Autotext: AutoML for Text Classification

by

Jorge Gustavo Madrid Pérez

Dissertation submitted in partial fulfillment of the requirements for the degree of

MSc. in Computer Science

at the

Instituto Nacional de Astrofísica, Óptica y Electrónica

August, 2019

Tonantzintla, Puebla, Mexico

Advisor:

Hugo Jair Escalante

Coordination of Computer Science

INAOE, Mexico

Contents

List of Figures

List of Tables

List of acronyms

AutoML Automated Machine Learning

BO Bayesian Optimization

BOW Bag Of Words

DL Deep Learning

GP Gaussian Process

HPO Hyper-Parameter Optimization

KNN K Nearest Neighbors

LDA Latent Dirichlet Allocation

LIWC Linguistic Inquiry and Word Count

LSA Latent Semanteic Allocation

MI Mutual Information

ML Machine Learning

NLP Natural Language Processing

RF Random Forest

SOTA State-Of-The-Art

SVM Support Vector Machine

W2V Word To Vec

XGB eXtreme Gradient Boosting

Acknowledgments

Esta investigación fue posible gracias a CONACyT a través de la beca 634936. Quisiera agradecer a todas las personas involucradas en el desarrollo de este proyecto. Primero, mi más sincero agradecimiento al Dr. Hugo Jair por su continuo apoyo y guía durante todo el proceso, sus comentarios y cuestionamientos han ampliado mi conocimiento y mi formación. Agradezco a todos mis profesores en el INAOE, especialmente, las observaciones, guía y conversaciones de mis sindoles: los doctores Ariel Carrasco, Luis Villaseor y Manuel Montes.

Este tiempo no hubiera sido lo mismo sin mis compañeros de maestría. Gracias por sus consejos y amistad.

Finalmente, esta tesis está dedicada con mucho cariño a mis padres y a mi hermana por su apoyo incondicional durante todos estos años; y a Brenda por creer en mí desde el inicio y animarme siempre.

Abstract

Non-experts in Machine Learning research have an increasing demand for easy-to-use methods to model solutions that use the large amounts of data available today, where such solutions are expected to perform at least as well as one build by a human with profound knowledge of ML and statistics. AutoML is the area that investigates the automation of Machine Learning. Some state-of-the-art methods for approaching this problem are already available, nonetheless none of them concentrate on the challenges of Natural Language Processing.

This work comprises an extensive study in text classification where 81 different problems were approached with the most commonly used algorithms. Leveraging the obtained metadata from these experiments, a method that automatically builds pipelines for classifying text documents is proposed. This method contemplates the optimization of a classification model and its hyper-parameters as well as the selection of the representation vector for a text in a given set of unprocessed text documents. A characterization for tasks is introduced as part of this method, but the reach of this novel description is not limited to AutoML problems.

Results in our experimentation show that the proposed AutoML method and the novel characterization outperform previous approaches for automating text classification and under certain circumstances obtain comparable results to state-of-the-art models. Being one of the first works to explore AutoML in NLP, several further questions can be derived from this thesis with potential impact for both fields.

Resumen

Personas con poca experiencia en aprendizaje de máquina tienen una creciente demanda de métodos fáciles de usar para modelar soluciones que aprovechen la gran cantidad de datos disponible en la actualidad, se espera que dichas soluciones funcionen al menos tan bien como una construida por un humano con profundo conocimiento en aprendizaje de máquina y estadística. AutoML es el área que investiga la *automatización* del aprendizaje de máquina. En el estado del arte existen algunos métodos disponibles que abordan dicho problema, sin embargo, ninguno de ellos se concentra en los retos de Procesamiento de Lenguaje Natural.

Este trabajo comprende un extenso estudio en classificación de textos donde 81 problemas son aborados usando los algoritmos más comunmente usados. Aprovechando los meta-datos obtenidos de estos experimentos, un método que construye automáticamente *pipelines* para clasificar documentos de texto es propuesto. Dicho método contempla la optimización de un modelo de clasificación así como la selección de una representación vectorial para los textos dado un conjunto de documentos sin preprocesamiento alguno. Una nueva caracterización para las tareas de clasificación es introducida como parte del método, pero como se muestra en este documento, el alcance de dicha forma de descripción no está limitada al problema de AutoML.

Los resultados obtenidos en nuestra experimentación muestran que el método de AutoML propuesto y la caracterización superan los resultados de enfoques anteriores para la automatización de tareas de clasificación de texto, y bajo ciertas cir-

cunstancias obtienen resultados comparables a modelos del estado del arte. Siendo uno de los primeros trabajos en explorar AutoML en NLP, varias preguntas pueden derivarse de esta tesis teniendo así impacto en ambas áreas.

Chapter 1

Introduction

1.1 Motivation

In an ever growing number of domains, Machine Learning has facilitated the solution to different problems, from face recognition to service robots, from financial to healthcare, ML has produced a number of pragmatic and powerful tools. The recent success of these applications has resulted in the development of systems that are seen as *intelligent* and has made Artificial Intelligence to gain attention from different research areas and businesses.

Nowadays the success of Machine Learning relies on manually modeling a complex pipeline which consists of a series of steps that receive a set of data, process it, and make predictions for solving a problem. The success of the pipeline itself depends on the availability of 2 crucial factors: data and human expertise. The first one has become easier to find thanks to the the development in communication technology and storage systems over the past few years. Nevertheless, the number of areas and disciplines that could benefit from ML is huge and human experts with profound knowledge in data science and an specific problem are still scarce in most

of them.

In addition, to manually design an ML pipeline consumes a lot of resources for a highly trained team of experts, it is required to suitably define the problem, and build a first pipeline with a pre-processing method for the data, a classification or regression model and a set of values for the hyper-parameters for such model. This pipeline is usually tested several times and fine tuned by modifying either the algorithms or their parameters or both. This task quickly becomes monotonous and is immensely time consuming for any business or research team, efforts of such group of experts should rather be focused on more complex and rewarding tasks.

Thus, the increasing demand for off-the-shelf tools. According to the *no free lunch theorem* [Wolpert et al., 1997] there isn't a single classification algorithm that outperforms the rest of the algorithms, it is safe to assume that the same applies to ML pipelines, as a consequence, for an unseen set of data a new pipeline has to be built to obtain optimal performance. The automation of the complete process for finding the *best* pipeline is known as AutoML [Guyon et al., 2015]. An ideal AutoML system would work as a black-box that, without any human intervention, receives as input a set of data and produces predictions as output.

Because of the recent interest in this area, the first successful systems have already been produced and are available for the public, for instance, there are some commercial solutions from some of the largest web services companies: Google Cloud AutoML, Microsoft AzureML and Amazon Web Services (with H20.ai). There are also some open source libraries for different programming language which have shown achievements in different aspects of AutoML: auto-sklearn [Feurer et al., 2015a], auto-weka [Thornton et al., 2013], TPOT [Olson et al., 2016a, Olson et al., 2016b], and auto-Keras [Jin et al., 2019]. Despite their success, such solutions make significant limitations to their systems, namely, the type of data is restricted to one particular domain or it is expected to be presented in a tabular form after some

pre-processing.

It is also worth considering the fact that the *raw* data (i.e. the format in which it was originally generated) is completely different between tasks, for instance, data can be text documents, images, human speech, financial data, etc. The techniques used for a particular problem change considerably from one domain to another. Such difference introduces a lot of challenges for AutoML and has motivated researchers to specialize in one specific domain or to start from the assumption that data has been already pre-processed, which is the case of the examples mentioned above.

This work recognizes the difficulty of creating a general AutoML system, able to work with any type of dataset for any type of problem, so it targets one of the most popular areas related to Machine Learning research, yet almost unexplored in AutoML to this date: Natural Language Processing (NLP). It is also the first work in this area to work only with raw text data, assuming no previous transformation to the text files. As a result, a comprehensive study of the construction of pipelines for NLP was accomplished, providing practical metadata for the automation of such process; findings from this research also contributes to the characterization of text mining tasks by incorporating language features from the raw data. This study of NLP from an AutoML perspective also generates further research questions. What pre-processing techniques and classification algorithms have a greater impact in a pipeline? What optimization methods are more suitable for NLP? What other problems can be solved exploiting the metadata from different classification tasks?

In NLP some tasks can be modeled as a classification or a regression problem, one of the most prominent is the automatic categorization of text documents or text classification, because of the number of problems and applications that can be approached as a text classification task, such as author profiling [López-Monroy et al., 2015], sentiment analysis [Nakov et al., 2016] or spam filtering [Fusilier et al., 2015], methods for pre-processing, feature extraction, feature selection and document rep-

resentation have been widely studied and developed over the last decades. Each of these being appropriate for different scenarios and types of tasks. However despite the progress achieved by the NLP community, nowadays it is still an expert who determines the pipeline of text classification systems, including pre-processing methods, representation and classification models together with their hyper-parameters.

Some attempts to automate the design of text classification pipelines have been made [Lam and Lai, 2001, Yogatama et al., 2015, Gomez et al., 2017, Ferreira and Brazdil, 2018]. Nonetheless, these works are limited in at least one sense: restrictions in the search space of text representations or classification models (up to 48), experimentation on few classification tasks (up to 9), shallow characterization of text classification tasks. Furthermore, all prior works assume a previous transformation of the text documents to a vector-type representation instead of dealing with raw data. Such restrictions have allowed to explore some approaches to automating text mining but are too tight for more *realistic* scenarios.

Instead, the above restrictions have been considerably relaxed for this thesis: an extensive study was conducted in 81 text classification tasks of both binary and multi-class problems, a larger number of representations was added to the search space including different feature extraction algorithms, and a novel characterization was proposed for text classification tasks. Subsequently, a new method for autonomous text classification task is proposed.

The proposed method takes a step forward the automated selection of the full *pipeline* for text classification by exploiting metadata from prior knowledge and using state-of-the art optimization algorithms.

One of the principal novelties of this work is that unlike prior approaches it deals with raw text documents instead of a matrix representations which might make the problem of modeling a *good* pipeline harder, but is also a more realistic scenario for non-experts and more importantly it enables the extraction of rich textual infor-

mation. A key research question of this study is whether meta-data extracted from unprocessed text is able to characterize text classification tasks.

Meta-learning studies the employment of learning techniques to exploit meta-data from various tasks and *learning* how to solve them in a data-driven way instead of the more traditional manual search. In this work a novel task-characterization based on traditional and language-based meta-features is proposed for text mining problems; it is studied if such characterization outperforms the use of only traditional meta-features, and how this data can be exploited for the recommendation of full text classification pipelines.

Results in numerous datasets show empirically the effectiveness of both the proposed characterization for tasks and the proposed method for the recommendation of pipelines for text classification problems. Surprisingly, the recommended pipelines outperform state-of-the-art Deep Learning methods under some specific conditions.

This research compiles metadata from numerous tasks and algorithms, the meta-learning study is interesting for NLP and AutoML fields. The proposed method illustrates the capability of such information by automatically recommending pipelines with acceptable performance in almost any English text classification problem. This study might serve as base for further studies in automating other text mining tasks or extending the method to other languages.

Some future developments could consist on exploring other language characteristics to be incorporated in the set of meta-features, studying which type of meta-features work best according to the type of task, or extracting more metadata from more algorithms to enhance our text mining understanding and developing more robust AutoML solutions. As a suggestion, the proposed method can be expanded with state-of-the art representations, more hyper-parameters, and more models; as in other successful AutoML applications, an optimization method can be added to

the first stage of the method to be combined with the existing meta-learning setting. What is the best optimization algorithm for automating text mining tasks is an interesting question that deserves deeper analysis.

The compilation of text datasets is another contribution of this work, so far the existing benchmark for state-of-the-art classification models is biased to a couple of tasks and few datasets with hundreds of thousands of examples. Could a subset of the presented compilation be a more suitable benchmark for text classification pipelines? A scenario where traditional representations can outperform DL models is displayed in this work, other interesting questions arise from these observations. Is it possible to automatically determine for a text mining task when is it better to opt for traditional algorithms rather than DL given a budget restriction? If so. What conditions influence more this decision?

AutoML in NLP is almost uncharted, this work is one of the first attempts in the automation of time-consuming activities of the field, it is also the largest study to date in this area, a vast number of questions can be derived from this research. The following chapters contain more information about related works, details of the implementation, and the potential impact of this study.

1.2 Research problem

AutoML for text classification is studied in this research, specifically, the problem of automatically producing predictions for a new corpus is addressed. To make accurate predictions it is necessary to select text mining and classification algorithms, and to optimize their hyper-parameters. The selected algorithms must follow certain order to form a *pipeline* that receives a set of unprocessed text documents grouped by categories, selects an algorithm to transform them into a vector representation, selects an optimized classifier that will be trained with the set of transformed doc-

uments, and finally makes class predictions for new documents. This approach can be seen as an extension of the AutoML problem [Feurer et al., 2015a], formally:

For $i = 1, ..., n + m$ let $d_i \in D$ denote an unprocessed set of text documents and $c_i \in C$ the corresponding target value. Given a training dataset $D_{train} = \{(d_1, c_1), ..., (d_n, c_n)\}$ drawn from the same underlying data distribution, as well as a loss metric $\mathcal{L}(,)$, the AutoML problem for text classification is to automatically produce test set predictions $\hat{c}_{n+1}, ..., \hat{c}_{n+m}$. The loss of a solution $\hat{c}_{n+1}, ..., \hat{c}_{n+m}$ to the AutoML problem for text classification is given by $\frac{1}{m} \sum_{j=1}^{m} \mathcal{L}(\hat{c}_{n+j}, c_{n+j})$

To produce a set of predictions that minimizes the loss metric two types of algorithms are necessary, namely, classification algorithms $\mathcal{A} = \{A^{(1)}, ..., A^{(k)}\}$ and their associated hyper-parameters spaces $\Lambda^{(1)}, ..., \Lambda^{(k)}$, as well as, algorithms that transform the raw text to a numeric vector also known as text representation $\mathcal{R} = \{R^{(1)}, ..., R^{(l)}\}$ and their corresponding hyper-parameters $\Phi^{(1)}, ..., \Phi^{(l)}$.

An important component of the construction of text classification pipelines in comparison to other classification tasks is the selection of a text representation. This has a critical impact on the document categorization performance, NLP experts have a prior knowledge of what representation works best according to the problem domain, the documents characteristics and the specific classification task. Even though, small modifications such as changing from 2-grams to 3-grams could make a big difference in the overall model performance.

Two other aspects make this problem worth to be automated, there isn't a representation that works for every problem and it can be composed with: semantic, lexical, or syntactic features. In addition, there isn't a general classification model and testing different algorithms with small modifications to their hyper-parameters consumes a lot of time; in practical terms it is impossible for a human to test all the possibilities. There is also the problem that a human can be easily biased based on their previous experience. Finally, we can establish a general objective for this

thesis.

Objective: To develop an AutoML method that chooses a representation and a classifier to categorize text documents, performing similarly to state-of-the-art methods among different NLP supervised classification tasks.

Since this is a complicated goal to achieve an strategy was followed with the following specific objectives.

- To characterize text corpora using meta-features.
- To exploit the characterization and prior knowledge from different datasets to recommend a representation for text documents.
- To implement and evaluate a method that autonomously selects a classification model and optimizes its hyper-parameters.

As a result, a two-stage AutoML method is proposed (see Figure 1.1). When a new set of raw text documents is given, the first stage focuses on exploiting meta-data, obtained from the extensive experimentation on a large number of text classification tasks, to recommend an algorithm that will transform raw text into a vector representation. The second stage uses the transformed set of text documents to select a classification algorithm and optimize its hyper-parameters. Ultimately, a full classification pipeline is recommended without the involvement of human experts.

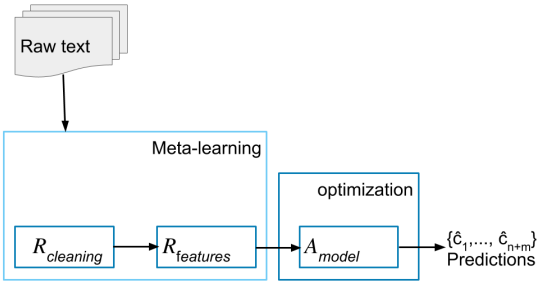

Figure 1.1: AutoML for text classification method.

1.3 Scope and limitations

An end-to-end method for automatically designing a model for classifying text documents is proposed. Despite AutoML aims to do this for any given machine learning task, this work has some limitations. First of all, the recommended *pipelines* are restricted to supervised text classification tasks, such tasks can be either multi-class or binary classification problems. The recommended representations are also limited to numeric vectors; graph-based representations are out of the scope of this work.

The proposed method is not guaranteed to work with languages other than English, other than that, no pre-processing is assumed for corpora, i.e. the proposed method expects as input the raw texts organized by their classes.

A characterization for text classification tasks is proposed, two problems were explored with this novel approach, classifying tasks by their type and recommending classification pipelines. However, related problems, such as determining a priori if a problem is *solvable*, with our method were not studied.

1.4 Thesis organization

This document is organized in 6 chapters. Chapter 2 presents the theoretical framework and concepts used throughout the thesis. Chapter 3 discusses the relevant works that are related to the research problem and to the proposed method. Chapter 4 explains the proposed method. Chapter 5 describes the corpora and the experimental setup to evaluate the method, likewise, it discusses the obtained results. Finally, Chapter 6 shows the conclusions of this research work and discusses its possible future directions.

Chapter 2

Theoretical framework

This chapter describes the fundamental concepts of this thesis. First, the task of text classification is defined and its implicated problems are briefly discussed, then the theoretical background for AutoML is presented.

2.1 Text classification

Written language is one of the most common forms of communication for the humankind, as a result, millions of text documents are produced every day. Text classification is the categorization of such documents into one or many categories, while this task has been done manually for centuries to organize or structure a document collection, in this work we use the term *text classification* to refer the automatic process of assigning a single label by *extracting* information from the raw text and *learning* a function from a set of examples (*corpus*).

Text classification has been used for a number of applications that include the automation of tasks commonly done by humans such as organizing literature by its genre [Kessler et al., 1997, Stamatatos et al., 2000] or news by its topic [Lewis et al.,

2004, Guo et al., 2006], it has also been useful in applications that leverage recent scientific advances, for instance, systems that detect depresion or aggressiveness in social media [Escalante et al., 2017,Schmidt and Wiegand, 2017], profiling an author by its genre, age, and nationality [Rangel et al., 2016, López-Monroy et al., 2015], or detecting the sentiments expressed in a text [Pang et al., 2008, Nakov et al., 2016].

Taking a *machine learning* approach, this problem can be modeled as a supervised classification task. Each example is a pair $(d_i, c_j) \in DxC$ where D is a set of text documents and $C = \{c_1, c_2, ..., c_n\}$ is a set of categories (also called classes). Specifically, the task is to find a function $h(d_i)$ that approximates the unknown function $f(d_i)$ that determines the correct class c_j given a document.

2.1.1 Pre-processing methods for text documents

The first step for classifying text documents is usually the pre-processing, which consists on at least one method to standardize the data and remove useless information. The most common of those methods are stop-word removal, stemming, and lowercase conversion, tokenization.

Stop-words are the words that appear the most within a document collection and that are independent from a particular topic or context, stop-words generally do not provide relevant information for discerning the class of one document to another and are commonly removed in the pre-processing stage [Uysal and Gunal, 2014].

Stemming is the procedure to obtain the root or stem from all the word derivations in the corpus, the stem can be as simple as the suffix of a word but due to the language complexity this is not always the case [Lovins, 1968], different algorithms have been developed for different languages, for instance, Porter algorithm is one of the most used for the English language [Porter, 1997].

Another common pre-procesing method, which is closely related to the fol-

lowing steps in text classification, is tokenization. It consists on the separation of elements from the text such as words, sentences, phrases, or other meaningful parts (tokens). After such segmentation it is easier to extract features from the text documents to build a representation.

2.1.2 Text representation

How to represent unstructured text documents is an active research field in NLP. State-of-the-art models are able to perform other tasks such as text generation or question answering [Devlin et al., 2018]. For problems in specific domains that are modeled as a text classification task usually a representation that includes task-specific information is build. This type of representations are not included in this study, alternatively, methods with a more general scope that focus on text classification tasks are described in this section.

Traditionally, the function $h(d_i)$ is known as *hypothesis, classifier* or *model* and it can't directly interpret the raw text, instead a compact form of representation is required, usually a vector obtained by a method that maps d_i to a set of features [Yan, 2009]. How to retrieve relevant information and how to find features to build such representations are fundamental questions in NLP.

One of the most common forms of representation for a text document is to use a vector of term *weights* $\vec{d_i} = \langle w_{1,i}, w_{2,i}, ..., w_{|T|,i} \rangle$ where T is the set of *terms* that appear at least once in the corpus. When the terms are words and the weights are binary, meaning whether it appears or not in a document, this representation is called *Bag of Words (BOW)*. Other weights often used are the term frequency $tf(t, d)$ and term frequency inverse document frequency $tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$ with $idf = \log \frac{|D|}{|d \in D : t \in d|}$.

There are two main drawbacks to this type of representation, the first one

is that the number of terms in some corpus can be large, thus, making it very hard for classifiers to deal with the large dimensionality. To address this, some simple techniques such as eliminating terms only appearing once or only using the n terms with greater frequency have shown to be effective, other methods for feature reduction or selection derived from machine learning can also be applied. The other disadvantage is that they fail to capture term correlations in the text.

A popular solution to the latter is the use of contiguous word or character sequences as terms instead of only one. This model is known as n-gram, bi-gram when $n = 2$, and tri-gram when $n = 3$. Usually, the value for n is not set greater than 4. Because of the exponential growth implied by including more terms, the problem of the dimensionality is accentuated. To deal with it, the same methods from BOW can be used. Another weakness of these models is that they aren't able to capture semantic information from the documents, therefore, synonymy and polysemy are concepts that are not considered and that can cause problems when solving a classification task. Techniques such as Latent Semantic Analysis [Deerwester et al., 1990] have been developed to include some semantic information.

LSA is a form of topic modeling, a representation can be build by learning latent topics performing a matrix decomposition on the document-term matrix using Singular value decomposition [Golub and Reinsch, 1971]. The idea behind it is to find co-occurrences of groups of words in different documents, each group of words is regarded as a hidden topics, by using this information it is possible to classify text documents. Another fundamental technique of topic modeling is latent Dirichlet allocation (LSA) that takes an statistical approach to find the hidden topics instead of using SVD. In particular, Dirichlet priors is employed for the document-topic and word-topic distributions, lending itself to better generalization [Blei et al., 2003].

An alternative approach for text representation is that based on linguistic information from the documents. What kind of features to extract depends on

the application context. A more generic tool for this is Linguistic Inquiry Word Count [Pennebaker et al., 2015] where a dictionary is used to tag the words from the document to a pre-defined set of categories that include reflected emotions, thinking styles, concerns, and other psycholinguistic aspects. LIWC has shown to be effective in classification tasks where reflected emotions are relevant [Rill-García et al., 2018].

Recent advances in text representations include word embeddings where terms are mapped to vectors of real numbers, there exist many methods that can generate such mapping, however, recent methods rely on the use of Neural Networks. A practical method of the above is Word2Vec, an algorithm that uses a two-layer Neural Network to learn a vector for words, it is also able to group similar words together since it uses the context where each word appears when learning the mapping from input text to the vectorspace. Similarly, for text classification tasks, documents can be mapped to a vectorspace [Le and Mikolov, 2014].

New representations are proposed every year, this work doesn't pretend to be an exhaustive study of all algorithms to date. A selection of the principal vector representations was made, these methods are usually the first approach to a text classification task but have shown effective results in a number of problems [Aggarwal and Zhai, 2012, Zhang et al., 2015, Iyyer et al., 2015]. All representations discussed in this thesis are limited to vector-type. Given the complexity of graph-based representations and since this work doesn't assume any prior transformation to the raw text, these are out of the scope of this work.

Other Deep Learning methods were also excluded from this research because their time requirements would have made it impossible to explore different approaches and because a lot of data is needed, which is not available for most of the corpora.

2.1.3 Classification models

After documents are pre-processed and a representations is build, a classifier needs to be selected. Such selection could have a negative or positive impact in the final category prediction. Given a dataset C, the goal of classification models is to *learn by example*. Essentially, a classification algorithm will adjust the parameters of the function $h(d_i)$ with feedback from each of samples in C.

Selecting a suitable classifier is not enough to guarantee a good performance for the model, an additional set of parameters needs to be set before the learning, also known as *training*, phase begins. Since small changes in the values of these *hyper-parameters* can have a high impact in the overall model, optimizing such values has become a relevant problem in machine learning research, this problem is further discussed later in this chapter.

In text classification, linear classifiers perform well and are usually preferred to non-linear models because the additional complexity of non-linear classification does not tend to pay for itself [Aggarwal and Zhai, 2012]. Popular classifiers in text mining include: Decision Trees, SVM Classifiers, Neural Network Classifiers, and Bayesian Classifiers [Aggarwal and Zhai, 2012], but in fact, almost any classifier can be adapted to text data [Aggarwal and Zhai, 2012].

A classifier not well studied in text classification that has gained a lot of attention recently is XGBoost. It is based on an ensemble technique called gradient boosting and has shown outstanding performance with tabular data [Chen and Guestrin, 2016]. Gradient boosting follows the idea that a set of weak hypothesis create a strong hypothesis, it trains several weak classifiers (in this case decision trees) by optimizing a loss function. XGBoost makes several algorithmic and system improvements to gradient boosting that allow the method to become scalable, faster, and more accurate [Chen and Guestrin, 2016].

2.1.4 Text classification pipeline

The success of correctly classifying text documents relies heavily on the correct selection of a text representation which is not an easy task. Empirically, it is known that the representation depends on the type of task that is being dealt with. For example, using characters instead of words for the n-gram model has shown to be effective for authorship attribution tasks. In addition, some classifiers work better with some representations and a combination of two or more representations can be used as the document vector.

A *pipeline* for text classification is the series of steps required to assign a label to a document since it is received as raw text. These include the pre-processing of the text (e.g. eliminating stop-words, removing terms, deleting special characters), selecting the features or terms to extract from the text, building the representation vector, determining if feature reduction is needed, if so which strategy to apply, selecting a classifier, and optimizing its hyper-parameters.

Extending the definition in [Zöller and Huber, 2019]. The algorithms to build a text classification pipeline can be grouped in 2 sets: those needed for building a text representation $\mathcal{R} = \{R_{cleaning}, R_{features}\}$ and those for classifying $A = \{A^{(1)}, ..., A^{(k)}\}$. Each algorithm $R^{(i)}$ is configured by a vector of hyper-parameters $\phi^{(i)}$ and each algorithm $A^{(j)}$ by a vector $\lambda^{(j)}$. A pipeline structure can be represented as a directed acyclic graph g where the nodes are the different algorithms and the edges the data flow through them. A text classification pipeline is then a set (g, R, ϕ, A, λ).

2.1.5 Pipeline performance

The performance of a given pipeline (g, R, ϕ, A, λ) can be evaluated in a corpus DxC with a loss metric $\mathcal{L}$. The performance is calculated as:

$$\pi = \frac{1}{n} \sum_{n}^{i=1} \mathcal{L}(\hat{c}, c),$$

where $\hat{c}$ is the predicted output of (g, R, ϕ, A, λ) after processing a sample $\vec{x_i}$ and c is the correct class for such text document d_i

For evaluating a text classification pipeline, datasets are divided in the same manner as in machine learning, a subset of the data is used by the model to learn D_{train}, a disjoint subset is used for the optimization of the hyper-parameters D_{valid}, and another disjoint set D_{test} is used by the model to make predictions and finally use the loss function $\mathcal{L}$ or analogously a measure for evaluation.

A popular evaluation measure is accuracy, which is defined as the ratio of correct predictions to the total number of input samples $\frac{TP+TN}{TP+TN+FP+FN}$. Accuracy results meaningless when classes are unbalanced (i.e. some classes have more samples than the other) which is common in many text classification tasks, for instance, when filtering spam, non-spam messages will outnumber the positive class; to overcome this problem it is common in these tasks to use the $f1$ score as a performance measure, this is the harmonic mean of two other measures: precision and recall, and is defined as $\frac{2TP}{2TP+TN+FP+FN}$.

Evaluation metrics are used to compare the performance of different pipelines and have a notion of the quality of them for solving a classification task, in the context of meta-learning this information is useful to acquire prior-knowledge. A related term are learning curves, these are the changes in performance of a classification model while optimizing its hyper-parameters. Learning curves are not only used to diagnose an overfit or underfit model but also to learn what hyper-parameters are more sensible to changes in its values, thus, they are also used in meta-learning to accelerate optimization methods.

2.2 AutoML

2.2.1 Hyper-parameter optimization

The performance of most machine learning algorithms heavily relies on the values of their hyper-parameters that are set before the training begins. Such values can be categorical like the kernel function of a Support Vector Machine, discrete as in the number of neighbors in Nearest Neighbors, or continuous like the error rate of a Neural Network. Hyperparameters have an effect not only on the performance of a model but also in its required time and memory. Hyper-parameters are usually *tuned* by a human-expert that combines knowledge with experimentation on a validation subset of the data D_{valid}. The problem of identifying a good value for hyper-parameters of a given classifier A is called the problem of *hyper-parameter optimization* [Bergstra and Bengio, 2012]. Formally defined as:

$$\lambda^{(*)} = \underset{\lambda \in \Lambda}{\operatorname{argmin}} \mathcal{L}(A_\lambda, D_{train}, D_{valid})$$

Where A is a machine learning algorithm and λ a vector of hyper-parameters in the hyper-parameter configuration space Λ. An algorithm with its hyper-parameters instantiated is denoted by A_λ.

The HPO problem has been extended for AutoML [Escalante et al., 2009,Feurer and Hutter, 2018], a widely adopted approach is Combined Algorithm Selection and Hyper-parameter optimization (CASH) [Thornton et al., 2013], where instead of selecting a set of values for hyper-parameters the task is to automatically design a machine learning pipeline, thus for the selection of pre-processing methods and machine learning algorithms these are treated as categorical hyper-parameters.

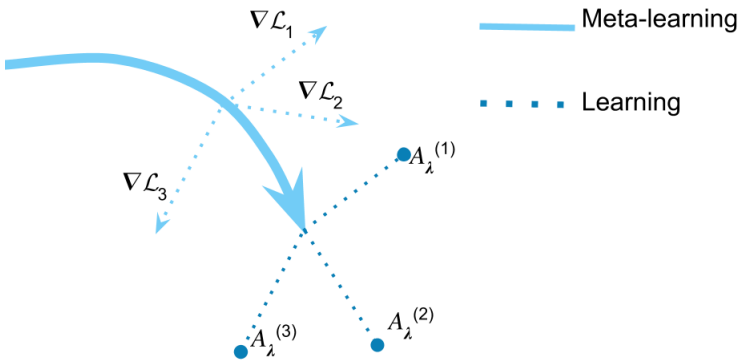

Figure 2.1: Meta-learning leverages learning metadata to *learn*

2.2.2 Meta-learning

For humans, experiences from the past are usually helpful when learning a new skill or for solving a new problem. Equivalently, in the context of machine learning, meta-learning takes advantage of prior experience or metadata, acquired when solving a wide range of tasks, to learn new tasks, thus, learning in a data-driven way instead of designing algorithms in a hand-engineered fashion [Vanschoren, 2018]. Figure 2.1 illustrates the process of learning from several tasks (meta-learning) and then adapting to new tasks (learning) with few data points [Finn et al., 2017]. The main goals are to speed up the learning process and to improve the quantitative performance of models. Meta-learning has had an impact into several machine learning problems such as learning to design optimization algorithms [Andrychowicz et al., 2016], automatically suggesting supervised learning pipelines [Feurer et al., 2015a], learning architectures for deep neural networks [Elsken et al., 2018], and few-shot learning [Ravi and Larochelle, 2016].

One of the fundamental challenges in meta-learning is how to define a task similarity, the larger the number of *similar* tasks available the more metadata can be extracted, when a new completely unrelated task is presented, prior experience will not be effective. As a result, the success of meta-learning relies on the availability of data from such tasks, fortunately, nowadays the production and access to huge

amounts of data from the real-world is possible.

2.2.3 Meta-features

A rich source of metadata are characterizations (meta-features) of the task at hand. Each task $t \in T$ is described with a vector $m(t_j) = (m_{j,1}, ..., m_{j,K})$ of K meta-features. This set is useful to define a task similarity measure based on, for instance, the Euclidean distance between $m(t_i)$ and $m(t_j)$ so that we can transfer information from the most similar tasks to the new task t_{new}. Moreover, together with prior evaluations, we can train a *meta-learner* to predict the performance of configurations in new tasks [Vanschoren, 2018].

Chapter 3

Related work

AutoML is closely related to two fields in machine learning, both of which have been widely studied for several years. The first of them is hyper-parameter optimization because AutoML is often seen as a generalization of the hyper-parameter optimization problem [Thornton et al., 2013] and because some of the methods with best empirical results often work well in both fields. The other is meta-learning or *learning to learn* [Vanschoren, 2018]. Optimization algorithms can be *warm-started* with the best pipelines from *similar tasks* and in general meta-learning helps to understand when and where to use a model or which hyper-parameter configuration to select.

3.1 Hyper-parameter optimization

Given the importance of the problem in empirical machine learning work, hyper-parameter optimization (HPO) has been studied for decades [Bozdogan, 1987,Kohavi and John, 1995,Bengio, 2000,Bergstra et al., 2011] . In Bergstra and Bengio, 2012, it was shown that for a machine learning algorithm, random search is one of the most effective methods for choosing values for hyper-parameters. Random search consists

in selecting randomly different combinations of values from the search space. In Bergstra and Bengio's experiments this method showed to achieve better results than manually selected values and exhaustive Grid Search given a restricted budget of time, later works have proposed different approaches but Random Search is still a common baseline for comparison, recent approaches for the HPO problem can be divided into three type of methods: *population-based, multi-fidelity,* and *model-based* [Feurer and Hutter, 2018].

Due to the non-convex nature of the problem, global optimization algorithms are preferred [Feurer and Hutter, 2018], some of the most common are population-based methods where the population is a combination of hyper-parameters values, and these include *genetic algorithms, particle swarm optimization,* and *evolutionary strategies* [Lessmann et al., 2005,Lorenzo et al., 2017,Hansen, 2016]. The population in these methods is improved iteratively by *mutation* and *crossover* (i.e. modifying some of its individuals and combining them between each other).

When manually selecting the hyper-parameters for a given learning algorithm a common practice is to test different configurations in a small subset of the data, analogously, multi-fidelity methods approximate the loss function of an optimization algorithm with *low fidelity* making a trade-off between the learning algorithm performance and its required computational resources. Some of these methods are based on learning curves and deciding to early stop the training of the least promising configurations [Domhan et al., 2015]. Learning curves can be obtained from the performance of a set of hyper-parameter values on different subsets of data or from the performance of a iterative optimization algorithm, for example.

Other multi-fidelity methods are strategies based on the multi-armed bandit problem [Jamieson and Talwalkar, 2016] that try to approximate the performance of different configurations to select the best. One of such strategies is *sequential* or *successive halving* algorithm [Karnin et al., 2013] where given a fixed time budget T,

it is evenly divided between $log_2 n$ rounds, in each round all arms are *pulled*, which in HPO this means to empirically test all configurations left, at the end of the round the worst half of arms (configurations) is eliminated. A problem for this strategy is the trade off between choosing to test few configurations in a large budget or a lot in small time. *HyperBand* [Li and Jamieson, 2018] strategy addresses this problem by dividing the budget between combinations of configurations with different budget sizes and applying *sequential halving* in each division.

Other common global optimization method is Bayesian Optimization (BO) which has been applied to different optimization problems related to machine learning, robotics, user interfaces, information extraction, and combinatorial optimization, many variants from the original algorithms have been developed over the years [Shahriari et al., 2016]. Recently, BO has also shown to be an effective method for HPO [Snoek et al., 2012, Klein et al., 2017].

BO is a sequential model-based approach that consists of two components: a Bayesian statistical model for modeling the objective function (*surrogate model*) and an *acquisition function* for deciding where to sample next [Frazier, 2018], thus, the model starts with a prior probability which is updated after each iteration with all the observations of the target function so far. After the surrogate model is fitted the acquisition function selects promising candidates by trading off exploration and exploitation, this function does not only have to predict correctly but it also has to be *cheaper* to evaluate than the target function, a common choice is the *excpected improvement* (EI) $E[I(\lambda) = E[max(f_{min} - Y, 0)]$. One important disadvantage of this method is its poor scalability when dealing with large search spaces, to address this in HPO some improvements have been proposed, one of them is the use of meta-learning [Feurer et al., 2015b] discused in the next section.

Scalability of the three approaches discussed in this section is an open question in HPO, research in this direction has lead to the development of AutoML systems

that scale one of these three type of methods by enlarging the search space to include not only hyper-parameter values but also pre-processing and learning algorithms. Integrating meta-learning techniques with these methods is also a common practice that, given the larger search space, seeks to speed up the optimization process and to increase the performance of the resulting model. Furthermore, these three approaches are not necessarily incompatible with each other, in fact, proposals to combine BO with both population-based [Golovin et al., 2017] and multi-fidelity [Falkner et al., 2018] methods are the state-of-the-art in HPO.

3.2 Meta-learning and automated machine learning

Following the three HPO methods previously presented, this section discusses recent approaches to the AutoML problem and how meta-learning is applied in each.

Population-based approaches include some of the earliest AutoML works, in Escalante et al., 2009, *particle swarm optimization* is applied not only to the HPO problem but for a *full model selection* or the optimization of a full machine learning pipeline including: pre-processing, feature selection, classifier and its hyper-parameters. This method is known as particle swarm model selection (PSMS) and was later extended to also use a genetic algorithm for the optimization [Sun et al., 2013]. Meta-learning has been used with population-based methods to improve their performance for example to *guide* the mutation or crossover operations [Schmidhuber, 1987] or to *seed* the initial population [Kordík et al., 2018], however, to date there isn't a direct application of meta-learning in population-based methods for general AutoML.

At the time of writing, Bandit-based methods and Bayesian Optimization are the most popular approaches to AutoML. One of the Bandit-based methods is *Active testing* [Abdulrahman et al., 2018], a meta-learning strategy that ranks the

pipelines predicting a multi-objective measure (A3R) that makes a trade-off between accuracy and runtime, it then iteratively tests the top N ranked configurations to find the best. For Deep Learning, Hyperband has shown to outperform random search and BO methods [Li and Jamieson, 2018]. Auto-Band [das Dôres et al., 2018] is an extension to Hyperband for AtuoML that incorporates meta-learning. With experiments on 322 datasets they build a knowledge-base and a meta-regressor to predict performance of different pipelines, given a new dataset Auto-band assigns a greater probability to the promising pipelines than those that are predicted to perform poorly, with this additions to Hyperband they were able to outperform other successful AutoML methods: *AutoWeka* and *Autosklearn*.

Bayesian Optimization has also proven to be a practical and efficient approach to AutoML, AutoWeka [Thornton et al., 2013] firstly defined AutoML as the *Combined Algorithm Selection and Hyperparameter Optimization* (CASH) problem. They used a Random Forest-based BO (SMAC) [Hutter et al., 2011] for optimization in their expanded search space. Later, Autosklearn [Feurer et al., 2015a] extended the AutoML problem and introduced two improvements to the SMAC optimization, the first is to select as a final model an ensemble of the best classifiers instead of using only one for the classification and the other was the addition of meta-learning to *warm-start* SMAC.

Both Auto-Band and Autosklearn rely on the use of meta-features for describing classification tasks. In this way, each task is usually represented by a vector where dimensions are associated to meta-features which can be as simple as the number of instances and features in a dataset and as complex as statistical measures from the data distribution. For finding *similar* tasks the Euclidean distance or other similarity measure can be used between vectors. Several measures have been proposed to work as meta-features and different sets have been tested and compared [Rivolli et al., 2018] however some studies suggest that the optimal set depends on the application [Vanschoren, 2018].

3.3 Automated text classification

In the context of text mining, meta-features from clustering text documents have been used directly for classification [Canuto et al., 2018]. In the context of meta-learning these features have been used only in very specific domains [Lee et al., 2007]. Nevertheless, few works have explored the automated selection of different parts of classification pipelines. With experiments in the Reuters-21578 corpus, [Lam and Lai, 2001] proposed to characterize documents with 9 (meta) features and to predict the classification error of different models using data from a previous phase, thus recommending a classification model. Despite their limited scope, given the lack of data and computational resources of the time, this work represents one of the first meta-learning approaches for text classification.

More recently, and in a more AutoML fashion, [Yogatama et al., 2015] searched for text representations with a state-of-the-art method: Bayesian Optimization [Snoek et al., 2012]. Their search space was limited to only word n-grams and experiments were performed in 8 datasets: 4 sentiment analysis tasks and 4 topic classification tasks. Nevertheless, they outperformed every linear classifier reported until their publication date.

Other works have explored different meta-learning approaches for text classification in small-scale, for example, [Gomez et al., 2017] addressed the problem with evolutionary computation methods. Specifically, a genetic algorithm was developed to learn rules to recommend a classification model and some hyper-parameters given the values of a set of 11 meta-features, thus finding relations between characterized corpora and the *best* fitted model. With a broader scope, [Ferreira and Brazdil, 2018] recommend *pipelines* with Active testing, in their work they also present statistical analysis of 48 pre-processing methods and 8 classifiers.

Another related work is that by [Pinto, 2008] where a set of features derived

Work	M-features	Tasks	Reps	Clsfs	Raw	Method
Lam, et al.	9	1	1	6	no	*Low-fidelity*
Gómez, et al.	11	9	1	42	no	Genetic algorithm
Yogamata, et al.	0	5	36	1	no	Bayesian optimization
Ferreira, et al.	0	3	48	8	no	Active Testing
Madrid, et al.	72	81	60	110	yes	AutoText

Table 3.1: Current approaches to automated text classification.

from the text was proposed for characterizing short-text corpora in the context of clustering. Although the goal of such reference was to characterize the hardness of text corpora and not AutoML, such work is relevant because their metrics inspired some of the meta-features considered in this thesis. In Chapter 3, it is described how these set of features are combined with other proposed ones for characterizing text collections in the context of automatic text mining.

In this thesis a novel approach to meta-learning of text representations is proposed. A set of meta-features is defined, comprising standard meta-features from the machine learning literature, features that have been used for other problems than meta-learning and novel meta-features that have not been used previously. Some of these meta-features are derived directly from raw text and aim at capturing complex language patterns. We approach the problem of recommending textual representations as well as classification algorithms. Whereas this problem has been approached in previous work, such references have always pre-processed text files to a matrix-document representation as input to their systems and have considered only a few representations and a very limited number of meta-features (up to 11). To the time of writing this is the largest scale study on meta-learning in the context of text mining.

Chapter 4

An AutoML method for text classification tasks

A solution to the research problem of this thesis is presented and detailed in this chapter. A novel meta-learning approach for recommending textual representations is proposed as a first step of the method, then, auto-sklearn is adopted for the classifier optimization. Figure 4.1 depicts both steps in the proposed method, initially raw data is taken in the meta-learning phase where pre-processing and feature extraction algorithms are recommended ($R_{cleaning}$,$R_{features}$), the text is transformed with those algorithms and the new representation is fed to auto-sklearn which recommends a classifier and values for its hyper-parameters (A_{model}). As a result the method recommends a text classification pipeline: an algorithm to transform the raw text into a vector representation and a classification model.

Compared to AutoML advances in other areas, previous approaches to the problem have failed to produce the same impact and results for NLP., this is mainly because of the limited scope of previous works which have only experimented in a small number of tasks exploring a few text representations, all consisting of different hyper-parameter values for the n-gram model. In a meta-learning context no other

work have extracted metadata directly from the text itself to leverage it when dealing with unseen tasks.

In comparison, the method described in this chapter has a broader approach, a wider range of representations has been added to the search space, a characterization that extracts rich meta-features from the text is presented for the first time and 4 strategies to exploit metadata and recommend textual representations are proposed. Additionally, a new set of datasets gathered from different sites have been comprised in a repository and made publicly available. The proposed method is the most ambitious attempt to date for the automation of text classification tasks.

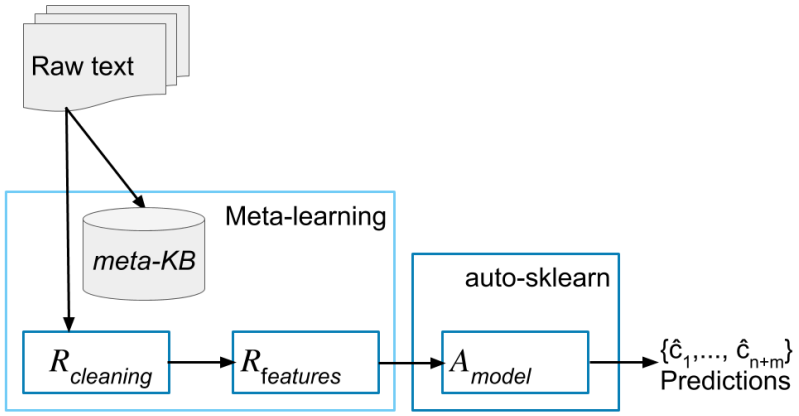

Figure 4.1: AutoText. A text representation is recommended via meta-learning, then, auto-sklearn selects a classifier.

4.1 Meta-learning of textual representations

A human-expert uses knowledge acquired in the past when a new task is presented, equivalently, *meta-learning* imitates this process. As the first step to the automated selection of text classification pipelines, a meta-learning method is proposed. It takes as input the labeled raw text from a corpus associated to a text classification task and automatically selects a representation. As an output the method recommends

vector representations for text classification tasks based on which one worked best for *similar* tasks. Although this approach is common within meta-learning [Vanschoren, 2018], it has not been widely explored for text classification (see Chapter 3).

In fact, this work is the first attempt to deal with raw data; despite some formatting and encoding issues that had to be addressed, this novelty is fundamental to the proposed method. It is a more likely setting for non-expert but the method also explores for the first time how to extract rich information from the language to incorporate it for the characterization of NLP tasks.

One of the key ideas of meta-learning is to describe a task and then find similar tasks from a large pool. The more similar two or more tasks are, the more likely it is that using the same model to solve them will perform similarly. Consequently, this meta-learning approach relies on 3 essential factors:

- Describing a task *clearly*
- Determining the similarity between tasks
- Extensive availability of different tasks

The proposed method acknowledges these three points. Firstly, a novel text classification task characterization is described, that unlike previous work it recognizes the difference between a text corpus represented in a tabular form and the corpus raw text itself. Secondly, different strategies are compared to find which leverages better prior knowledge finding the most similar tasks. Lastly, a wide variety of tasks is studied, 81 text corpora were gathered among different public websites. A complete list of them can be found in Chapter 5 as well as a brief description.

Several vector-type representations have been selected for our method to choose from. Table 4.1 sums up the feature extraction methods that with some preprocessing techniques or hyper-parameters give a total of 60 representations. While

not exhaustive, this work is the first to consider representations not only based on simple features, but also those based on topic modeling, embeddings, and semantic analysis.

Features	Hyper-parameters
N-grams	[words, char], stop_words[None, 'English'], range[1,3], weight[bi,tf,tfidf]
LDA	stop_words[None,'English']
LSA	stop_words[None,'English'], weight[tf,tfidf]
LIWC	categories[60]
W2V	pre_trained[True,False], vector[mean, sum]

Table 4.1: Representations considered

The selected representations can be regarded as the main approaches with a broad scope for the categorization of text documents, a description of them can be found in Chapter 2. Recent efforts for text representations are centered on Deep Learning methods where usually a recurrent network is trained to learn a language model. Results of such methods have outperformed the *traditional* text representation forms in a number of tasks including text classification problems. Despite their success there are critical limitations of these methods.

The most relevant of these limitations being the need for large amounts of data (benchmark datasets for DL have at least 100,000 documents and up to millions). Although big datasets that contain millions of samples for the English language are already available, that is not the case for most of the text classification tasks. Besides, language components vary noticeably from one context to another, this has been accentuated with the recent emergence of social media and other specialized websites. For instance, text samples from corporate emails will be completely different from those found on Facebook, where even the spelling of words will change and additional characters like *emojis* will appear.

To address the lack of samples in every scenario a common practice is to *transfer* learning between tasks by training a model on the larger dataset and fine-tuning in the smaller and specialized corpus. In these situations, the proposed method in this research explores whether it is possible to outperform DL representations with the traditional ones being selected in a data driven way by exploiting information obtained from a large number of both big and small corpora for text classification tasks.

Due to the quantity of data required and the number of operations, DL also consumes a lot of computational time. Since experimentation for this method involves a variety of tasks and metadata is collected for as many configurations as possible, introducing DL representations to the pool wouldn't adjust to the scope of this study.

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

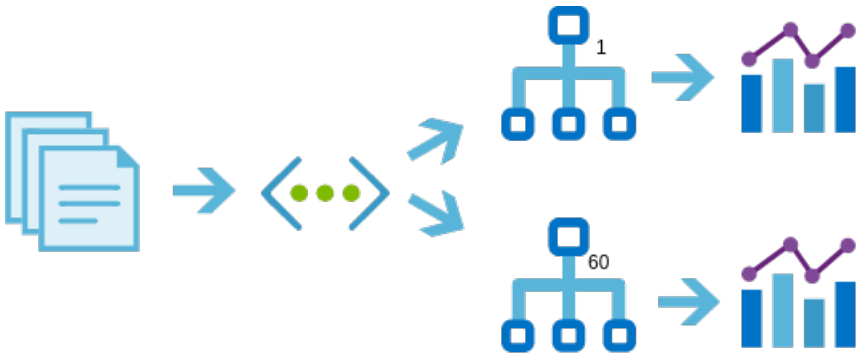

Figure 4.2: Offline phase. All representations are evaluated.

The first step of the method is an offline phase that is executed once; the purpose of it is to extract metadata for each of the 60 representations and all of the tasks, intuitively, we want to know which is the best representations for each of the tasks. Then the extracted knowledge will be used for recommending the most

suitable representation according to previous experimentation.

So, the 60 configurations are evaluated. A common practice in AutoML is to approximate the performance of a configuration using surrogate models or *heuristics* that reduce the search time [Vanschoren, 2018, Feurer and Hutter, 2018, Quanming et al., 2018]. Because obtaining an exact evaluation for all the combinations of representations, classifiers, and hyper-parameter values is prohibitively expensive to be invoked repeatedly, the performance on an linear SVM classifier is proposed to estimate the quality of each representation. The hyper-parameters of this classification model are detailed in Table 4.2. The linear SVM is a popular classification model in text classification problems given its robustness to high dimensionality and its general effectiveness in these tasks [Allahyari et al., 2017, Aggarwal and Zhai, 2012].

It is also common to make the model evaluations with either holdout or cross-validation techniques depending on the size of the datasets [Vanschoren, 2018, Ali and Smith, 2006]. For evaluating the text representations, a 3-fold cross validation was selected considering 3 to be a fair number of splits for evaluating small datasets and not too time-consuming for large datasets.

As the evaluation measures for this phase two metrics where selected: *accuracy* because of its wide adoption in the community in a number of tasks and *f1 score* for the tasks where the positive class is more relevant or when there is presence imbalance in a dataset. Although metadata for both measures was obtained, only one can be selected at a time as a loss function to optimize, which to select will depend for instance on what type of task is being attacked, as default, *accuracy* showed *good* performance for diverse datasets.

This process is repeated for all of the 81 text classification tasks, each of them described by a vector of novel meta-features extracted directly from the documents of the corpus associated to each of the tasks. As a result a *meta-knowledge-base* that contains information about the performance of each of the representations for

Hyper-parameter	Value
Kernel	linear
Loss function	Squared Hinge loss
Stopping criteria	0.0001
C	1.0
Max iterations	1000
Multiclass strategy	one vs all

Table 4.2: Hyper-parameters for the Support Vector Machine used to evaluate representations in the offline phase.

each of the tasks is generated (see Figure 4.3. The predicting phase leverages this metadata to recommend a text representation for a classification problem. Even though, the offline phase requires a lot of processing time, it is only executed once. Moreover, metadata for new tasks or for new representations can be added with ease to the *knowledge base*, the rest of the steps of this method will then make use of the new data without further modifications.

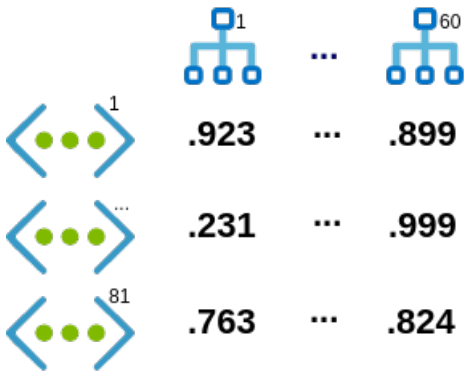

Figure 4.3: Offline phase. A meta-knowledge base is built experimenting on all tasks. This data is later used to train *meta-models* that recommend representations.

Previous works related to text classification with meta-learning have always assumed prior pre-processing for the text documents, the input for our method does

not require any pre-processing as it takes the raw text and its output can be useful for both human experts designing text classification pipelines and for complementing other optimization methods for AutoML (e.g. it can be useful for warm-starting Bayesian Optimization [Feurer et al., 2015b] for a wider search space of text representations or easily combined with existing AutoML solutions).

4.1.1 Proposed meta-features

Our method applies meta-learning to learn from the performance of different representations on the 81 corpora which are stored and described in a *knowledge base* with a meta-features vector. This is a common approach to characterizing classification tasks (see Chapter 2) however none of the prior work have acknowledged the complexity and features of written natural language.

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

Meta-features are a common form of characterizing tasks, these attempt to capture relevant information from datasets related to their distribution form or the statistics of their features [Vanschoren, 2018]. Some sets of meta-features have proved to be useful for supervised machine learning problems, however these are not descriptive enough to characterize tasks in text classification.

Traditionally, meta-features extract metadata from a dataset to capture rel-

evant information and use it to characterize classifications tasks, commonly used measures include statistics of the distribution of the classes or more simple characteristics like the number of samples and attributes. Such meta-features have proved to be useful for supervised machine learning problems and for AutoML. However, extracting them usually requires a tabular representation of the data, in the case of text documents some representation such as Bag-of-Words would be necessary.

When a representation is selected some fundamental characteristics of language are lost, extracting *traditional* meta-features from raw text would result in a limited characterization of the task. In the proposed set this type of features are contemplated as well as other attributes extracted directly from the raw text. A total of 72 meta-features are proposed combining traditional meta-learning features with NLP ones. Below they are organized in groups and a description is provided. Table 4.3 defines the symbols of some corpus data extracted from text that is used to calculate a number of the proposed measures.

For the non-traditional meta-features metrics that somehow measure the following characteristics of the written language are proposed: vocabulary, complexity of sentences, lexical features, writing style, readability. Despite not being exhaustive the new characterization should capture differences between different usage contexts.

- **General meta-features.** The *number of documents* and the *number of categories*.

- **Corpus hardness.** These features aim to capture information on the *hardness* of text corpora, they were originally used in [Pinto, 2008]

  Domain broadness. Measures that capture the broadness degree of the corpus, a narrow broadness includes terms closely related to each other while a wide one has more diversity on its terms. For instance, we would expect a corpus of several news categories such as 20Newsgroups to have a wider broadness than a

Symbol	Description
n	Number of documents
k	Number of classes
$D = \{d_1, d_2, ..., d_n\}$	Corpus
$C = \{C_1, C_2, ...C_k\}$	Classes
$\|D\|$	Number of words in the corpus
$\|d_i\|$	Words in document i
$\|C_i\|$	Words in class i
$V(...)$	Vocabulary of...
$\varphi(C_i, C_j)$	Similarity
$ENDC(D)$	Expected Number of DoCuments
$tf(t_i, D)$	Frequency of term t_i
$DL(D)$	Average Document Length

Table 4.3: Nomenclature of metadata from corpus.

corpus of hotel reviews. Measures included are based on the vocabulary length and overlap: *Supervised Vocabulary Based (SVB)*:

$$SVB(D) = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(\frac{|V(C_i)| - |V(D)|}{|D|}\right)^2}$$

Unsupervised Vocabulary Based (UVB):

$$UVB(D) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{|V(d_i)| - |V(D)|}{|D|}\right)^2}$$

Macro-averaged Relative Hardness (MRH):

$$MRH(D) = \frac{1}{k(k-1)/2}\sum_{i,j=1;i<j}^{k}\varphi(C_i, C_j)$$

$$\varphi\left(C_i, C_j\right) = \frac{|C_i^* \cap C_j|}{|C_i \cup C_j|}$$

Class imbalance, Refers to the document distribution across the classes. A simple *Class Imbalance (CI)* ratio was implemented:

$$CI(D) = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(|C_i| ENDC(D)\right)^2}$$

$$ENDC(D) = \frac{|D|}{k}$$

Stylometry. The writing style of a text usually attributed to a specific author. *Stylometric Evaluation Measure (SEM)*, this is obtained by calculating the asymmetrical Kullback-Leibler distance of the term frequency distribution of D $P(t_i)$ with respect to its Zipfian distribution $Q(t_i)$

$$P(t_i) = \frac{tf(t_i, D)}{\sum_{t_i \in V(D)} tf(t_i, D)}$$

$$Q(t_i) = \frac{1/i^*}{\sum_{r=1}^{V(D)} 1/r^s}$$

$$SEM(d) = \sum_{t_i \in V(D)} P(t_i) \log \frac{P(t_i)}{Q(t_i)}$$

Shortness. Features based on the length of the text or the vocabulary used in the corpus. *Vocabulary Length (VL)*:

$$VL(D) = \frac{1}{n}\sum_{i=1}^{n} |V(d_i)|$$

Vocabulary Document Ratio (VDR):

$$DL(D) = \frac{1}{n}\sum_{i=1}^{n} |d_i|$$

$$VDR(D) = \frac{\log(VL(D))}{\log(DL(D))}$$

and average *word length*.

- **Statistical and information theoretic.** Meta-features that are derived from a document-term matrix representation of the corpus. The corpus documents

are described with a document-term matrix and the following meta-features are calculated:

min, max, average, standard deviation, skewness, kurtosis, ratio average-standard deviation, and entropy of: vocabulary distribution, documents-per-category and words-per-document:

Landmarking. 70% of the documents are used to train 4 simple classifiers and their performance on the remaining 30% was used based on the intuition that some aspects of the dataset can be inferred: *data sparsity - 1NN, data separability - Decision Tree, linear separability - Linear Discriminant Analysis, feature independence Naïve Bayes.* The *percentage of zeros* in the matrix was also added as a measure for sparsity.

Principal Components (PC) statistics. Another frequent group of meta-features come from Statistics derived from a PC analysis: from the Principal Component Analysis of the data, in our representation the components are the most relevant terms for a corpus, we included the following measures: *pcac* from [Gomez et al., 2017]; for the first 100 components, the same statistics from documents per category and their *singular values sum, explained ratio and explained variance*, and for the first component its *explained variance.*

- **Lexical features.** The distribution of parts of speech tags was included. The frequency of some lexical items will be higher depending on the task associated to a corpus, for instance a corpus for sentiment analysis may have more adjectives while a news corpus may have less. The words were tagged in the document and computed the average number of *adjectives, adpositions, adverbs, conjunctions, articles, nouns, numerals, particles, pronouns, verbs, punctuation marks* and *untagged words* in the corpus.

- **Corpus readability.** Statistics from text that determine readability, com-

plexity and grade from textstat library[1]: *Flesch reading ease:*

$$206.835 - 1.015\left(\frac{|D|}{|Sentences(D)|}\right) - 84.6\left(\frac{|Syllables(D)|}{|D|}\right)$$

SMOG grade:

$$1.043\sqrt{|Polysyllables(D)| \times \frac{30}{|Sentences(D)|}} + 3.1291$$

Flesch-Kincaid grade level:

$$0.39\left(\frac{|D|}{|Sentences(D)|}\right) + 11.8\left(\frac{|Syllables(D)|}{|D|}\right) - 15.59$$

Coleman-Liau index:

$$0.0588L - 0.296S - 15.8$$

where L is the average number of letters per 100 words and S the average number of sentences per 100 words, *automated readability index:*

$$4.71\left(\frac{|Chars(D)|}{|D|}\right) + 0.5\left(\frac{|D|}{|Sentences(D)|}\right) - 21.43$$

Dale-Chall readability score:

$$0.1579\left(\frac{|difficult\_words(D)|}{|D|}\right) + 0.0496\left(\frac{|D|}{|Sentences(D)|}\right)$$

the number of difficult words, Linsear Write formula:

$$\frac{(3|complex\_words(D)|) + (|D| - |complex\_words(D)|)}{2|Sentences(D)|}$$

where complex words are those with more than 3 syllables*Gunning fog scale:*

$$0.4\left(\frac{|D|}{|Sentences(D)|}\right) + 40\left(\frac{|complex\_words(D)|}{|D|}\right)$$

and the *estimated school level to understand the text* that considers all the above tests.

Apart from general, statistical and PC based, the rest of the listed features have not been used in a meta-learning context. After the offline phase takes place, for a new task the same meta-features are extracted and compared with the prior knowledge, to recommend a representation.

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

Group	Meta-features
General	number of documents, number of categories
Hardness	SVB, UVB, MRH, CI, SEM, VL, VDR
Statistical	word per document statistics, document per category statistics, landmarking metrics, PC statistics
Lexical features	Parts of Speech distribution
Readability	SMOG, Flesch-Kincaid, Coleman-Liau, Dale-Chall, difficult words, Linsear Write, Gunning, school level

Table 4.4: Summary of proposed meta-features.

4.1.2 Recommendation of textual representations

For the predicting phase, 4 strategies were considered that leverage learned experiences and make predictions in a new task. Which (meta)model to adopt could be approached as another meta-learning problem itself, the possibility and effectiveness of several meta-learning levels is a known but unexplored problem in the field.

The first strategy follows the most common approach in meta-learning, determining the closest task with a *similarity* measure. For strategies 2-4 different models were tested with the evaluation described in Section 5.3. Ranking configurations by their expected performance is common within meta-learning [Vanschoren, 2018] methods, this is done by training regression models. XGB and RF have successfully accomplish this ranking task [Vanschoren, 2018,Brazdil and Giraud-Carrier, 2018] and thus were selected for the models to test, analogously, the classification version of those models was also chosen. Because of their success in many machine learning tasks, SVM classifiers and regressors with different kernels were also tested. For comparison, some linear regression methods were also tested obtaining poor results. Finally, the best performing models were selected: Random Forest classifier for strategy (2) and Random Forest regressor for both strategies (3) and (4). All

strategies are described below.

- **(1)** or **1NN** Using directly the representation with best performance of the *nearest* corpus. This strategy directly follows the idea of finding the most similar task in order to know what model will work best. The Euclidean distance is used to determine the *similarity* between the new task and those in the knowledge base. This approach can also be seen as classifying unseen tasks with a Nearest Neighbors algorithm using only 1 neighbor, in which case each of the 60 representations constitutes a class.

- **(2)** Predicting the representation as a classification problem, where each representation is a class and every prior task is a sample represented by its 72 meta-features. $m(t_i) = (m_{i,1}, ..., m_{i,72})$ In this case every sample was labeled with the representation with best performance as its class $(m(t_i), R_\phi^{(j)}) \in Mx\mathcal{R}$, where M is the set of meta-features and $\mathcal{R}$ the set of representations. Thus, the problem is to find a model that automatically select the *correct* class $h(m(t_i)) = R_\phi^{(j)}$, finding complex patterns among the tasks and using 81 samples for training. Since the dimensionality of this problem is big given the number of samples available this isn't an easy task, so different classification models were tested. In the end, a Random Forest classifier was selected as the model for this strategy. Hyper-parameters of this RF model are listed below in Table 4.5. Strategy (1) can be considered a particular case of (2), however, a distinction is made because the former is widely popular but the latter isn't.

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

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

Table 4.5: Hyper-parameter for the Random Forest Classifier used in strategy (2).

finally, a Random Forest Regressor was found to work best.

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

Table 4.6: Hyper-parameter for the Random Forest Regressor used in strategies (3) and (4).

- **(4)** Predicting the rank of each representation and selecting the one with the best predicted rank. 60 regression models are trained with performances in 81 different tasks. Given a new task the 60 trained models predict the expected rank for each representation, the results are ordered and the representation with lowest rank is recommended. Like before various regression models were tested and again a regression with Random Forest was selected (see Table 4.6).

Strategies (1) and (3) are the most common approaches in meta-learning [Vanschoren, 2018]. Both follow an intuitive idea, either finding the most similar task

or learning to predict the performance of a pipeline. Strategies (2) and (4) respectively expand those ideas and address some of the particular disadvantages of (1) and (3). It is also important to notice that any of the strategies can be expanded from recommending a single representation to the best n.

(1) is more sensitive to *outliers*, similar tasks where the representation is completely different or tasks that are not related to the others, (2) is more robust to such situations if considered as a traditional classification problem. (4) simplifies the regression problem from (3), performances can vary little between representations making it hard to train a model for any task, the ranking can potentially be easier to predict accurately.

Between regression strategies (3 and 4) and classification strategies (1 and 2) there is also an important difference in computation time. A different prediction for either ranking or performance is needed for each of the representations, thus, 60 different models have to be trained, in comparison, only one is necessary for the classification strategies. The difference is accentuated if the search space is expanded. The proposed method includes both approaches since the decision of which to utilize will depend on the resources available for a certain user.

4.2 Full pipeline selection

The meta-learning approach comprises the first step in the proposed method for recommending text classification pipelines. It leverages prior knowledge from extensive experimentation to select automatically pre-processing methods to form a representation for the raw text documents. However, the problem of selecting a classification model and optimizing its hyper-parameters remains.

To address it state-of-the-art AutoSklearn is adopted. This receives as input tabular data and recommends and ensemble of the best classification models with

their hyper-parameters optimized. In the proposed method, the corpus is transformed used the previously recommended representation and then given as input to AutoSklearn.

4.2.1 AutoSKlearn

Auto-sklearn is an AutoML solution that has succeeded in recent academic competitions [Feurer et al., 2015a, Guyon et al., 2017]. It is the state of the art on AutoML for tabular data and for that reason it was considered for this study. Auto-Sklearn is implemented in scikit-learn [Pedregosa et al., 2011], it initially comprised 15 classification algorithms, 14 preprocessing methods, and 4 data preprocessing methods. Similarly to Auto-WEKA [Thornton et al., 2013], Feurer et al. tackle the Combined Algorithm Selection and Hyper-parameter problem (CASH) or full model selection [Feurer and Hutter, 2018, Escalante et al., 2009], which they approach with SMAC, a three-based Bayesian optimization method. There are two key components that make Auto-Sklearn so competitive.

The first is based on *meta-learning*, complementary to Bayesian optimization, it is used to warmstart the optimization by quickly suggesting instantiations The second feature is the automated ensemble construction of models evaluated during optimization, when finding the best model instead of discarding the rest of the models found in the Bayesian optimization process, Feurer, et. al. store them and then build and ensemble using a greedy *ensemble selection* algorithm.

For the method proposed in this work the meta-learning component is skipped because all of the task are text classification and because it doesn't work well with sparse matrices. With BO the best classifiers are selected and an ensemble is returned as the recommended model. The output of this method combined with that of AutoSKlearn results in an automatic full pipeline recommendation for any text classification task.

4.3 Discussion

A two stage method was proposed in this chapter, being one of the first attempts to recommend full pipelines for text classification tasks some limitations were necessary. One of them is that even if the scope of the method is to solve any text mining task most of the datasets in the knowledge base are in English and some of the meta-features proposed only work for the English language, thus, there is no guarantee for the method to work on corpora that includes texts in another language.

Another limitation is the search space of textual representations, the method includes the largest search space until now but it is by no means an exhaustive pool of representations, as a consequence, the *optimal* pipeline is impossible to recommend if it needs an algorithm not included in the method; the restriction for the search space exists mainly due to the *hard* decisions of the meta-learning phase. The pool of representations could be expanded by extending this method with an optimization algorithm (see Chapters 2 and 3). Which is the best optimization method in the context of NLP is out of the scope of this research and is left as an open question for future work.

On the other side, the contributions of this method should be highlighted. A pipeline recommendation can be made for any type of text classification task without the intervention of human experts. The proposed characterization can be useful for future research in NLP since it can compare different tasks by incorporating information of a corpus and the language used in it. The ability to process raw text instead of pre-processed data is a novelty for AutoML in NLP, it is a crucial feature for any AutoML system yet almost unexplored to date.

Finally, the compilation of vast text classification tasks is another important contribution of the method. To the time of writing benchmark datasets for NLP are limited to some tasks where a large number of documents are included, the two

main type of tasks in such benchmark are sentiment analysis and topic categorization (for news). The bias for such tasks has left out a number of relevant tasks in NLP that have been studied for decades until the advent of Deep Learning; the complete compilation of datasets as well as the metadata extracted so far for different representations has been made available to the public. The new information is relevant for research in both NLP and AutoML fileds. More details about the diversity of the tasks can be found in Chapter 5.

Chapter 5

Experiments and results

To test the effectiveness of the proposed method a series of experiments were designed. First, the effectiveness of the meta-learning approach is validated, then, the whole method is tested and compared to state-of-the-art classification methods. This chapter describes the datasets used for such experimentation, it also details the configuration and discusses the results obtained.

5.1 Datasets

For the extraction of the meta-features and the experimental evaluation, 81 text corpora associated to different problems were collected. Each corpus was associated with a task-type-label according to the associated classification problem, where the considered labels were: *authorship analysis, sentiment analysis, topic/thematic tasks, irony* and *deception detection*. Figure 5.2 illustrates the distribution of the datasets as labeled by their task.

Tables 5.1, 5.2, 5.3 show the full list of datasets. Some of them are well known benchmarks (e.g. Yelp) while the rest can be found in competition sites like

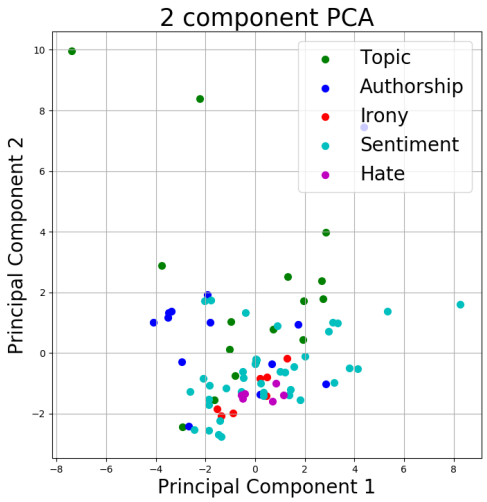

Figure 5.1: 2 PCA of the datasets described by their meta-features. Each corpus was tagged with a task-type-label.

Kaggle and SemEval. After pre-processing each corpus to share the same format and codification 72 meta-features were extracted for each of the 81 collections. To accelerate the feature extraction process the number of documents were limited to 90,000 per category, where these were randomly sampled from the categories of the corpus. The resultant matrix of size 81×72 comprises our *knowledge base* characterizing multiple corpora. All of the metadata reported in this work is publicly available at `https://github.com/jorgegus/autotext`.

5.2 Predicting the task

The novel characterization proposed in this research is not limited to select representations, the proposed set of meta-features can describe text mining tasks to compare them or study their differences. In order to evaluate the effectiveness of such characterization and making use of all the collected metadata from the variety of tasks, the problem of recognizing the classification task-type of a dataset is proposed.

Name	Task	# of docs	Voc size	# of classes
20 Newsgroups	Topics	18828	229710	20
Women's reviews	Author	23473	15153	8
Amazon cellphones	Sentiment	999	2241	2
Every song	Author	20779	48752	40
authorship_poetry	Author	200	9141	6
SouthPark episodes	Author	11953	14068	5
Spanish songs	Author	3947	35571	23
Bias Politics	Sentiment	5000	21328	2
Brown	Topics	500	48778	15
Progressive tweets	Topics	1159	5491	4
ccat	Author	1000	20416	10
Classic	Topics	7095	29518	4
Cyber trolls	Hate	20001	21193	2
Davidson hate	Hate	24678	24289	2
BBC News	Topics	2225	33771	5
BBC Summaries	Topics	2225	22921	5
Doctor deception	Sentiment	556	4453	2
Op_spam-	Sentiment	800	8819	2
Op_spam+	Sentiment	800	6548	2
Restauran reviews	Sentiment	400	5353	2
Deflate	Sentiment	11786	25616	5
Gender-microblog	Author	781	2439	2
Gender-twitter	Author	19953	50910	4
Imperium	Hate	6593	28031	2
Hate tweets	Hate	24783	41639	3
Iro-eduReyes	Irony	20000	32714	2
Iro-humReyes	Irony	19870	30485	2
iro-mohammad	Irony	1929	6040	2
Iro-polReyes	Irony	20000	31882	2
iro-riloff	Irony	2080	6132	2
Iro-semeval18	Irony	4466	10906	2
Kaggle hate	Hate	6594	25646	2

Table 5.1: List of datasets.

Name	Task	# of docs	Voc size	# of classes
Machado	Topics	246	79461	8
Hate-Malmasi	Hate	7162	14456	3
masc_tagged	Topics	389	43234	20
Medium papers	Topics	185	530	3
Movie reviews	Sentiment	2000	39768	2
polarity	Sentiment	1386	36614	2
Politic	Topics	5000	21328	9
Pros cons	Sentiment	45875	14015	2
Women's clothing	Sentiment	23486	15160	5
rawdata_cric	Author	158	13787	4
rawdata_fin	Author	175	15517	6
rawdata_nfl	Author	97	8940	3
rawdata_travel	Author	172	15560	4
Recommendations	Sentiment	23486	15160	2
Relevance economic news	Sentiment	8000	53162	3
Relevance short news	Sentiment	5007	20111	3
Reuters	Topics	13328	41600	84
Sarcasm Headlines	Irony	26709	25437	2
IMDB short	Sentiment	748	3401	2
Sent-semeval16	Sentiment	30631	36451	3
sent-semevalSA	Sentiment	6999	18042	3
Twitter-airline	Sentiment	14640	18614	3
Twitter-self-dirve	Sentiment	7156	18017	6
Short yelp	Sentiment	1000	2379	2
Sharktank	Sentiment	706	5175	2
smsspam	Sentiment	310	1610	2
Socialmedia disaster	Sentiment	10860	33768	2
Starter test	Sentiment	10876	33606	3
subjectivity	Sentiment	10000	21001	2
Tripadvisor reviews	Sentiment	17223	32423	5
Sentences polarity	Sentiment	10662	18408	2
Yahoo answers	Sentiment	1459998	180241	10

Table 5.2: List of datasets.

Name	Task	# of docs	Voc size	# of classes
YouTube	Sentiment	1956	5929	2
Yelp	Sentiment	699998	125757	5
Ag News	Topics	127598	64504	4
Kickstarter	Sentiment	215513	81252	2
News_Categories	Topics	124989	37183	30
Ohsumed	Topics	56984	79479	23
Short Amazon	Sentiment	568454	68831	5
Amazon	Sentiment	3649998	139289	5
sarcasm	Sentiment	1010826	62765	2
Amazon B	Sentiment	3999998	138968	2
Sentiment140	Sentiment	1600000	93115	2
Semeval17	Sentiment	62618	62304	3
Yelp B	Sentiment	597998	113897	2
Sogou news	Topics	509998	42991	5
Dbpedia	Topics	629998	199912	14
Victorian authorship	Author	53678	9977	45
Stanford	Sentiment	25000	95550	2

Table 5.3: List of datasets.

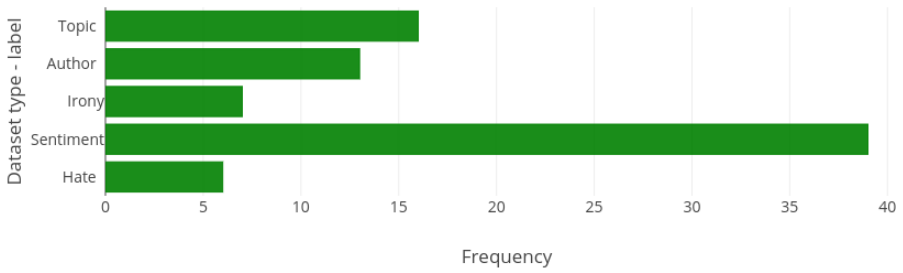

Figure 5.2: Tasks distribution.

The ultimate goal of this work is to automatically suggest pipelines for solving text classification problems. As a first step in such direction, it is shown in this section that the proposed meta-features can be used as predictive variables to learn models able to recognize the type of task associated to a dataset. Different text classification tasks can be derived given the same dataset, our set of meta-features also acknowledges this since some of the proposed measures provide statistical information about the classes.

In NLP it is empirically known that certain methods work better according to the type of task that is aimed, for instance, character-based n-grams are known to perform better than other representations in authorship attribution tasks because they determine better an author's style. Identifying correctly the type of task that is tackled is a fundamental step when modeling a text classification *pipeline*, thus it is proposed to automate this task in pursuit of an automated recommendation system.

Using the proposed set of meta-features, this problem was studied as both, a multiclass (predicting one of the 5 task labels) and a binary (distinguishing one label from the rest at a time) classification problem. The same classifiers discussed on section 4.1.2 were considered for the evaluation: Random Forest (RF), XGBoost (XG), linear Support Vector Machines (SVM) and KNN (1 neighbor).

For the evaluation a leave-one-out evaluation is adopted: 80 tasks were used for training and 1 for testing, repeating this process 81 times, each time changing the test task; the average performance over the 81 folds is reported. As evaluation measures for the binary classification approach accuracy and f_1 measure for the positive class are reported; in the case of the multiclass problem average accuracy and Macro-averaged-f_1 score are used instead.

	Accuracy				f_1			
Task / Model	XG	RF	SVM	KNN	XG	RF	SVM	KNN
Hate	0.94	0.94	0.93	0.91	0.29	0.29	0.00	0.36
Irony	**0.95**	0.93	0.91	0.93	**0.67**	0.25	0.00	0.5
Sentiment	**0.89**	0.85	0.67	0.69	**0.83**	0.77	0.00	0.60
Topics	0.86	**0.89**	0.80	0.77	0.62	**0.64**	0.00	0.30
Author	**0.90**	0.89	0.84	0.84	**0.60**	0.52	0.00	0.38
All 5-tasks	**0.77**	0.75	0.48	0.51	**0.64**	0.59	0.13	0.43

Table 5.4: Task prediction results with 72 meta-features

Table 5.4 shows the results obtained by the 4 classifiers. It can be seen that for XG and RF performance for all of the tasks is greater than random guessing. The high accuracy contrasted by moderate f_1 values reveals the models are favouring the majority class. In fact, high imbalance makes prediction quite difficult, specially for the *hate* and *irony detection* tasks where there are 6 and 7 positive examples, respectively.

An additional experiment involved a feature selection process prior to the classification stage. Since the number of possible meta-feature subsets is 2^72, it results impossible to experiment with all of them, hence, mutual information was used to rank the top 25 features. Then, subsets with the top ranked meta-features, from 2 to 25, were used for training and predicting. Table 5.5 shows the best performance obtained and the optimal number of meta-features found when performing feature

selection. It can be seen that there is a performance improvement after the selection of meta-features in all binary cases. Improvements are dramatic in terms of the f_1 measure in some cases (e.g., *Hate, Topics, Author*). Surprisingly, for some problems only few meta-features were required to achieve better performance, see, e.g., *Hate*. For the multiclass problem meta-feature selection did not improve the initial results on either evaluation measures.

Task	Model	K	Accuracy	f_1
Hate	RF	2	0.94 (+0%)	0.55 (+89.6%)
Irony	XG	15	0.96 (+1%)	0.73 (+8.9%)
Sentiment	XG	24	0.90 (+1%)	0.85 (+2.4%)
Topics	RF	3	0.90 (+1%)	0.75 (+17.1%)
Author	RF	3	0.91 (+1%)	0.70 (+16.6%)
5 tasks	XG	12	0.70 (-7%)	0.64 (+0%)

Table 5.5: Results with meta-feature selection

Table 5.6 shows the complete subsets of features considered for obtaining the results from Table 5.5. Meta-features are ordered by their mutual information values. It is hard to find a common pattern but it was found that some features are part of almost every subset: the *percentage of adverbs,* the *number of categories,* vocabulary overlapping in classes: *MRH,* and some statistic of *documents per category.* Hence showing the importance of the novel meta-features extracted from raw text. For hate detection and authorship analysis simple statistical measures appear to be better to describe the corpora, for the rest of the tasks the subsets that improved the original performance include a wide variety of meta-features from the groups.

Hate	Irony	Sentiment	Topics	Author	All 5 TASKS
number of categories	**number of categories**	**dpc_min**	**adverbs**	**dpc_min**	**number of categories**
dpc_min	**dpc_kurtosis**	**numerals**	**MRH**	**100pca_skewness**	**dpc_kurtosis**
Flesch reading ease	**adpositions**	**SMOG**	**pronouns**	**dpc_max**	**dpc_min**
dpc_kurtosis	**wpd_average**	**unmarked**	nouns	feature independence NB	**dpc_entropy**
zeros in matrix	**Flesch reading ease**	**pca_singular_sum**	punctuation marks	number of documents	**MRH**
voc_skewness	**zeros in matrix**	**pca_explained_variance**	dpc_entropy	pca_kurtosis	**adverbs**
dpc_entropy	**readability index**	**adpositions**	number of categories	pca_explained_ratio	**adjectives**
pca_explained_variance	**Kincaid grade**	**pca_max**	scholar grade	dpc_entropy	**wpd_average**
imbalance degree	**dpc_min**	**number of categories**	SMOG	pcac	**Flesch reading ease**
voc_kurtosis	**dpc_skewness**	**wpd_average**	data separability DT	pca_explained_variance	**pca_explained_variance**
	Linsear	**Gunning**			**zeros in matrix**
	MRH	**Linsear**			**SMOG**
	pca_singular_sum	**zeros in matrix**			
	voc_kurtosis	**Articles**			
	pca_min	**Flesch reading ease**			
		dpc_skewness			
		MRH			
		adverbs			
		Coleman-Liau			
		number of documents			
		nouns			
		wpd_entropy			
		pca_explained_variance			
		conjuctions			
		dpc_entropy			

Table 5.6: Meta-features identified as relevant after feature selection. We show the ranked features for each problem, in bold we show the features used for obtaining the results from Table 5.5.

5.3 Recommending textual representations

The 4 meta-learning strategies described in Chapter 4 were evaluated with unseen tasks following a leave-one-out setting, using the results from 60 representations in the rest of the tasks as knowledge to decide which representation to recommend. The objective for the strategies, then, is to select what in exhaustive search was found to be the *best* representation. the average performance achieved by our strategies is compared in 5 runs against the best solution found and the average performance of all of the considered representations. Table 5.7 shows the average performance for each strategy after 5 runs in terms of the average accuracy and average rank. Figure 5.3 depicts the performance of the method and the baselines in 9 corpora (these representative corpora were selected to cover a wide variety of tasks and because they are well known benchmarks).

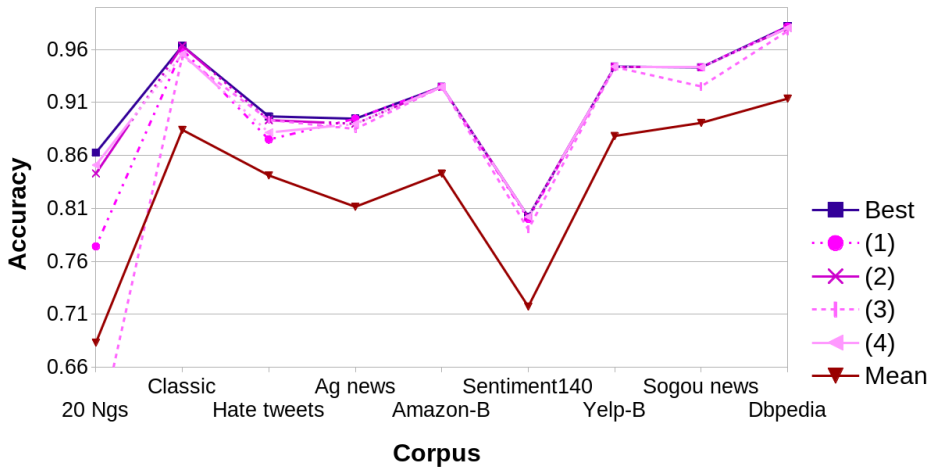

Figure 5.3: Accuracy of the strategies in 9 selected corpora.

The 4 strategies clearly outperform the mean of the performance of all representations and while in terms of average ranking they could be closer to the optimal, the average accuracy of (2) and (4) strategies was only 2% behind the best. (2) also found the best representation 35% of the time. Results show strong evidence that the

Method	Best	(1)	(2)	(3)	(4)	Mean
Avg Accu	77.06±0	73.75±0	**75.25±0.12**	73.34±0.34	**75.20±0.07**	68.45±0
Avg Rank	1.00±0	14.20±0	**8.71±0.46**	14.30±1.31	**8.51±0.34**	30.30±0
# of 1s	81.00±0	17.00±0	**25.80±0.45**	4.20±0.84	14.80±0.84	0.00±0

Table 5.7: Average accuracy [0,1] and average rank [60,1] of different strategies in 81 corpus, the last row indicates the number of times the best representation was predicted. (1) Nearest corpus, (2) classification, (3) performance regression, (4) rank prediction.

meta-learning approach finds relations between corpora and pipeline performances that exploits prior knowledge for the autonomous classification of texts.

From the 72 proposed meta-features we tested different subsets according to their Gini importance from the Random Forest used in strategy (2). A subset of 38 meta-features improved the results relatively by 8% with (2), and 38% with (1) in terms of average ranking. This subset was also compared against a subset comprised of 19 *traditional* meta-features used in related work. Using strategy (2) our subset outperformed the *traditional* one by almost 0.8% in average accuracy and 3 places in average rank. The results also showed a significant difference between using both subsets ($p<.001$ Student's t-test) The subset of 28 meta-features is detailed in Tables 6.1 and 6.2 in the appendix. And the results for the average ranking of the different subsets are depicted in Figure 5.4.

Method	(1)	(2)	(3)	(4)
Avg Accu	75.16±0	**75.39±0.13**	73.57±0.14	75.16±0.05
Avg Rank	8.68±0	**8.00±0.47**	14.42±0.53	9.05±0.25
# of 1s	**27.00±0**	**26.44±0.56**	6.40±1.34	16.00±0.87

Table 5.8: Results after meta-feature selection

The results for all the methods after applying (meta) feature selection are pre-

sented in Table 5.8. The number of meta-features was reduced by 50%. Strategies (3) and (4) maintained their performance while strategy (1) got an important improvement in its average accuracy and average rank. Strategy (2) remains as the best option for the meta-learning phase. The consistency of the results among the different approaches shows the effectiveness of the proposed set of meta-features.

Figure 5.4: Comparison of different meta-feature subsets.

In addition, the strategies were compared with commonly used representations such as pre-trained Word2Vec and Bag-of-Words outperforming them in average by 9% and 3% respectively, Figure 5.5 depicts this comparison (between strategy (4) and W2V) in the 9 corpora we selected. Despite the robustness of such common representations their performance can usually be improved by fine tuning some of their hyper-parameters or they are largely outperformed by another, as shown in the results the strategies are able to find these improvements.

After extensive experimentation with different approaches it was found that all strategies and all different subsets of meta-features give poor recommendations in some datasets. Specifically, for corpora in Table 5.9, the representations selected were never ranked in the top-10. Such results adhere to the intuitive idea behind

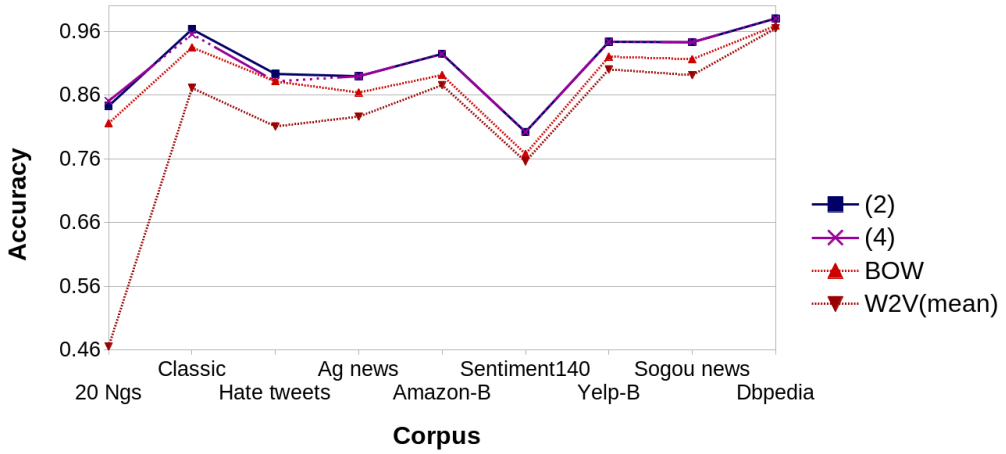

Figure 5.5: Accuracy comparison between (2), (4), Word2Vec, and BOW in 9 corpora.

meta-learning. The metadata extracted from each of these corpora, despite being associated to a text classification task, isn't related to the rest of the metadata stored in the *knowledge base*. How to determine a priori if a new dataset will work with the available metadata wasn't explored in this study but is an interesting question for further research.

Corpus	Size	Categories	Avg. rank
age-Wom	23473	8	20.88
cyber_trolls	20001	2	40.85
davidson-hate	24678	2	29.43
iro-mohammad	1929	2	26.84
medium-papers	185	3	27.90
relevance-economic-news	8000	3	20.20

Table 5.9: Datasets with complications. Avg. rank among all strategies and subsets is shown.

In some of these datasets the task is completely unique and there is no other similar task, that is the case of age-Wom, cyber_trolls, and relevance-economic-news

corpora, where their associated task: profiling women by age ranges given their reviews in some products, detecting *trolls* in social media, and deciding whether a news will have a relevant impact in the economy of a country, are very specific and data for similar problems will be hard to find.

On the other hand, the rest of the datasets from this analysis present tasks that are rather common in the *knowledge base*: irony detection and topic categorization. It is difficult to identify a single reason for the failure of the method, by looking at the texts from these corpora they are very similar to texts from the similar tasks, however, the representation that worked for the rest isn't the best for these special cases. Such behaviour isn't alarming, it is actually expected from the *hard* meta-learning approach proposed.

In spite of the promising results of the proposed method, the search space isn't exhaustive, thus, finding the *perfect* representation or pipeline couldn't be possible with the current approach. Nevertheless, the experimentation of this work is extensive and the method presented here presents results that are solid enough to continue research in this direction. The special cases where it fails can be easily addressed by extending this method with state-of-the-art optimization techniques allowing the enlargement of the search space or the fine-grain hyper-parameter optimization of the current algorithms in consideration, some examples of this approach (in other areas) can be found in Chapter 3.

5.4 Recommending pipelines

5.4.1 Comparison with state-of-the-art

The final method: AutoText (meta-learning of textual representations + AutoSKlearn) was compared against some of the most recent text classification methods in the lit-

erature, all of which are *Deep Learning* methods. Following the same validation that such works make, the proposed method was trained and tested using the benchmark datasets used for Deep Learning, these are also the largest datasets available for the meta-learning phase.

As evaluation measure most works report accuracy, each score is taken directly from what is reported in the cited papers, for AutoText the dataset being tested on is removed from the *knowledge base*. Additionally, the experiments are not limited to the recommendation of a single representation but also a concatenation of the top 3 best representations, experiments are performed using strategies (2) and (4) and using the 72 meta-features and the sub-set of 38 after feature selection.

Method	AG	Dbpedia	YelpB	Yelp	Yahoo	Amazon	AmazonB
CNN	92.36	98.69	95.64	62.05	73.43	**59.57**	**95.07**
LEAM	92.45	99.02	95.31	64.09	**77.42**	-	-
ULMfit	**94.99**	**99.2**	**97.84**	**70.02**	-	-	-
AutoText1	83.88	93.23	85.98	45.36	53.98	46.24	80.64
AutoText2	81.97	85.90	83.78	49.17	29.12	40.03	79.39
AutoText3	84.15	87.62	88.26	46.09	10	20	50

Table 5.10: Results in benchmark datasets. For DL methods, claimed results from the original papers are used for comparison.

While most works focus on solving a specific task some propose a more *general* classification method, those were selected for the comparison against the method from this thesis, namely: character level Convolutional Neural Networks(CNN) [Zhang et al., 2015], Label Embeddings Attentive Model (LEAM) [Wang et al., 2018], and Universal Language Model Fine-tuning (ULMfit) [Howard and Ruder, 2018]. Table 5.10 includes the results for this comparison. Variations of Autotext are listed below:

- AutoText1: Strategy (4), 72 meta-features
- AutoText2: Strategy (2), 38 meta-features
- AutoText3: Strategy (4), 72 meta-features, top-3 representations

For the largest datasets, AutoText performs well but it is outperformed by the state-of-the-art. There was a slight improvement in some of the results when a concatenation of 3 representations was used, but this approach failed for 3 datasets because of the huge dimensionallity. Deep Learning approaches rely heavily on the availability of extensive data and do not generalize well to different tasks. To demonstrate this point one of the SOTA methods was implemented and tested on smaller datasets. Table 5.11 shows the comparison of ULMfit and AutoText1 in a wider variety of tasks and smaller datasets.

Method	20NGs	Movie reviews	Reuters	SemEval18
ULMfit	79.74	77.61	72.53	62.42
AutoText1	**85.09**	**84.90**	**72.79**	**64.82**

Table 5.11: Results in diverse tasks. Accuracy achieved by each method is reported after training with 70% of the data and testing with the remaining 30%.

In this setting AutoText outperforms ULMfit most of the time, it was also found that with very small datasets a Deep Learning method can be easily overfitted achieving performance of a random classifier making it necessary to fine-tune the hyper-parameters of such models whereas the proposed method requires little to none human intervention; for a fair comparison and because of memory limitations, only the last layer of the ULMfit's language model and the classification neural network were fine-tuned (instead of *gradual unfreezing*), the rest of the hyper-parameters of ULMfit were set to default [Howard and Ruder, 2018]. The aim of this comparison is not to criticize Deep Learning methods, as a mater of fact a *tailored*

model could significantly improve its results, but rather to show evidence that a more classical pipeline can achieve similar performance when selected optimally and in a data driven fashion, potentially saving thousands of computation hours due to the inherent requirements of fine-tuning DL models.

Chapter 6

Conclusions and future work

6.1 Conclusions

This one of the first studies so far in NLP regarding meta-learning and AutoML research. The findings and proposals of this work make several noteworthy contributions to both fields. Furthermore, they have enhanced the understanding of the construction of pipelines for text mining tasks. Several questions were out of the scope for this work, however, this research could serve as base for future studies in AutoML for text classification.

A novel characterization for text classification corpora is proposed using a set of 72 meta-features and evaluated in 81 datasets that combine generic meta-learning and natural language processing techniques. Experimental results showed that this metadata can be used to recommend models with acceptable performance in many tasks.

Metadata extracted from extensive experimentation as well as the compilation of datasets was made public; this information along with the proposed characterization can potentially help understand differences between the diverse text mining

tasks, determine existing biases in the construction of text corpora, or defining new benchmarks for text classification algorithms with a broad scope.

The problem of automatically predicting the type text classification task is proposed and approached from meta-features derived from text. Experimental results demonstrate that the proposed meta-features entail discriminative information that could be useful for other meta-learning tasks. Results of a meta-feature selection analysis has shown that *traditional* meta-features are not good enough to characterize datasets by themselves, proving the effectiveness of the newly introduced ones.

As part of the developed method, a meta-learning stage was also proposed, it takes as input an unprocessed corpus and without human intervention builds a model to solve a text classification task, first focusing on the selection of a vector-based representation and then employing auto-sklearn to optimize a classification model. The results show empirically that this approach is able to characterize tasks and approximate an optimal representation.

The final method recommends a full pipeline for text classification problems. In extensive experimentation it obtained satisfactory results among a large number of tasks without the traditional tuning of a human expert, the method has also been made publicly available in order to encourage further extensions to it and develop a robust system for both non-experts in NLP and data scientist that seek to reduce the devoted time to adjust hyper-parameters.

Evidence also supports the effectiveness of the method by achieving results comparable to state-of-the-art Deep Learning approaches in datasets with few samples. To conclude this document some suggestion of future research directions are discussed below.

6.2 Contributions

1. A method that automatically recommends a text representation, a classifier, and its hyper-parameters values given a corpus. The proposed method takes as input a raw set of documents and automatically transforms it, trains a classifier, and produces predictions. It represents the first effort for solving any text classification task without human input, further research can also extend the method to include state-of-the-art optimization algorithms or to aim for solving other text mining problems.

2. A novel set of meta-features that describe a text classification task. The characterization showed to be effective for describing tasks and two problems were addressed exploiting this information, however, it would be interesting to explore other problems with this approach. For instance. What makes different a task from another? Is this metadata useful for generating useful datasets to aid other Machine Learning tasks?

3. The problem of text classification task-type prediction is introduced. Manually determining this is usually the first step when dealing with a new task. Future work for this problem could help to understand if this step is helpful for building efficient pipelines or human bias to such decisions could potentially ignore promising algorithms.

4. An empirical study of the different representations used across types of tasks. The proposed method relies on the metadata obtained from this study, while being the largest to date, it can be extended to include DL models or exploited to understand different steps in a classification pipeline.

Derived from the research in this area and from this thesis three papers have been or will be presented:

Jorge G Madrid, Hugo Jair Escalante, Eduardo F Morales, Wei-Wei Tu, Yang Yu, Lisheng Sun-Hosoya, Isabelle Guyon, and Michèle Sebag. Towards AutoML in the presence of Drift: first results. In Workshop AutoML 2018 @ ICML/IJCAI-ECAI, Stockholm, Sweden, July 2018. Pavel Brazdil, Christophe Giraud-Carrier, and Isabelle Guyon.

Jorge G Madrid, Hugo Jair Escalante and Eduardo F Morales. Meta-learning of textual representations. In Workshop AutoML 2019 @ ICML, Long Beach, USA, July 2019. Katharina Eggensperger, Matthias Feurer, Frank Hutter, and Joaquin Vanschoren. Also presented in ECMLPKDD Workshop on Automating Data Science (ADS).

Jorge G Madrid and Hugo Jair Escalante. Meta-learning of text classification tasks. Accepted in the 24th Iberoamerican Congress on Pattern Recognition (CIARP 2019).

6.3 Future work

This research has thrown up many questions in need of further investigation. The representations in the search space are by far the most varied compared to related work but because it is a prominent research area where new representations are proposed every year. What other representations should be explored for AutoML? The first stage of the proposed method can be used to *warm-start* an optimization technique with the proposed meta-learning setting allowing to expand the search space and ideally finding pipelines that perform better than those designed by humans, as shown by AutoML in other fields. This work approaches the problem by sequentially recommending a representation and a classifier, the question of whether optimizing both type of algorithms at the same time will perform better remains.

Characterizing text mining tasks is one of the novelties of this work. Two

problems where tackled with the proposed description vector: identifying the type task associated with a corpus and recommending text representations based on what worked previously for similar tasks. Future work might explore existing problems or suggest new ones with the proposed characterization.

The findings suggest that meta-features related to language richness and complexity outperform those using only statistics from the dataset. The set of meta-features discussed in this document is not a definitive list, what other language features could be exploited to characterize text mining tasks is another interesting research question that could be studied.

Some topics out of this thesis scope also need further research. Currently Deep Learning models are very time-consuming, in AutoML for DL, specifically, Neural Architecture Search, this situation is aggravated. Computational requirements of DL should not be disregarded. Reducing the experimentation time is one of the main motivations for AutoML. In addition, such demanding conditions have a negative impact in real life scenarios, for instance, small research teams or companies could not afford the required hardware or would lose a significant part of their financing for a small gain in accuracy.

To continue the wide adoption of Machine Learning in different areas, efficient algorithms and methods have to be developed. The extensive meta-learning study from this thesis might provide some insights to the construction of optimal text classification pipelines. Further investigation in AutoML is needed for optimizing existing DL methods but also to automatically determine what are the correct conditions for adopting a DL model over traditional pipelines. The proposed characterization of tasks provides a practical mechanism to approach both research directions.

This work comprises a first attempt towards the automated recommendation of full text classification pipelines. The source code of the method is available under an open source license at: https://github.com/jorgegus/autotext.

Bibliography

[Abdulrahman et al., 2018] Abdulrahman, S. M., Brazdil, P., van Rijn, J. N., and Vanschoren, J. (2018). Speeding up algorithm selection using average ranking and active testing by introducing runtime. *Machine learning*, 107(1):79–108.

[Aggarwal and Zhai, 2012] Aggarwal, C. C. and Zhai, C. (2012). *A Survey of Text Classification Algorithms*, pages 163–222. Springer US, Boston, MA.

[Ali and Smith, 2006] Ali, S. and Smith, K. A. (2006). On learning algorithm selection for classification. *Applied Soft Computing*, 6(2):119–138.

[Allahyari et al., 2017] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*.

[Andrychowicz et al., 2016] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989.

[Bengio, 2000] Bengio, Y. (2000). Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900.

[Bergstra and Bengio, 2012] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

[Bergstra et al., 2011] Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

[Bozdogan, 1987] Bozdogan, H. (1987). Model selection and akaike's information criterion (aic): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370.

[Brazdil and Giraud-Carrier, 2018] Brazdil, P. and Giraud-Carrier, C. (2018). Metalearning and algorithm selection: progress, state of the art and introduction to the 2018 special issue.

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

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

[das Dôres et al., 2018] das Dôres, S. C. N., Soares, C., and Ruiz, D. (2018). Bandit-based automated machine learning. In *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 121–126. IEEE.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

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

[Domhan et al., 2015] Domhan, T., Springenberg, J. T., and Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[Elsken et al., 2018] Elsken, T., Metzen, J. H., and Hutter, F. (2018). Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*.

[Escalante et al., 2009] Escalante, H. J., Montes, M., and Sucar, L. E. (2009). Particle swarm model selection. *Journal of Machine Learning Research*, 10(Feb):405–440.

[Escalante et al., 2017] Escalante, H. J., Villatoro-Tello, E., Garza, S. E., López-Monroy, A. P., Montes-y Gómez, M., and Villaseñor-Pineda, L. (2017). Early detection of deception and aggressiveness using profile-based representations. *Expert Systems with Applications*, 89:99–111.

[Falkner et al., 2018] Falkner, S., Klein, A., and Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1436–1445.

[Ferreira and Brazdil, 2018] Ferreira, M. J. and Brazdil, P. (2018). Workflow recommendation for text classification with active testing method. In *Workshop AutoML 2018@ ICML/IJCAI-ECAI*.

[Feurer and Hutter, 2018] Feurer, M. and Hutter, F. (2018). Hyperparameter optimization. In [Hutter et al., 2018], pages 3–38. In press, available at http://automl.org/book.

[Feurer et al., 2015a] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015a). Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970.

[Feurer et al., 2015b] Feurer, M., Springenberg, J. T., and Hutter, F. (2015b). Initializing bayesian hyperparameter optimization via meta-learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Finn et al., 2017] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org.

[Frazier, 2018] Frazier, P. I. (2018). A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*.

[Fusilier et al., 2015] Fusilier, D. H., Montes-y Gómez, M., Rosso, P., and Cabrera, R. G. (2015). Detection of opinion spam with character n-grams. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 285–294. Springer.

[Golovin et al., 2017] Golovin, D., Kochanski, G., and Karro, J. E. (2017). Black box optimization via a bayesian-optimized genetic algorithm. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. To be submitted to Opt2017 Optimization for Machine Learning, at NIPS 2017.

[Golub and Reinsch, 1971] Golub, G. H. and Reinsch, C. (1971). Singular value decomposition and least squares solutions. In *Linear Algebra*, pages 134–151. Springer.

[Gomez et al., 2017] Gomez, J. C., Hoskens, S., and Moens, M.-F. (2017). Evolutionary learning of meta-rules for text classification. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 131–132. ACM.

[Guo et al., 2006] Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. (2006). Using knn model for automatic text categorization. *Soft Computing*, 10(5):423–430.

[Guyon et al., 2015] Guyon, I., Bennett, K., Cawley, G., Escalante, H. J., Escalera, S., Ho, T. K., Macia, N., Ray, B., Saeed, M., Statnikov, A., et al. (2015). Design of the 2015 chalearn automl challenge. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[Guyon et al., 2017] Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H., Escalera, S., Liu, Z., Jajetic, D., Ray, B., Saeed, M., Sebag, M., et al. (2017). Analysis of the automl challenge series 2015-2018.

[Hansen, 2016] Hansen, N. (2016). The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

[Howard and Ruder, 2018] Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

[Hutter et al., 2011] Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer.

[Hutter et al., 2018] Hutter, F., Kotthoff, L., and Vanschoren, J., editors (2018). *Automatic Machine Learning: Methods, Systems, Challenges*. Springer. In press, available at http://automl.org/book.

[Iyyer et al., 2015] Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

[Jamieson and Talwalkar, 2016] Jamieson, K. and Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248.

[Jin et al., 2019] Jin, H., Song, Q., and Hu, X. (2019). Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. ACM.

[Karnin et al., 2013] Karnin, Z., Koren, T., and Somekh, O. (2013). Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pages 1238–1246.

[Kessler et al., 1997] Kessler, B., Nunberg, G., and Schutze, H. (1997). Automatic detection of text genre. In *35th Annual Meeting of the Association for Computational Linguistics*, pages 32–38.

[Klein et al., 2017] Klein, A., Falkner, S., Bartels, S., Hennig, P., and Hutter, F. (2017). Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial Intelligence and Statistics*, pages 528–536.

[Kohavi and John, 1995] Kohavi, R. and John, G. H. (1995). Automatic parameter selection by minimizing estimated error. In *Machine Learning Proceedings 1995*, pages 304–312. Elsevier.

[Kordík et al., 2018] Kordík, P., Černỳ, J., and Frỳda, T. (2018). Discovering predictive ensembles for transfer learning and meta-learning. *Machine Learning*, 107(1):177–207.

[Lam and Lai, 2001] Lam, W. and Lai, K.-Y. (2001). A meta-learning approach for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 303–309. ACM.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

[Lee et al., 2007] Lee, S.-I., Chatalbashev, V., Vickrey, D., and Koller, D. (2007). Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of the 24th international conference on Machine learning*, pages 489–496. ACM.

[Lessmann et al., 2005] Lessmann, S., Stahlbock, R., and Crone, S. F. (2005). Optimizing hyperparameters of support vector machines by genetic algorithms. In *IC-AI*, pages 74–82.

[Lewis et al., 2004] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

[Li and Jamieson, 2018] Li, L. and Jamieson, K. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18:1–52.

[López-Monroy et al., 2015] López-Monroy, A. P., Montes-y Gómez, M., Escalante, H. J., Villaseñor-Pineda, L., and Stamatatos, E. (2015). Discriminative subprofile-specific representations for author profiling in social media. *Knowledge-Based Systems*, 89:134–147.

[Lorenzo et al., 2017] Lorenzo, P. R., Nalepa, J., Kawulok, M., Ramos, L. S., and Pastor, J. R. (2017). Particle swarm optimization for hyper-parameter selection in deep neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 481–488. ACM.

[Lovins, 1968] Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31.

[Nakov et al., 2016] Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., and Stoyanov, V. (2016). Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*, pages 1–18.

[Olson et al., 2016a] Olson, R. S., Bartley, N., Urbanowicz, R. J., and Moore, J. H. (2016a). Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 485–492, New York, NY, USA. ACM.

[Olson et al., 2016b] Olson, R. S., Urbanowicz, R. J., Andrews, P. C., Lavender, N. A., Kidd, L. C., and Moore, J. H. (2016b). *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I*, chapter Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pages 123–137. Springer International Publishing.

[Pang et al., 2008] Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

[Pedregosa et al., 2011] 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.

[Pennebaker et al., 2015] Pennebaker, J. W., Boyd, R. L., Jordan, K., and Blackburn, K. (2015). The development and psychometric properties of liwc2015. Technical report.

[Pinto, 2008] Pinto, D. (2008). On clustering and evaluation of narrow domain short-text corpora. *PhD. UPV*.

[Porter, 1997] Porter, M. F. (1997). An algorithm for suffix stripping. In *Readings in information retrieval*, pages 313–316. Morgan Kaufmann Publishers Inc.

[Quanming et al., 2018] Quanming, Y., Mengshuo, W., Hugo, J. E., Isabelle, G., Yi-Qi, H., Yu-Feng, L., Wei-Wei, T., Qiang, Y., and Yang, Y. (2018). Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*.

[Rangel et al., 2016] Rangel, F., Rosso, P., Verhoeven, B., Daelemans, W., Potthast, M., and Stein, B. (2016). Overview of the 4th author profiling task at pan 2016: cross-genre evaluations. In *Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings/Balog, Krisztian [edit.]; et al.*, pages 750–784.

[Ravi and Larochelle, 2016] Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning.

[Rill-García et al., 2018] Rill-García, R., Villaseñor-Pineda, L., Reyes-Meza, V., and Escalante, H. J. (2018). From text to speech: A multimodal cross-domain approach for deception detection. In *International Conference on Pattern Recognition*, pages 164–177. Springer.

[Rivolli et al., 2018] Rivolli, A., Garcia, L. P., Soares, C., Vanschoren, J., and de Carvalho, A. C. (2018). Towards reproducible empirical research in meta-learning. *arXiv preprint arXiv:1808.10406*.

[Schmidhuber, 1987] Schmidhuber, J. (1987). Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*.

[Schmidt and Wiegand, 2017] Schmidt, A. and Wiegand, M. (2017). A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

[Shahriari et al., 2016] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

[Snoek et al., 2012] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

[Stamatatos et al., 2000] Stamatatos, E., Fakotakis, N., and Kokkinakis, G. (2000). Automatic text categorization in terms of genre and author. *Computational linguistics*, 26(4):471–495.

[Sun et al., 2013] Sun, Q., Pfahringer, B., and Mayo, M. (2013). Towards a framework for designing full model selection and optimization systems. In *International Workshop on Multiple Classifier Systems*, pages 259–270. Springer.

[Thornton et al., 2013] Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 847–855, New York, NY, USA. ACM.

[Uysal and Gunal, 2014] Uysal, A. K. and Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112.

[Vanschoren, 2018] Vanschoren, J. (2018). Meta-learning. In [Hutter et al., 2018], pages 39–68. In press, available at http://automl.org/book.

[Wang et al., 2018] Wang, G., Li, C., Wang, W., Zhang, Y., Shen, D., Zhang, X., Henao, R., and Carin, L. (2018). Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331.

[Wolpert et al., 1997] Wolpert, D. H., Macready, W. G., et al. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

[Yan, 2009] Yan, J. (2009). *Text Representation*, pages 3069–3072. Springer US, Boston, MA.

[Yogatama et al., 2015] Yogatama, D., Kong, L., and Smith, N. A. (2015). Bayesian optimization of text representations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2100–2105.

[Zhang et al., 2015] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

[Zöller and Huber, 2019] Zöller, M.-A. and Huber, M. F. (2019). Survey on automated machine learning. *arXiv preprint arXiv:1904.12054*.

Appendix

6.4 Meta-feature subsets

Meta-feature selection
average word length
document per category:
min
max
average
standard deviation
average / stdev
entropy
word per document:
average
skewness
entropy
Imalance Degree
SEM
UVB
SVB
MRH_J
VDR

Table 6.1: 38 Meta-features selected by Gini importance

Meta-feature selection
max vocabulary
average vocabulary
sd vocabulary
skweness vocabulary
avg/stdev vocabulary
pca:
singular values sum
explained ratio
explained variance
explained variance (1)
pca max
pca skewness
pca kurtosis
data sparsity
data separability
linear separability
% of zeros
% of adpositions
% of adverbs
% of conjunctions
% of nouns
% of numbers
% of untagged words
difficult words

Table 6.2: 38 Meta-features selected by Gini importance