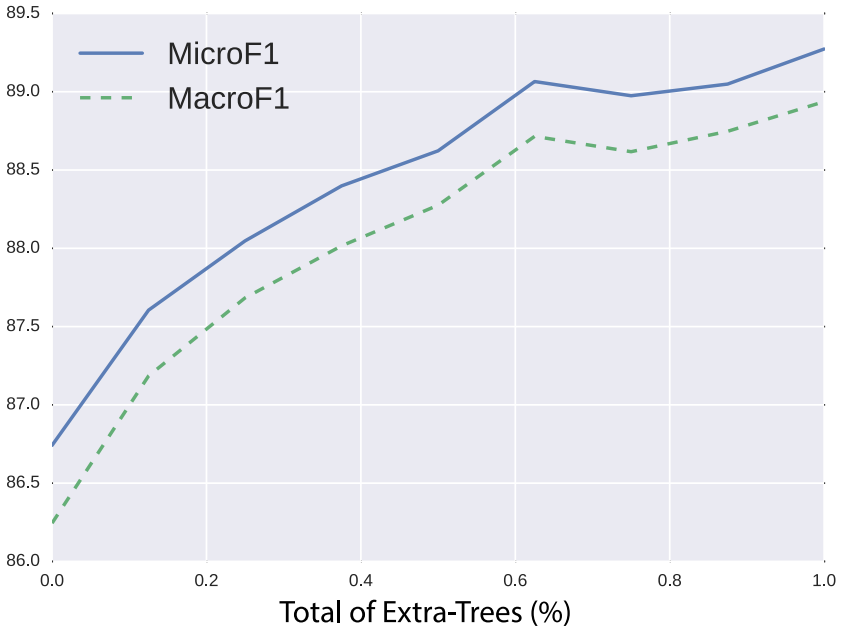
Overall, according to the reported experimental results, the BERT classifier is the only approach consistently among the best results in both topic categorization and sentiment analysis. These results provide empirical evidence towards the benefits of improving the generalization power of random forests by using elaborated strategies to reduce overfitting.

Finally, for both sentiment analysis and topic classification, BERT is one of the most effective classification methods that combines discriminative pieces of evidence derived from exploring the complex sub-regions of the input space. Other classification approaches provide completely different strategies to exploit discriminative patterns, which motivates us to stack these different classification approaches to combine the potentially complemental information captured by each of them (see Chapter 6).

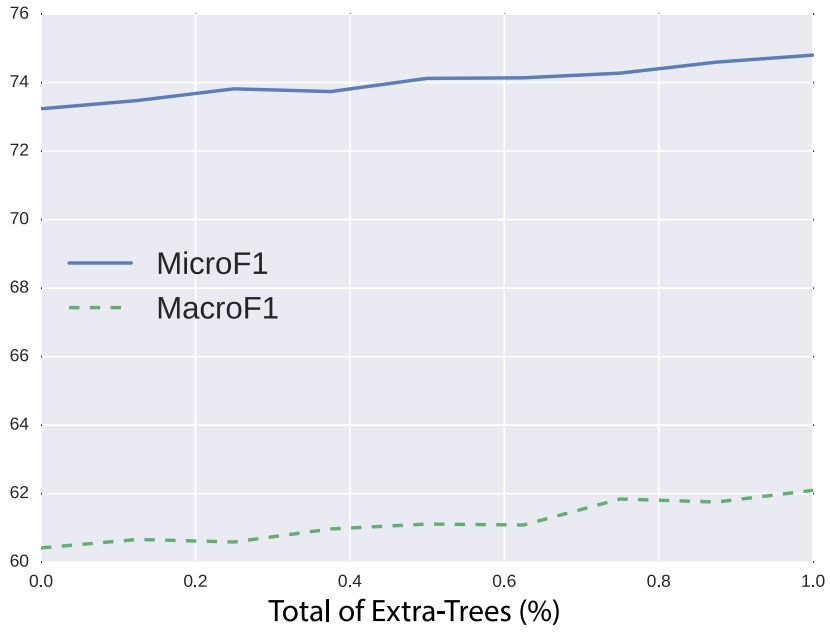
5.4.3 Effects of Extra-Trees as weak-learner

An important aspect to be further analyzed is the influence of the additional randomization enjoyed by BERT through the use of Extra-Trees (instead of traditional RFs, as done in BROOF). To this end, we analyze the effect of gradually increasing the randomness in the training process over MicroF$_1$ and MacroF$_1$. Here, the number of iterations and trees were fixed to 200 and 8, respectively.

Figure 5.2 shows the obtained results, where the x-axis represents the proportion of Extra-Trees composing the ensemble (0% means the absence of Extra-Trees, which degenerates to BROOF, and 100% represents the BERT classifier, composed entirely by Extra-Trees). There is a clear growth tendency in MicroF$_1$ and MacroF$_1$ as we increase the proportion of Extra-Trees composing the ensemble. Thus, the additional randomization procedure employed in BERT plays an important role in improving the BROOF algorithm.



(a) 20NG



(b) ACM

Figure 5.2: Effect of Extra-Trees as weak-learner.

Chapter 6

Stacking RF-Based Learners

Ensembles of classifiers are an extensively studied machine learning technique [Rokach, 2009]. Among several learning methods used in practice, ensembles are one of the most effective in several applications and can be divided into two groups: ensembles of homogeneous and heterogeneous classifiers. The former relies on learning many versions of the same classification technique, each one built by somehow disturbing the training set. Then, usually one averages the predictions in order to come up with final decisions with higher generalization capabilities. The latter combines a set of distinct learning methods trained with the same training set and then produces the final decisions according to the predictions made by these classifiers.

Regarding the text classification realm, Dong and Han [2004] use what they call Moderated Asymmetric Naïve Bayes (MANB) as a base learner in their homogeneous ensemble configuration. Several homogeneous ensemble methods are contrasted, such as k-fold partitioning, bagging and boosting, as well as a heterogeneous ensemble method which combines SVM and NB learning algorithms. Salles et al. [2015] combine two well-known homogeneous ensemble techniques, bagging and boosting, by exploiting Random Forests (RF) as "weak learners" in the boosting framework. The combination is achieved by means of "smoothly" updating the weights of only the out-of-bag instances. This combination mitigates the overfitting issue faced by RF models in textual classification tasks and leverages its generalization power. Recently, Onan et al. [2016] empirically evaluate the effectiveness of ensemble learning methods on textual documents represented by keywords. They apply different keyword extraction strategies on the dataset. The authors evaluate five different homogeneous ensemble methods that use four different base classifiers. In [Bi et al., 2007; Pui et al., 2006; Danesh et al., 2007], ensembles of heterogeneous classifiers are proposed, by combining classical text classification methods (e.g., SVM, kNN, NB and Rocchio). In all

analyzed cases, significant improvements were observed when compared to the traditional classifiers. Furthermore, a combination of several distinct polarity classifiers for Twitter sentiment analysis is proposed by Kanakaraj and Guddeti [2015]. Unlike these works, we here aim at combining homogeneous and heterogeneous classifiers by stacking different RF-based methods in an original manner.

There are two common techniques to combine predictions of distinct classifiers, namely, fixed combining methods and trainable combining methods [Nguyen et al., 2016]. An advantage of applying fixed methods for ensemble systems is that there are no need to train a meta-classifier since they do not take into consideration the the meta-level training set when combining the learners. Thus, they are less time-consuming than their counterparts. Many fixed combining methods can be found in literature, such as Sum, Product, Vote, and Average [Bauer and Kohavi, 1999; Kuncheva, 2002].

On the other hand, trainable combining methods work on meta-level data, in order to learn how to combine the base learners' outputs. Although exploiting meta-level data to extract knowledge usually leverages classification effectiveness, it comes at the price of additional computational effort [Nguyen et al., 2016].

6.1 Stacking

Perhaps, the most relevant studies about trainable combining methods are based on Stacking (a.k.a. Blending), which was originally conceived as "Stacked Generalization" by Wolpert Wolpert [1992]. In this case, a meta-level training set $\mathbb{D}_{meta}$ is generated by applying a **cross-validation** procedure, in which the original training set $\mathbb{D}_{train}$ is divided into K equally sized disjoint sets $\mathbb{D}^k_{fold}, \forall k = 1, ..., K$. Each base-level classifier is learned by considering $\mathbb{D}_{train} \setminus \mathbb{D}^k_{fold}$ as training set and reserving $\mathbb{D}^k_{fold}$ for testing. Formally, for all base learning algorithms L_i and $\forall k = 1, ..., K : h^k_i = l_i(\mathbb{D}_{train} \setminus \mathbb{D}^k_f old)$. Subsequently, each learned classifier h^k_i produces a estimation for the posterior probability $p^k_i(c_m|x_j)$ that an observation x_j belongs to a class c_m; $\forall x_j \in \mathbb{D}^k_{fold} : h^k_i(x_j) = p^k_i(\mathbb{C}|x_j) = (p^k_i(c_1|x_j), p^k_i(c_2|x_j), ..., p^k_i(c_M|x_j))$. The obtained meta-level training set $\mathbb{D}_{meta}$ is thus:

$$\begin{bmatrix} p_1(y_1|x_1) & \dots & p_1(y_M|x_1) & \dots & p_K(y_1|x_1) & \dots & p_K(y_M|x_1) \\ p_1(y_1|x_2) & \dots & p_1(y_M|x_2) & \dots & p_K(y_1|x_2) & \dots & p_K(y_M|x_2) \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ p_1(y_1|x_N) & \dots & p_1(y_M|x_N) & \dots & p_K(y_1|x_N) & \dots & p_K(y_M|x_N) \end{bmatrix}$$

Finally, a combining classifier is trained on the meta-level training set and used to produce the final prediction.

Several methods have been designed to exploit label information in the meta-level training set. In one strategy, the predictions of classifiers are grouped according to the given classes and then a template associated with each label is built. Three methods using that strategy are Multiple Response Linear Regression (MLR) [Ting and Witten, 1997], Decision Template (DT) [Kuncheva et al., 2001] and the recently proposed VIG [Nguyen et al., 2016]. MLR assumes that each classifier weights differently each class. In this case, the combining algorithm is based on the M linear combinations of the posterior class probabilities and the associated class weights. The predicted class is then decided by selecting the maximum value among these combinations. The Decision Template strategy [Kuncheva et al., 2001] groups the meta-level training set according to the classes of each instance. Then, Decision Templates are built by averaging the meta-level instances observed in each class (forming centroids). Subsequently, the predicted class is decided by selecting the class label of the Decision Template that is more similar to the meta-data unlabeled observation. To this end, the authors propose eleven similarity functions based on fuzzy logic. Due to its simple computation, this method has low computational cost in both training and testing. Merz [1999] proposed a combination of Stacking, Correspondence Analysis (CA) and K-Nearest Neighbor (KNN), in the form of a single learning algorithm called SCANN. The goal of such algorithm is to find the underlying relationship between the learning observations and the predictions of the base classifiers, by applying CA to an indicator matrix formed by the learned meta-level instances and their corresponding true labels. Then, a kNN procedure is employed to classify the unseen data in the new scaled space. Recently, Nguyen et al. [2016] proposed a combining classifier based on the variational inference which is based on the assumption that instances belonging to a given class of the meta-level training set are drawn from a multivariate Gaussian distribution. Thus, M distributions are estimated by using Bayesian variational inference on the instances belonging to each class. The predicted class for a new unseen observation is given by selecting the label associated with maximum posterior probabilities computed by the M multivariate Gaussian models. All aforementioned methods are used as baselines in our stacking experiments.

Moreover, all those methods rely on the meta-level training sets obtained by costly procedures, such as cross-validation while combining bagging-based methods. In contrast, we here propose to take advantage of the out-of-bag (OOB) samples, naturally available by the bagging procedure, to yield an unbiased meta-level training set. This allows our solution to stack bagging-based methods with any other learning method without additional computational cost while using any combining methodology. In fact, some works attempt to improve the bagging procedure by utilizing stacking to

combine the base classifiers [Ting and Witten, 1997; Wolpert and Macready, 1996]. For instance, Ting and Witten [1997] propose a variant of bagging, where stacking rather than uniform majority voting is used to achieve the combination. The meta-level training set is obtained by the predictions of the each base learner over the original training set, which can lead the meta-classifier to overfit. In contrast to Ting and Witten [1997], Wolpert and Macready [1996] propose a linear combination in which the coefficients are estimated by the OOB error, thus generating a less biased combination. Those methods differ from ours since here we construct the meta-level data for bagging-based methods in a distinct manner, while also treating them as black-boxes when combining them with any other learning method.

6.2 Degree of Disagreement among Classifiers

Based upon the results reported in Section 5.4, one aspect should be clear by now: the analyzed RF-based classifiers do excel in both the explored text classification tasks. One question that naturally arises is: *can we explore these methods somehow in order to learn an even more effective classifier?* This is what we pursue in this section.

In order to combine RF-based classifiers, these classifiers should exhibit some degree of complementarity. In fact, each RF based classifier explore different learning strategies to come up with more effective predictions. However, it is still important to assess whether these distinct strategies do produce complementary information that could be explored to leverage classification effectiveness. To this end, we quantify such complementarity degree by means of the Degree of Disagreement [Skalak, 1996; Kuncheva and Whitaker, 2003] observed for a pair of classifiers. Let h_i and h_j be two classifiers, applied to examples from a validation set $\mathbb{D}_{valid}$ (e.g., a fold). Also, let $\mathbb{D}_{00}$ be the examples misclassified by both h_i and h_j ($n_{00} = |\mathbb{D}_{00}|$), $\mathbb{D}_{01}$ be the examples correctly classified just by h_i (($n_{01} = |\mathbb{D}_{01}|$)) and $\mathbb{D}_{10}$ be the examples correctly classified just by h_j (($n_{10} = |\mathbb{D}_{10}|$)). Finally, let $\mathbb{D}_{11}$ be the examples correctly classified by both learners (($n_{11} = |\mathbb{D}_{11}|$)). The Degree of Disagreement $\text{Dis}_{i,j}$ between h_i and h_j is given by:

$$\text{Dis}_{i,j} = \frac{n_{01} + n_{10}}{n_{00} + n_{01} + n_{10} + n_{11}} \tag{6.1}$$

This is a symmetric statistic that captures to what extent two classifiers disagree in terms of prediction. Classifiers with low disagreement degree tend to have similar behavior in terms of correctly or incorrectly classifying unseen examples and thus have low complementarity. In order to offer a more appropriate measure of disagreement

degree between classifiers, however, one should also take into account their prediction capabilities. That way we guarantee a proper comparison between their behavior. For this purpose, we propose a normalized degree of disagreement metric. Recall that, in order to have minimal degree of disagreement, we must have $\mathbb{D}_{01} = \emptyset$ or $\mathbb{D}_{10} = \emptyset$. Also, we have that h_i accuracy can be expressed as $R_i = \frac{n_{01}+n_{11}}{\sum_{a,b\in 0,1} n_{a,b}}$, with R_j expressed analogously. It is straightforward to show that $\text{Dis}_{i,j}^{\min} = R_i + R_j - 2\min(R_i, R_j)$ (appendix A.1). Similarly, we maximize the degree of disagreement when n_{00} and n_{11} tend to 0. In that case, with some algebraic manipulation (see in appendix A.1), one can show that $\text{Dis}_{i,j}^{\max} = \min(R_i + R_j, 2 - R_i - R_j)$. With such derivations in place, the normalized degree of disagreement metric is defined as

$$\text{Dis}_{i,j}^{\text{norm}} = \frac{\text{Dis}_{i,j} - \text{Dis}_{i,j}^{\min}}{\text{Dis}_{i,j}^{\max} - \text{Dis}_{i,j}^{\min}}. \tag{6.2}$$

The Degree of Disagreement values computed for the exemplified cases can be found on Table 6.1. This gives us some evidence that the explored learning methods, such as BROOF, LazyNN_RF and BERT do have some complementary information that can potentially be explored in order to come up with more effective learners. This is the main motivation to what we propose here: a novel strategy to stack RF based classifiers that, besides producing highly effective meta-learners, it also enjoys a significantly reduced runtime, guaranteeing its applicability on large classification problems.

$Dis_{i,j}^{\text{norm}}$	BROOF	LAZY	SVM	NB	KNN
BERT	0.12	0.29	0.30	0.32	0.33
BROOF	-	0.29	0.32	0.32	0.33
LAZY	-	-	0.31	0.32	0.19
SVM	-	-	-	0.27	0.30
NB	-	-	-	-	0.33

(a) 4UNI

$Dis_{i,j}^{\text{norm}}$	BROOF	LAZY	SVM	NB	KNN
BERT	0.07	0.18	0.23	0.32	0.25
BROOF	-	0.20	0.23	0.35	0.29
LAZY	-	-	0.25	0.32	0.23
SVM	-	-	-	0.29	0.25
NB	-	-	-	-	0.29

(b) ACM

$Dis_{i,j}^{\text{norm}}$	BROOF	LAZY	SVM	NB	KNN
BERT	0.03	0.10	0.09	0.19	0.23
BROOF	-	0.11	0.08	0.19	0.22
LAZY	-	-	0.07	0.18	0.18
SVM	-	-	-	0.15	0.20
NB	-	-	-	-	0.23

(c) REUT90

$Dis_{i,j}^{\text{norm}}$	BROOF	LAZY	SVM	NB	KNN
BERT	0.15	0.23	0.28	0.39	0.35
BROOF	-	0.26	0.24	0.41	0.42
LAZY	-	-	0.29	0.43	0.38
SVM	-	-	-	0.37	0.36
NB	-	-	-	-	0.39

(d) 20NG

Table 6.1: Normalized Degree of Disagreement

6.3 Proposed Stacking Strategy for Bagged Models

We now introduce an efficient way of stacking bagging-based classifiers. Recall that when stacking classifiers, one usually relies on k fold cross-validation procedures to estimate the *a posteriori* class probabilities for each example, to serve as input for the meta-classifier [Wolpert, 1992; Ting and Witten, 1997; Kuncheva et al., 2001; Nguyen et al., 2016]. Based on these predicted *a posteriori* class distribution estimates, the meta-classifier induces a relationship between these predictions and the true class. However, such estimation strategy may be very costly and sometimes ineffective, since it depends on learning k different models to estimate the probability distributions that serve as input for the stacking procedure. In order to cope with this problem, we rely on the out-of-bag samples produced by the bootstrap technique performed by bagging-based classifiers, such as Random Forests, in order to estimate the *a posteriori* probability distributions for the training samples, thus producing the meta attributes to be fed to the stacked classifier. Since this information is promptly generated at training time by bagged classifiers, our proposed meta learner can thus be built with negligible additional computational effort.

In details, recall that the bootstrap procedure (random sampling with replacement) generates samples $\mathbb{D}_{boot}$ comprising of approximately $1-e^{-1} \approx 63\%$ of the original training set $\mathbb{D}_{train}$, with the remaining 36% samples being the so-called out-of-bag samples [Hastie et al., 2009]. In the bagging training process, this procedure is repeated in order to produce several distinct training sets $\mathbb{D}^j_{boot}$ for building the ensemble composing trees h_j. Thus, we here propose to use $\mathbb{D}^j_{oob} = \mathbb{D}_{train} \setminus \mathbb{D}^j_{boot}$ to estimate the ensemble class probability distribution at a point $x \in \mathbb{D}_{train}$ to be used as meta attributes to train a stacked classifier. This comes at a very low cost, since the meta attributes can be efficiently computed during the training stage of bagged learners, without the needs to perform costly estimation strategies, such as cross-validation. Therefore, let M be the number of bootstrap iterations, $\mathbb{D}^j_{boot}|^M_{j=1}$ be the bootstrap samples and $h_j|^M_{j=1}$ the classifiers trained with the corresponding bootstrap samples. We compute the ensemble OOB probability distribution estimates $p^{oob}(\mathbb{C}|x)$ for each instance $x \in \mathbb{D}_{train}$ as follows:

$$p^{oob}(\mathbb{C}|x) = \frac{\sum_{j=1}^{M} p^{h_j}(\mathbb{C}|x) I[x \in \mathbb{D}^j_{oob}]}{\sum_{j=1}^{M} I[x \in \mathbb{D}^j_{oob}]}, \tag{6.3}$$

where I denotes an indicator function that returns 1 when the m-th classifier did not use x as training instance, 0 otherwise, and $p^{h_j}(\mathbb{C}|x)$ is the class *a posteriori* distribution estimated by h_j. In other words, the OOB *a posterori* class probability distribution $p^{oob}(\mathbb{C}|x)$ is assessed by averaging the class probability distributions $p^{h_j}(\mathbb{C}|x)$, estimated

by each individual tree that was built without using x as training sample.

Thus, in order to make the performance gain more tangible, let $N = |\mathbb{D}_{train}|$ and $Cost^{rf}_{cv-stack}(N)$ be the cost of stacking only one RF by means of K-fold cross-validation. This cost can be expressed as the sum of the cost of training a RF in the entire dataset, training it K times on a slightly reduced dataset and testing it K times, which is showed by the following equation: $Cost^{rf}_{cv-stack}(N) = Cost^{rf}_{train}(N) + K \times Cost^{rf}_{train}(\frac{(K-1)}{K}N) + K \times Cost^{rf}_{test}(\frac{N}{K})$. The first term of the equation is the inherent cost of training an ensemble since we have to train the base learner at least once. However, the last two terms are related to the meta-attributes generation which may be costly depending on the size of the K. In our proposed approach, we can eliminate the last two terms once we are able to estimate the meta-attributes while training the decision trees composing the RF with negligible additional cost since the cost of estimating $p^{h_j}(\mathbb{C}|x)$ is $\Theta(\log(0.63N))$ for decision trees [Louppe, 2014]. Therefore, we have that $Cost^{rf}_{oob-stack}(N) \approx Cost^{rf}_{train}(N)$. In general terms, we can speedup the stacking process up to K times.

Moreover, the estimated *a posteriori* probability $p^{oob}(\mathbb{C}|x)$ can be naturally used in the traditional stacking framework as meta-features, alongside with the meta-features obtained by traditional means (shown in table bellow), in order to train any meta-learner as discussed in the Section 6.1. By doing so, one can stack bagged-based models efficiently either with bagged and non-bagged ones, thus, making feasible the applicability of such stacking systems in real world problems.

$$\begin{bmatrix} p^{oob}_{rf}(\mathbb{C}|x_1) & \dots & p^{oob}_{bert}(\mathbb{C}|x_1) & \dots & p^{cv}_{svm}(\mathbb{C}|x_1) & \dots & p^{cv}_{knn}(\mathbb{C}|x_1) \\ p^{oob}_{rf}(\mathbb{C}|x_2) & \dots & p^{oob}_{bert}(\mathbb{C}|x_2) & \dots & p^{cv}_{svm}(\mathbb{C}|x_2) & \dots & p^{cv}_{knn}(\mathbb{C}|x_2) \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ p^{oob}_{rf}(\mathbb{C}|x_N) & \dots & p^{oob}_{bert}(\mathbb{C}|x_N) & \dots & p^{cv}_{svm}(\mathbb{C}|x_N) & \dots & p^{cv}_{knn}(\mathbb{C}|x_N) \end{bmatrix}$$

Table 6.2: Example of meta-training set generated by mixed approaches. For instance, the first two columns were generated by our OOB approach and the last ones by K-fold cross validation.

6.4 Experimental Evaluation

We now report our experimental evaluation of the proposed stacking model. To this end, we consider all the previously explored datasets regarding topic categorization and sentiment analysis (see Section 5.4.1). We contrast the proposed RF-based stacked classifier against traditional stacking of classical state-of-the-art methods in text cate-

gorization (e.g. SVM, kNN, Naïve Bayes). Then, we analyze our proposal in terms of runtime.

6.4.1 Results and Discussion

As we detail in the following, stacking RF-based classifiers brings substantial improvements in classification effectiveness over traditional methods and their combinations in both analyzed text classification tasks. The proposed RF-based stacking is even able to produce as good results as the combination of all evaluated classifiers in most analyzed datasets, at a much lower runtime, guaranteeing its applicability in text classification tasks.

		20NG	4UNI	ACM
All_{stack}	micF1	**92.21 ± 0.44**	**86.85 ± 1.17**	**79.02 ± 0.72**
	macF1	**92.01 ± 0.42**	**79.43 ± 2.23**	**66.25 ± 1.01**
BROOF+LAZY+BERT+LXT	micF1	90.63 ± 0.57	**86.79 ± 0.86**	**77.83 ± 0.80**
	macF1	90.4 ± 0.57	**79.63 ± 1.91**	**63.42 ± 0.92**
SVM+NB+KNN	micF1	90.65 ± 0.45	84.95 ± 1.15	**77.78 ± 0.73**
	macF1	90.42 ± 0.44	75.86 ± 1.48	**65.08 ± 1.71**
SVM+BERT	micF1	90.85 ± 0.50	**86.65 ± 1.05**	**77.25 ± 0.60**
	macF1	90.68 ± 0.50	**80.13 ± 2.49**	**66.29 ± 0.97**
BERT	micF1	89.45 ± 0.46	84.61 ± 0.98	74.8 ± 0.59
	macF1	89.13 ± 0.58	73.61 ± 1.85	62.1 ± 0.99
SVM	micF1	90.06 ± 0.43	83.48 ± 1.08	75.4 ± 0.66
	macF1	89.93 ± 0.43	73.39 ± 2.17	**63.84 ± 0.55**

(a) Datasets: 20NG, 4UNI and ACM

		REUT90	SPAM	MED
All_{stack}	micF1	**80.76 ± 1.24**	**96.06 ± 0.78**	**88.76 ± 0.11**
	macF1	**39.28 ± 1.14**	**95.87 ± 0.82**	**81.62 ± 0.34**
BROOF+LAZY+BERT+LXT	micF1	**80 ± 1.60**	**96.13 ± 0.86**	87.10 ± 0.07
	macF1	**38.66 ± 2.85**	**95.94 ± 0.90**	79.36 ± 0.59
SVM+NB+KNN	micF1	**78.53 ± 1.09**	93.67 ± 0.34	87.96 ± 0.05
	macF1	37.10 ± 1.41	93.37 ± 0.34	80.33 ± 0.57
SVM+BERT	micF1	**78.43 ± 1.33**	**95.91 ± 0.79**	87.65 ± 0.03
	macF1	36.8 ± 2.45	**95.72 ± 0.82**	80.55 ± 0.52
BERT	micF1	67.33 ± 0.72	**96.11 ± 0.52**	83.68 ± 0.32
	macF1	29.24 ± 1.40	**95.93 ± 0.55**	74.25 ± 0.37
SVM	micF1	68.19 ± 1.15	92.55 ± 0.8	86.19 ± 0.05
	macF1	31.95 ± 2.59	92.12 ± 0.87	78.46 ± 0.42

(b) Datasets: REUT90, SPAM and MED

Table 6.3: Topic categorization - Obtained results for stacking and top-performer base classifiers.

		AMAZON	BBC	DEBATE	DIGG	MYSPACE
All_{stack}	micF1	**75.6**	**86.84**	**80.14**	**76.47**	**86.1**
	macF1	**74.44**	**51.56**	**77.86**	**65.8**	**68.27**
BROOF+LAZY+BERT+LXT	micF1	**75.6**	**86.84**	**79.89**	**75.96**	**86.94**
	macF1	**74.44**	50.03	**77.41**	**62.71**	**68.88**
SVM+NB+KNN	micF1	**75.37**	**86.7**	**78.83**	**77.5**	**86.09**
	macF1	**74.14**	46.44	**76.71**	**64.84**	**65.98**
SVM+BERT	micF1	**75.96**	**86.7**	**79.99**	**76.46**	**86.11**
	macF1	**74.85**	47.34	**77.88**	**63.05**	**67.14**
BERT	micF1	**75.76**	**86.04**	**79.38**	**75.7**	**86.46**
	macF1	**74.16**	**53.78**	**77.24**	**63.03**	**67.35**
NB	micF1	74.68	**86.84**	76	**76.09**	85.5
	macF1	73.32	46.48	73.7	**64.38**	60.39

(a) Datasets: AMAZON, BBC, DEBATE, DIGG and MYSPACE

		NYT	TWEETS	TWITTER	YELP	YOUTUBE
All_{stack}	micF1	**68.1**	**89.16**	**75.71**	**94.82**	**83.93**
	macF1	**67.35**	**87.08**	**74.83**	**94.82**	**81.11**
BROOF+LAZY+BERT+LXT	micF1	**67.79**	**89.01**	**76.23**	**94.22**	81.5
	macF1	**66.87**	**86.63**	**75.26**	**94.22**	77.63
SVM+NB+KNN	micF1	**67.41**	**88.35**	**75.32**	93.84	**84.13**
	macF1	**66.53**	**86.07**	**74.24**	93.84	**81.05**
SVM+BERT	micF1	**67.73**	**88.25**	**75.89**	**94.64**	81.42
	macF1	**66.92**	**85.81**	**74.79**	**94.64**	77.85
BERT	micF1	**68.14**	**88.3**	**76.15**	**94.26**	79.85
	macF1	**67.01**	**85.49**	**74.71**	**94.26**	76.47
NB	micF1	**67.1**	86.82	**74.88**	90.44	**83.43**
	macF1	66.06	84.4	**73.89**	90.44	**80.3**

(b) Datasets: NYT, TWEETS, TWITTER, YELP and YOUTUBE

Table 6.4: Sentiment analysis - Obtained results for stacking and top-performer base classifiers.

The results regarding the topic categorization task can be found in Table 6.3. First, note that the combination of RF-based approaches $BERT + BROOF + LXT + LAZY$ achieves better results over the two best base classifiers (BERT and SVM) in most datasets, with substantial gains of up to 17% and 21% in MacroF$_1$ and MicroF$_1$, respectively. In fact, stacking RF-based approaches clearly produces superior results when compared to the top-notch SVM and BERT classifiers. The combination $BERT + BROOF + LXT + LAZY$ is also superior to the combination of classifiers $SVM + NB + KNN$ in 4UNI and SPAM. These results provide evidence to our claim that stacking RF-based methods for text classification is a strong alternative to the stacking of traditional text classification approaches, since besides effective they are highly parallelizable, easily parameterized and efficiently combined by our OOB stacking proposal.

Furthermore, stacking RF-based methods also performs as well as the combina-

tion of all classification approaches (i.e., All_{stack}) in all but 20NG and MED datasets. This indicates that traditional classification methods may provide supplementary discriminative information for RF-based methods, which is the expected since they follow completely different classification paradigms.

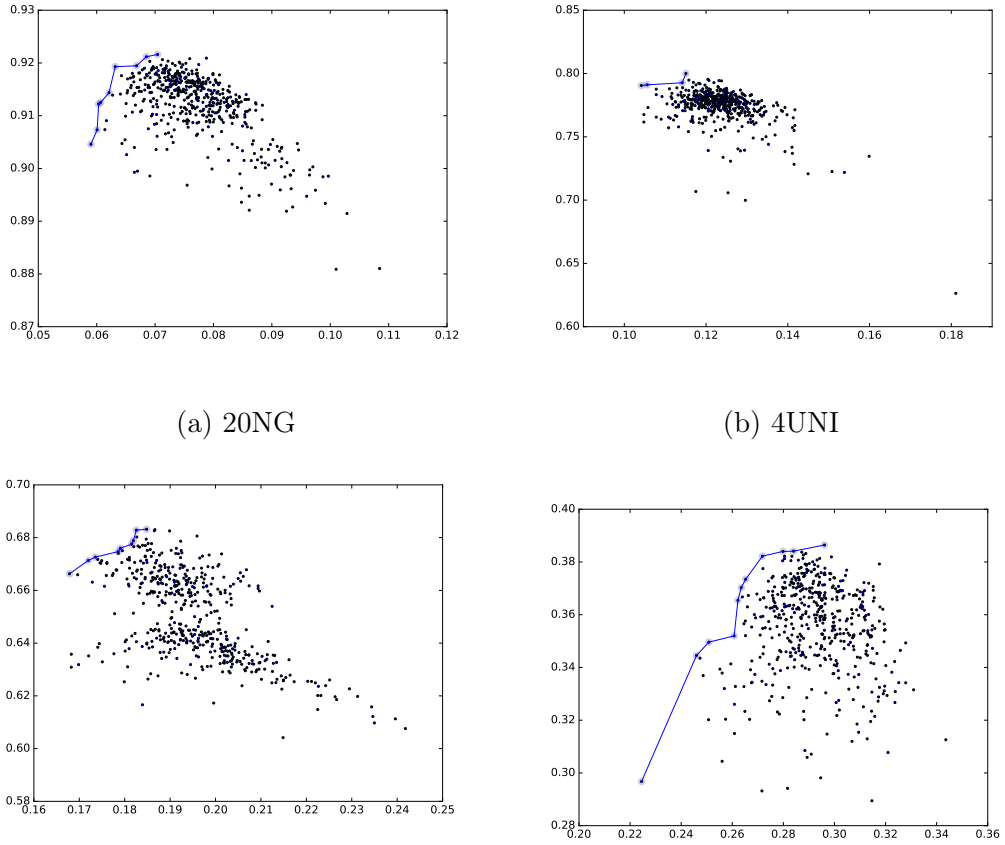
Considering the sentiment analysis datasets, one can observe that most of the results regarding stacking different classifiers are statistically tied, as shown in Table 6.4. Moreover, BERT is not substantially improved when stacked with other classifiers. In fact, $BERT$ ties with $BROOF+LAZY+BERT+LXT$ in all datasets. Also, $BERT$ also ties with All_{stack} in all datasets but one, YOUTUBE (in fact, due to the presence of the NB classifier in the All_{stack} ensemble). These results show that, despite the potential benefits of stacking various methods, the proposed $BERT$ classifier remains as one of the top performers when detecting the sentiment of textual documents, being a strong candidate for consideration.

6.4.2 Effectiveness vs. diversity tradeoff

In the following, we perform an analysis of the Pareto's frontier of the generated ensembles. With this, we want to confirm that the RF-based methods are important to generate ensembles of classifiers with high generalization power and complementarity/diversity.

Figure 6.1 exhibits a scatter plot for several datasets, in which each point represents a stacking ensemble out of the possible ensembles generated by the combination of the nine base classifiers (number of classifiers composing the ensemble varies from 2 to 9). The x-axis represents the diversity metric *Double-Fault*, which was chosen because it is the pairwise metric that best represents the relationship between accuracy and diversity of an ensemble system [Kuncheva and Whitaker, 2003]. The metric was originally used by Giancinto and Roli [2001] to form a pairwise diversity matrix for a classifier pool and subsequently to select classifiers that are least related. The metric estimates the likelihood of both classifiers incorrectly classifying the same sample (the smaller the value of the metric, the higher the diversity/complementarity). The y-axis is the effectiveness metric MacroF_1. For those points, we calculate the Pareto's frontier which maximizes the effectiveness metric and minimizes the Double-Fault metric (maximizes diversity), which are desirable characteristics of any ensemble method (highly diverse and effective). The ensembles in the frontier showed in Figures 6.1 (a), (b), and (c) are the following:

- $(LXT, NB), (LAZY, NB), (SVM, KNN), (LXT, SVM, NB)$, $(LAZY, SVM, NB), (LXT, SVM, NB, KNN)$,



(a) 20NG (b) 4UNI

(c) ACM (d) REUT90

Figure 6.1: MacroF_1 vs. *Double-Fault* - each point represents a stacking out of all possible combination of the 9 base classifiers. The line connecting the highlighted points is the Pareto's frontier.

$(LXT, SVM, NB, KNN, XT), (LAZY, BERT, LXT, SVM, NB, KNN),$
$(LAZY, LXT, SVM, NB, KNN, FA)$

- $(BERT, SVM), (BERT, LXT, SVM), (BROOF, BERT, SVM, KNN), (BERT, SVM, NB)$
- $(SVM, NB), (LAZY, SVM, NB), (BERT, SVM, NB),$
  $(LAZY, BERT, SVM, NB), (LAZY, BERT, LXT, SVM, NB),$
  $(LAZY, BERT, SVM, NB, KNN), (BROOF, LAZY, SVM, NB, KNN),$
  $(BERT, LXT, SVM, NB, AEA), (BROOF, BERT, LXT, SVM, NB, KNN)$
- $(SVM, NB), (BERT, SVM, NB), (BROOF, SVM),$
  $(BROOF, LAZY, SVM, NB), (BROOF, BERT, SVM, NB),$
  $(BERT, LXT, SVM, NB), (BERT, SVM, NB, AEA),$
  $(BROOF, BERT, LXT, SVM, NB), (BROOF, BERT, LXT, SVM),$
  $(BROOF, BERT, SVM, NB, FA, AEA),$
  $(BROOF, LAZY, BERT, SVM, KNN, FA, AEA)$

One can notice that as we go from left to right on the Pareto's frontier both,

effectiveness and diversity[1] decreases. This implies that not necessarily a highly diverse combination will result in more accurate classifiers, but that there is a tradeoff between effectiveness and diversity of the base classifiers of the ensemble. By analyzing the Pareto's frontier, one can clearly notice that the RF extensions play an important role in providing a balance between diversity and generalization power in the most effective ensembles. It is also possible to note that the loss in diversity brought by the insertion of more similar methods into the ensemble (e.g., RF-based ones) is compensated by the high generalization power of those methods. These results along with the effectiveness analysis corroborate our hypothesis: by using RF-based algorithms in multi-classifier systems, it is possible to achieve higher effectiveness.

6.4.3 Computational Time and Effectiveness to Stack Bagging-based Methods

Table 6.5 shows the average time to combine all nine base learning algorithms using ours and the baseline stacking approaches, which generate the meta-level data by means of 5-fold cross-validation. We also contrast our proposed method with a fixed combining method (simple voting), which is the fastest possible combining strategy since it does not require training. As can be seen in Table 6.5, the stacking using our strategy based on Out-of-Bag (OOB) samples combined with RF as meta-level learner shows significant speedups in relation to all other stacking strategies, without degrading its predictive performance. Specifically, in the MEDLINE dataset, the stacking approaches using cross-validation were not able to handle the dataset in a suitable time. Moreover, one can notice that our approach is competitive regarding execution time when compared to a simple voting algorithm while excelling in effectiveness.

	20NG	4UNI	ACM	REUT90	MEDLINE
OOB-RF	**7244**	**2413**	**6762**	**13163**	**458013**
RF	20479	7358	17194	29256	-
MLR	21455	7334	17170	29131	-
DT	20448	7327	17163	27124	-
SCANN	21651	7330	17166	29007	-
VIG	20620	7499	17335	30296	-
Voting	2921	1046	2451	4017	375297

Table 6.5: Avg. time in seconds to combine (training + testing time) all 9 base learning algorithms with different stacking strategies. In the cases that a method was not able to handle a dataset, we marked the corresponding table cell with '-'.

[1]In this Section, we will use the terms diversity and complementarity interchangeably.

Table 6.6 shows the effectiveness of different combination strategies, such as MLR, DT, SCANN and the recently proposed VIG in the largest datasets[2]. The results of these strategies are never superior to the use of RF as a meta-level combiner. Despite not being designed as a meta-level combiner, RF is capable of achieving the best effectiveness results due to its capabilities of automatically identifying discriminative patterns on the relationship among classifier results. Moreover, the proposed usage of OOB can provide the same discriminative evidence as the output of the RF-based methods obtained by means of cross-validation, which eliminates the need for costly procedures to stack RF-based methods.

		20NG	4UNI	ACM	REUT90	MED
OOB-RF	microF1	**92.24**	**86.87**	**78.99**	**80.25**	**88.76**
	macroF1	**92.04**	**79.35**	65.75	**40.8**	**81.62**
RF	microF1	**92.21**	**86.85**	**79.02**	**80.76**	-
	macroF1	**92.01**	**79.43**	66.25	**39.28**	-
MLR	microF1	91.36	85.34	77.99	76.33	-
	macroF1	91.14	76.62	**68.26**	35.64	
DT	microF1	91.44	83.82	77.53	68.19	-
	macroF1	91.23	76.11	65.07	32.43	-
VOTING	microF1	91.55	84.27	77.7	66.71	86.91
	macroF1	91.29	72.2	64.31	28.89	77.41
SCANN	microF1	90.87	85.31	76.84	71.14	-
	macroF1	90.55	73.88	63.08	29.25	-
VIG	microF1	90.13	82.14	75.11	19.4	-
	macroF1	89.65	74.33	65.49	3.17	-

Table 6.6: Obtained results for different stacking strategies. In the cases that a method was not able to handle a dataset, we marked the corresponding table cell with '-'.

[2]This excludes the small SPAM dataset.

Chapter 7

Conclusions and Future Work

In this work, we propose a boosted version of the extremely randomized trees classifier, named BERT, in order to leverage the learner's capability to minimize bias while maintaining high predictive power by properly reducing variance. As our experimental analysis reveal, our proposal enjoys top-notch classification effectiveness, being among the top performers in the vast majority of cases covering two challenging text classification tasks, namely, topic categorization and sentiment analysis. We also propose to stack the explored RF-based classifiers in order to exploit the complementarity observed among those classifiers. Unlike traditional stacking, that makes use of cross-validation procedures to learn the meta-features to be fed to the stacking procedure, we here rely on the out-of-bag samples obtained through bootstrapping the training set when learning the forests. More specifically, the out-of-bag samples are used to estimate the *a posteriori* class distributions used by the stacking procedure to learn the underlying input/output relationships. We show that such novel stacking approach is not only able to provide state-of-the-art classification effectiveness, but also at a significantly lower runtime.

7.1 Main Contributions

We advance the state-of-the-art in text classification by proposing a novel derivation of the RF classifier. More specifically, we propose a new boosted version of the RF classifier, based on some ideas explored by the BROOF classifier: the so-called Boosted Extremely Randomized Trees (BERT) classifier. While BROOF is able to mitigate the overfitting issue faced by RF classifier when applied to high-dimensional noisy data, by avoiding the generation of overly complex trees, it offers limited capability of bias reduction (through the so-called selective out-of-bag based weight update strategy)

as shown by Salles et al. [2017]. Thus, bias may still pose as an important factor to contribute to the error rate. To tackle this potential issue, we here propose to introduce another source of randomization in the boosted strategy proposed by Salles et al. [2015]in order to achieve a better bias-variance tradeoff, by building tree in a more extreme fashion as proposed by Geurts et al. [2006]. This novel strategy has the following motivations: (*i*) we expect to avoid overly complex models (and thus mitigate overfitting) through the application of the BROOF-like strategies; (*ii*) to provide more control over the learner's bias through the additional randomization offered by building extremely randomized trees [Geurts et al., 2006]; and, finally, (*iii*) to exploit the fact that the extremely randomized trees have shown to be more robust to noise than the RF classifier. Our proposed classifier outperforms the analyzed state-of-the-art classifiers, including SVM, kNN, Naïve Bayes, BROOF, and LazyNN_RF.

Moreover, motivated by the fact that distinct learning methods may complement each other [Kuncheva and Whitaker, 2003], uncovering specific structures that underlie the input/output relationship of the data at hand, in this work we also propose to exploit the complementary characteristics of the recently proposed RF-based approaches and ours, by stacking them in order to learn an even more effective meta-classifier. Their level of disagreement is high, which motivates our idea. Up to our knowledge, this is the first attempt to combine the three main ensemble strategies: bagging, boosting and stacking.

Finally, when stacking classifiers, one usually relies on k fold cross-validation procedures to estimate the *a posteriori* class probabilities for each example, to serve as input for the meta-classifier [Wolpert, 1992; Ting and Witten, 1997; Kuncheva et al., 2001; Nguyen et al., 2016]. Based on these predicted *a posteriori* class distribution estimates, the meta-classifier induces a relationship between these predictions and the true class. However, such estimation strategy may be very costly and sometimes ineffective, since it depends on learning k different models to estimate the probability distributions that serve as input for the stacking procedure. In order to cope with this problem, we show that we avoid additional computation efforts to learn a stacked classifier by exploiting the efficient and unbiased out-of-bag (OOB) error estimate, an out-of-the-box estimate naturally produced by the bootstrap procedure used in each RF-based learner.

7.2 Future Work

Clearly, there is still room for improvements. Regarding the BERT strategy, we plan to investigate the benefit of out-of-bag error estimate applied to a more sophisticated early stop strategy in order to avoid unnecessary boosting iterations, which may lead the model to overfitting. In the context of runtime performance, BERT, as well as BROOF, are costly to train since they inherent the burden of boosting (sequential, hardly parallelizable) and Random Forests (extremely deep trees in high-dimensional space with many noise/irrelevant attributes). Hence, tackling these issues is paramount to the applicability of these methods. In order to take fully advantage from parallelizable potential of the Random Forests (Extra-Trees) built at each iteration of BROOF (BERT), it is worth to invest in alternative parallel formulations for the tree building process as the one proposed by Jansson et al. [2014]. Focusing on high-dimensional data with many noise/irrelevant attributes, an alternative to avoid extremely deep trees(caused by repeatedly selecting irrelevant attributes while splitting the nodes) may be achieved by means of weighted sampling for selecting the subset of candidate features. The weight of the candidates features are assessed, for example, by means of chi-square test [Xu et al., 2012]. This approach may considerably speedup the process of building the trees in high-dimensional data leading to more compact trees since the weighted sampling increases the likelihood of selecting more discriminative features at each split.

Regarding the bagging-based stacking strategy, we plan to investigate if non-bagging strategies, such as the traditional kNN, Naïve Bayes and SVM classifier, could benefit from the bootstrap procedure in order to come up with a uniform stacking strategy that generalizes to any other classifier. Also, in the same vein of exploring the out-of-bag samples to estimate the *a posteriori* class distributions in our stacking approach, we could explore the out-of-bag error estimates in order to select the candidate features for decision nodes in order to improve both effectiveness and efficiency of the RF based classifiers, such as BERT, BROOF, LazyNN_RF, LXT and the traditional RF.

Bibliography

Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley. ISBN 0-201-39829-X.

Bauer, E. and Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1):105--139. ISSN 1573-0565.

Bi, Y., Bell, D., Wang, H., Guo, G., and Guan, J. (2007). Combining multiple classifiers using dempster's rule for text categorization. *Appl. Artif. Intell.*, 21(3):211--239. ISSN 0883-9514.

Breiman, L., F.-J. S. C. and Olshen, R. (1984). *Classification and Regression Trees. The Wadsworth and Brooks-Cole statistics- probability series*. Taylor Francis.

Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, 24(2):123--140. ISSN 0885-6125.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32. ISSN 0885-6125.

Campos, R., Canuto, S., Salles, T., de Sá, C. C. A., 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 2017, Shinjuku, Tokyo, Japan, August 07-11, 2017*.

Campos, R. R. and Gonçalves, M. A. (2016). Bert: Melhorando classicação de texto com Árvores extremamente aleatórias, bagging e boosting. In *31st of the Brazilian Symposium on Databases, Salvador, Brazil, October 04-07, 2016*.

Campos, R. R., Gonçalves, M. A., and Salles, T. C. (2016). Quando a amazônia encontra a mata atlântica: Empilhamento de florestas para classificação efetiva de texto. In *4th Symposium on Knowledge Discovery, Mining and Learning, Recife, Pernambuco, Brazil, October 09-11, 2016*.

Canuto, S., Gonçalves, M., Santos, W., Rosa, T., and Martins, W. (2015). An efficient and scalable metafeature-based document classification approach based on massively parallel computing. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 333--342, New York, NY, USA. ACM.

Cord, M., Cunningham, P., Cord, M., and Cunningham, P. (2008). *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval (Cognitive Technologies)*. Springer-Verlag TELOS, Santa Clara, CA, USA, 1 edition. ISBN 354075170X, 9783540751700.

Danesh, A., Moshiri, B., and Fatemi, O. (2007). Improve text classification accuracy based on classifier fusion methods. In *Information Fusion, 2007 10th International Conference on*, pages 1–6. IEEE.

de Sá, C. C. A., Gonçalves, M. A., de Sousa, D. X., and Salles, T. (2016). Generalized BROOF-L2R: A general framework for learning to rank based on boosting and random forests. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 95--104.

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

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1):3133--3181. ISSN 1532-4435.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119--139. ISSN 0022-0000.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63(1):3–42. ISSN 0885-6125.

Giancinto, G. and Roli, F. (2001). Design of effective neural network ensembles for image classification purposes. *IMAGE VISION AND COMPUTING JOURNAL*, 19:699--707.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10--18. ISSN 1931-0145.

Hao Chen, T. K. H. (1999). Evaluation of decision forests on text categorization.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning*. Springer.

Hutto, C. and Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text.

Jansson, K., Sundell, H., and Boström, H. (2014). gpurf and gpuert: Efficient and scalable gpu algorithms for decision tree ensembles. In *Proceedings of the 2014 IEEE International Parallel & Distributed Processing Symposium Workshops*, IPDPSW '14, pages 1612--1621, Washington, DC, USA. IEEE Computer Society.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features.

Kanakaraj, M. and Guddeti, R. M. R. (2015). Performance analysis of ensemble methods on twitter sentiment analysis using nlp techniques. In *IEEE International Conference on Semantic Computing (ICSC)*. IEEE.

Khan, A., Baharudin, B., Lee, L. H., Khan, K., and Tronoh, U. T. P. (2010). A review of machine learning algorithms for text-documents classification. In *Journal of Advances In Information Technology*. Academy Publisher.

Kuncheva, L. I. (2002). A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286. ISSN 0162-8828.

Kuncheva, L. I., Bezdek, J. C., and Duin, R. P. W. (2001). Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34:299--314.

Kuncheva, L. I. and Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach. Learn.*, 51(2):181--207. ISSN 0885-6125.

Lewis, D. D. (1995). Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research*

and Development in Information Retrieval, SIGIR '95, pages 246--254, New York, NY, USA. ACM.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361--397. ISSN 1532-4435.

Louppe, G. (2014). *Understanding Random Forests: From Theory to Practice*. PhD thesis, University of Liège.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA. ISBN 0521865719, 9780521865715.

Merz, C. J. (1999). Using correspondence analysis to combine classifiers. *Mach. Learn.*, 36(1-2):33--58. ISSN 0885-6125.

Nguyen, T. T., Nguyen, T. T. T., Pham, X. C., and Liew, A. W.-C. (2016). A novel combining classifier method based on variational inference. *Pattern Recogn.*, 49(C):198--212. ISSN 0031-3203.

Onan, A., Korukoğlu, S., and Bulut, H. (2016). Ensemble of keyword extraction methods and classifiers in text classification. *Expert Syst. Appl.*, 57(C):232--247. ISSN 0957-4174.

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.

Pui, G., Fung, C., Yu, J. X., Wang, H., Cheung, D. W., and Liu, H. (2006). A balanced ensemble approach to weighting classifiers for text classification. In *ICDM '06. Sixth International Conference on Data Mining*. IEEE.

Quinlan, J. R. (1986). Induction of decision trees. *Mach. Learn.*, 1(1):81--106. ISSN 0885-6125.

Quinlan, J. R. (1996). Bagging, boosting, and c4.5. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725--730. AAAI Press.

Rocha, L., Mourão, F., Pereira, A., Gonçalves, M. A., and Meira, Jr., W. (2008). Exploiting temporal contexts in text classification. In *Proc. CIKM*, pages 243--252.

Rokach, L. (2009). Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. In *Computational Statistics Data Analysis, In Press, Corrected Proof.*

Salles, T., Gonçalves, M., and Rocha, L. (2017). Phd dissertation: Random forest based classifiers for classification tasks with noisy data. Federal University of Minas Gerais.

Salles, T., Gonçalves, M., Rodrigues, V., and Rocha, L. (2015). Broof: Exploiting out-of-bag errors, boosting and random forests for effective automated classification. In *Proc. of the 38th International ACM SIGIR Conference on Inf. Retrieval*, pages 353--362. ACM.

Segal, M. R. (2004). Machine learning benchmarks and random forest regression. Technical report, University of California.

Skalak, D. B. (1996). The sources of increased accuracy for two proposed boosting algorithms. In *In Proc. American Association for Arti Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*, pages 120--125.

Thelwall, M., Buckley, K., and Paltoglou, G. (2012). Sentiment strength detection for the social web. *J. Am. Soc. Inf. Sci. Technol.*, 63(1):163--173. ISSN 1532-2882.

Therneau, T., Atkinson, B., and Ripley, B. (2015). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-10.

Ting, K. M. and Witten, I. H. (1997). Stacking bagged and dagged models. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 367--375, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Wolpert, D. and Macready, W. G. (1996). Combining stacking with bagging to improve a learning algorithm. Technical report.

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

Wolpert, D. H. and Macready, W. G. (1999). An efficient method to estimate bagging's generalization error. *Machine Learning*, 35(1):41--55. ISSN 1573-0565.

Xu, B., Guo, X., Ye, Y., and Cheng, J. (2012). An improved random forest classifier for text categorization. *JCP*, 7(12):2913–2920.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Inf. Ret.*, 1:69–90. ISSN 1386-4564.

Appendix A

A.1 Normalized Degree of Disagreement

Theorem A.1.1. *Let R_i and R_j be the accuracy rates of any two classifiers h_i and h_j. The minimal degree of disagreement that any two classifiers may have, given their accuracy rates, is defined by $Dis_{i,j}^{min} = R_i + R_j - 2\min(R_i, R_j)$.*

Proof. Let N_{10}, N_{01}, N_{00} and N_{11} be, respectively, the rate of instances correctly classified by h_i and misclassified by h_j, the rate of instances correctly classified by h_j and misclassified by h_i, the rate of instances misclassified by both and the rate of instances correctly classified by both. In order to have minimal degree of disagreement, we must have $N_{01} = 0$ or $N_{10} = 0$. Also, we have that h_i and h_j accuracy can be expressed as $R_i = N_{10} + N_{11}$ and $R_j = N_{01} + N_{11}$.

Thus, if $N_{10} = 0$ we have that $N_{11} = R_i$ and $R_i \leq R_j$ since $N_{11} \leq N_{01} + N_{11} \; \forall \; N_{01} \geq 0$. Analogously, if $N_{01} = 0$ we have that $N_{11} = R_j$ and $R_j \leq R_i$ since $N_{11} \leq N_{10} + N_{11} \; \forall \; N_{10} \geq 0$.

Therefore, we have that

$$N_{11} = \min(R_i, R_j) \tag{A.1}$$

Knowing that $Dis_{i,j} = N_{10} + N_{01}$, we can rewrite it in function of the accuracy rates so that we have that $Dis_{i,j} = R_i + R_j - 2 \times N_{11}$.

When the degree of disagreement is minimal, N_{11} is defined by the equation A.1, hence:

$$\mathrm{Dis}_{i,j}^{min} = R_i + R_j - 2\min(R_i, R_j) \tag{A.2}$$

□

Theorem A.1.2. *Let R_i and R_j be the accuracy rates of any two classifiers h_i and h_j. The maximal degree of disagreement that any two classifiers may have, given their accuracy rates, is defined by $Dis_{i,j}^{\max} = \min(R_i + R_j, 2 - R_i - R_j)$.*

Proof. Let N_{10}, N_{01}, N_{00} and N_{11} be, respectively, the rate of instances correctly classified by h_i and misclassified by h_j, the rate of instances correctly classified by h_j and misclassified by h_i, the rate of instances misclassified by both and the rate of instances correctly classified by both. We maximize the degree of disagreement when N_{00} and N_{11} tend to 0. Also, we have that h_i and h_j accuracy can be expressed as $R_i = N_{10} + N_{11}$ and $R_j = N_{01} + N_{11}$.

Thus, if $N_{00} \geq 0$ and $N_{11} = 0$ then $1 - N_{00} = R_i + R_j$. Since $1 - N_{00} \leq 1 \ \forall \ N_{00} \geq 0$ we have that

$$N_{11} = 0 \ \text{ and } \quad R_i + R_j \leq 1 \tag{A.3}$$

Analogously, if $N_{11} \geq 0$ and $N_{00} = 0$ then $1 + N_{11} = R_i + R_j$. Since $1 + N_{11} \geq 1 \ \forall \ N_{11} \geq 0$ we have that

$$N_{11} = R_i + R_j - 1 \ \text{ and } \quad R_i + R_j \geq 1 \tag{A.4}$$

Knowing that $Dis_{i,j} = N_{10} + N_{01}$, we can rewrite it in function of the accuracy rates so that we have that $Dis_{i,j} = R_i + R_j - 2 \times N_{11}$.

When the degree of disagreement is maximal, N_{11} is defined by the equation A.3 or A.4 given their conditions, hence:

$$\text{Dis}_{i,j}^{max} = \begin{cases} R_i + R_j & \text{if } R_i + R_j \leq 1 \\ 2 - R_i - R_j & \text{otherwise} \end{cases} \tag{A.5}$$

We have that $R_i + R_j \leq 2 - R_i - R_j$ when $R_i + R_j \leq 1$. Conversely, $2 - R_i - R_j \leq R_i + R_j$ when $R_i + R_j \geq 1$. Therefore, we can rewrite the equation A.5 in a single one:

$$\text{Dis}_{i,j}^{max} = \min(R_i + R_j, 2 - R_i - R_j) \tag{A.6}$$

□