Workflow Recommendation for Text Classification Problems

by

Maria João Ferreira

Master Dissertation in Data Analytics

Supervised by
Professor Alípio Jorge
Professor Pavel Brazdil

Faculdade de Economia
Universidade do Porto
2017

Aos meus pais.

Bibliographic note

Maria João Ferreira, born in Vila Nova de Famalicão, achieved a Bachelor in Economics from the Faculty of Economics of the University of Porto. From 2014 to 2015, she worked as Junior Researcher at cef.up – University of Porto research center in Economics and Finance.

For the project detailed in this dissertation, Maria João was a Graduate Researcher at INESC.TEC and is currently, employed at Farfetch as a Business Analyst.

Acknowledgments

I would also like to acknowledge the support of project NanoSTIMA: Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring and Analytics/NORTE-01-0145-FEDER-000016, which is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

I would like to thank Professor Pavel Brazdil, who was such a great supervisor throughout this project. Always available, eager to help, very supportive and understanding, he has contributed deeply to my betterment as a researcher and professional.

I am also very grateful to my colleagues and friends at INESC.TEC, in particular Miguel Cachada, Salisu Abdulrahman, Tiago Cunha and Pedro Abreu whose input, support and specially those sanity checks along the way made this thesis possible. Thank you.

I would also like to use this opportunity to thank my manager, Paula Brochado and all the other members of the *T-Recs* team for the support and understanding during the critical phase of writing this dissertation. I am really lucky to be surrounded daily by you awesome people.

Gostaria de agradecer em particular aos meus pais, que para além de serem os meus maiores pilares e melhor exemplo, são também os melhores amigos e confidentes que irei alguma vez ter. Obrigada à minha mãe por aturar pacientemente as minhas trocas de humor e ataques de panico. Obrigado ao meu pai por me chamar a razão e conseguir sempre acalmar-me (não obstante de me irritar profundamente ao fazê-lo). Obrigada por me terem trazido até aqui e por todas as coisas que não tenho espaço ou palavras para agradecer.

Resumo

Desde que a escrita foi inventada, o texto é a ferramenta humana para comunicação através do tempo e do espaço. Classificação de texto é a tarefa de catalogar documentos de acordo com categorias pré-definidas. Esta tarefa foi sempre importante para a organização da vida humana e para a sociedade.

A internet, e em particular a internet móvel, tornou possível escrevermos e lermos em qualquer lugar. A quantidade de texto que é criado e guardado todos os dias tornou imperativo que a tarefa de classificação de textos seja automatizada. As aplicações desta tarefa são inúmeras: desde filtragem de emails até encaminhamento de *queries* ou personalização de conteúdo em redes socias. No entanto, nem sempre é claro qual a melhor forma abordar este problema. As possibilidades podem tornar-se esmagadoras particularmente quando o tempo é escasso se consideramos métodos de pré-processamento, algoritmos de classificação e até afinação de parâmetros.

Neste estudo, olhamos para 384 combinações de métodos de pré-processamento e algoritmos de classificação – processos de classificação, aplicados a 50 tarefas de classificação de texto. A performance e o tempo de execução destes processos foram usados para construir rankings, em que os processos são avaliados por uma medida que combina a taxa de acerto e o tempo de execução, A3R. Adoptamos dois métodos de ranking que usam uma abordagem de meta-aprendizagem, *Average Ranking* e *Active Testing* (Abdulrahman, 2017) ao problema deselecção de processos de classificação de texto. Neste estudo também propomos uma nova ferramenta

de análise dos constituintes dos processos, uma meta-regressão. Esta ferramenta mostrou resultados muito promissores, ao ser capaz de medir com precisão quais os elementos mais benéficos dos processos e identificar os que são irrelevantes.

Palavras-Chave: classificação de textos, meta-aprendizagem, selecção de processos de classificação, análise de regressão.

Abstract

Ever since writing was invented, text is used to communicate crossing the boundaries of time and space. Text classification is the task of categorising documents according to pre-defined labels. This task has always played an important role in organizing human life and society.

Internet and in particular, mobile internet has enabled us to write and read just about anywhere. The amount of text created and stored every day has made it imperative that the task of classifying text is automated. From email filtering to query routing or social media content personalization, the applications of text classification are boundless. However, it is not always entirely clear how to solve these problems. The options may seem overwhelming, particularly when time is of the essence if preprocessing methods, classification algorithms and even hyperparameter tuning are considered.

In this study we looked at 384 combinations of preprocessing methods and classification algorithms – *workflows*, applied to 50 text classification tasks. The performance and runtime of these workflows were used to construct rankings, where workflows were graded according to a measure that combined accuracy and runtime, A3R. We adapted two methods for ranking that use a metalearning approach, *Average Ranking* and *Active Testing* (Abdulrahman, 2017) to the workflow selection problem for text classification. In this study we also propose a new tool for analysis of the elements of workflows, a *meta-regression*. This tool has shown very promising results by accurately measuring the most beneficial elements of workflows and

identifying the redundant or irrelevant ones.

Keywords: text classification, metalearning, classification workflow selection problem, regression analysis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In a world with multiple available options and limited information, how to choose the best course of action? This is the set-up for the *cold-start* problem (Schein et al., 2002) and this dissertation puts forward a solution for this problem – for at least *some* scenarios. The scenarios in question are classification problems in machine learning, specifically text classification problems.

The focus of this dissertation lies in finding what is the most appropriate way to tackle a text classification problem. *Tackling* refers here to the complete pipeline of these tasks, from the preprocessing methods applied to the texts to the algorithm that will perform the classification. The pipelines of operations are referred here as *workflows*.

We propose a methodology that guarantees an ordered list of workflows that can be executed with the objective to identify the best alternative. We have also conceived a method that produces insight about which constituents of the workflows are more important for the good performance of the classification task. This allows us to confront the *cold start problem* in text classification from two fronts: listing informed suggestions and also considering which options have more impact on the good solution of the problem.

1.1 The specificities of text classification tasks

Text classification has always played an important part in organizing human life and society. Ever since writing was invented, text was used to communicate crossing boundaries of time and space. The internet and in particular *mobile* internet, allows people to write and read just about anywhere and this creates massive amounts of information. The amount of text created and stored everyday is at the level of numbers that we are simply not able to comprehend. Document categorization (Cai and Hofmann, 2004), news filtering (Konstan et al., 1997), document routing (Joseph, 2002) and personalization of social media content (Agichtein et al., 2008) are just some of the current applications of text classification. We need automatic systems to help us to carry out these tasks.

Classification tasks in machine learning consist of automatically assigning the correct labels to some instances based on the study of previous examples. When applied to text documents, it consists of labelling new text documents with their category. Current approaches involve training a classifier on previous instances and constructing a model that captures the regularities in the examples of each category. This model is able to identify these regularities in new examples. Moreover, in the case of text classification in particular, the texts must be processed before being used in training due to the particular *unstructured* format of this data. Figure 1.1 represents a simplified overview of the knowledge discovery process on text classification.

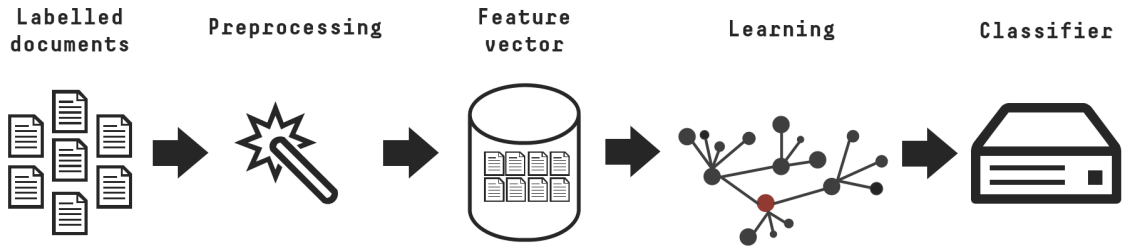

Figure 1.1: Knowledge discovery process in text classification.

Each of the steps featured in our figure can be implemented in different ways and the result will depend on the decisions made.

It is not possible to find one best solution for all problems as the *No Free Lunch Theorem* (Giraud-Carrier and Provost, 2005; Schaffer, 1994) informs us. A good balance for this problem comes with the metalearning (Brazdil et al., 2008) approach to the selection of algorithms, that we have adapted to the selection of workflows. Metalearning utilizes information gathered in past learning episodes and this way guides users by providing informed suggestions for a new learning problem.

However, not enough work has been done on applying the metalearning approach specifically to the text classification problem. We believe that preprocessing for text classification is fundamentally different from preprocessing for other classification tasks. Not only there are specific methods for text classification, but also efficient dimension reduction is much more urgent on these tasks. For instance, the exhaustive study of classification pipelines performed in the context of the creation of Auto-SKLearn (Feurer et al., 2015) does include text classification tasks. However, this study ignored the specific nature of preprocessing methods for text classification problems, using already *structured* data.

Although some surveys have been done on the performance of classification algorithms on this type of tasks (such as Sebastiani (2002); Namburu et al. (2005); Aggarwal and Zhai (2012)) and preprocessing methods (as seen in Yang and Pedersen (1997); Forman (2003)), we were not able to find a study that fully explored and related these two distinct phases specifically for text classification tasks.

The reasoning behind preprocessing methods in text classification often stems from semantics. For instance, the argument for removing stop-words from documents is that since these type of words appear very frequently on every document (they consist of prepositions, pronouns, adverbs) they do not add any value for the classification task. However, could there be an algorithm that would achieve

better results if we kept those frequent words? Are there domains in which this preprocessing method is more relevant than others? The lack of satisfying answers to questions like these has served as a motivation to perform our own study on the subject.

1.2 Objectives and contribution of this thesis

This study was developed with a several goals in mind and constitutes an effort to make a contribution in the following ways:

- ***Empirical study of a wide variety of text classification tasks and workflows***: In this study we present the empirical findings from testing 384 distinct workflows on 50 text classification datasets.
- ***Application of algorithm selection methods to text classification***: We follow methodology developed in the context of algorithm selection for general classification task and adapt it to the workflow selection for text classification problems upholding the principles of the original methods.
- ***Produce workflow recommendations to text classification problems***: our system is able to produce rankings of workflows for TC problems. We have considered different ranking methods and a measure that combines accuracy rate and runtime, A3R. This was compared to the baseline that used accuracy only.
- ***Provide a new tool for analysis of elements of workflows***: We have proposed a novel tool for analysing the utility of the constituents of the workflows, based on their impact on the accuracy rate of workflows.

1.3 Overview of the organization of this dissertation

This dissertation is organized into five chapters. The next chapter, Chapter 2 surveys related work. Chapter 3 describes the methodology we propose and adopted in this study. Then, in Chapter 4 the experimental *set-up* and results are presented and analysed. Finally, in Chapter 5 we present the conclusions and future work.

Chapter 2

Literature review

2.1 Text classification

As established in Chapter 1, this thesis will explore the text classification (TC) problem. This application consists of the adaptation of the common classification task onto text documents. This task amounts to an automated process of labelling unseen document instances based on the analysis of previous examples (Mitchell, 1997; Feldman and Sanger, 2006).

Figure 2.1 shows an overview of the pipeline of a TC task. Starting from training examples of text, which have already been assigned labels. These are transformed into a feature vectors format and ran through a machine learning algorithm. The algorithm identifies patterns within the training examples and constructs a predictive model. When confronted with a new instance of text or document, the same feature vector transformation is performed and the predictive model is able to automatically assign *expected* labels.

Applications of text classification The applications of text classification spread in a wide variety of domains and contexts. One of the most common is *Spam Filtering* used by e-mail providers, in which the category of *junk mail* is automatically

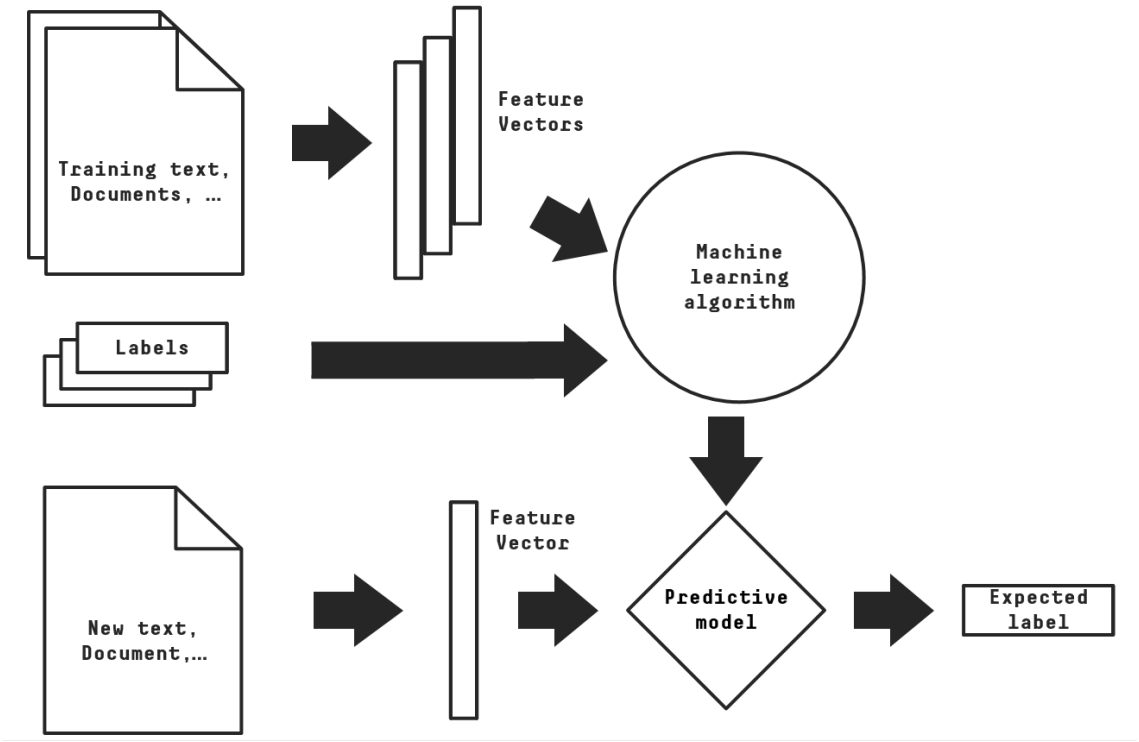

Figure 2.1: Pipeline of text classification problems - Adapted from Rehurek (2014).

assigned, allowing the user never to view those (Androutsopoulos et al., 2000). *Routing* of customer service tickets yields huge gains in efficiency responding to queries (Sebastiani, 2002; Joseph, 2002). News articles organization, where such articles can be directly assigned into a category such as *sports* or *politics* is another often mentioned TC task (Aggarwal and Zhai, 2012). This can be broadened into text organization and retrieval by applying it to other less specific domains, like for instance digital libraries, web collections, scientific literature or even social networks feeds (Agichtein et al., 2008).

In the next sections the formal definition of these type of learning problems is presented as well as the distinction between single and multi-label problems in text classification and its implications. After we explore how the transformation into feature vector of documents is achieved also known as preprocessing methods.

2.1.1 Defining text classification problems

Text classification consists of assigning $\{1, 0\}$ to each pair $\langle d_j, c_i \rangle \in D \times C$, where D is a text collection and $C = [c_1, c_2, ..., c_p]$ is a set of p predefined *labels*. If 1 is assigned to $\langle d_j, c_i \rangle$, it reflects the decision that d_j belongs to the category c_i, and 0 is assigned otherwise. Formally, this task can be defined as a way to *approximate the unknown target function* $\hat{\Phi} : D \times C \rightarrow \{1, 0\}$ *(that describes how the documents ought to be classified) by means of a function* $\Phi : D \times C \rightarrow \{1, 0\}$ *called the classifier (rule or hypothesis or model) such that* $\hat{\Phi}$ *and* Φ *coincide as much as possible* (Sebastiani, 2002).

During this training phase, the *classifier* learns from n documents (the *training* sample) that are already arranged into p separate folders, where each folder corresponds to one class (Namburu et al., 2005; Hotho et al., 2005). In the so called *hard version* of the classification problem, one label will be assigned to the new instances, whereas in the *soft version*, a probability to the example belonging to the category is computed (Aggarwal and Zhai, 2012).

Single and multi-label classification problems

An important distinction for types of TC tasks is one that involves how many classes are assigned to a single text document. This is known as the difference between *single-* and *multi-class* categorization of texts. In single-class problems, each text is assigned only one class, whereas in multi-class classification each example can be filed under any number of categories (Feldman and Sanger, 2006). Another way to distinguish these tasks is by considering *multi-class* categorization as problems with *overlapping categories*, whereas *single-class* have *non-overlapping categories*. A particular case of *single-label* problems is the *binary* case. In these problems, the text documents can only be assigned two possible classes and each document is singularly assigned to the one class and one class only (Sebastiani, 2002; Feldman

and Sanger, 2006). This can be translated in saying that $d_j \in D$ can either be assigned to c_i or to its complementary $\bar{c}_i$. In reality, all TC problems can be translated into a *binary* problem, by transforming the decision into *does* d_j *belong to category* c_i *or not?*, which works not only for single-class problems with more than two classes but also for multi-classes ones. A simple transformation makes it possible for algorithms designed for binary categorization to be used for multi-class classification, while algorithms for multi-class cannot be used for either the binary or the single-class TC (Sebastiani, 2002).

Formally, in order to transform a multi-label (or a single-label with more than two categories) into a binary problem, one can divide the original problem into p (number of categories) *independent* tasks — it is assumed normally that labels are stochastically independent between each other, meaning that *for any* $c_k, c_r, \forall k, r \in C, k \neq r$*, the value of* $\hat{\Phi}(d_j, c_k)$ *is independent of* $\hat{\Phi}(d_j, c_r)$ (Sebastiani, 2002). It is expected that problems that have to undergo this transformation achieve worse results, which comes directly from the fact that the original binary case is the simplest version of the classification problem, while multi-label or single-label problems with more than two categories are indeed more complex problems.

2.1.2 Preprocessing for text classification

Adapting the classification problem to text is incredibly useful. However, considering the type of input for these tasks is text documents, a lot of processing is necessary in order to be able to feed it into data mining processes. This results from the fact that these processes only *understand* highly structured data, which usually means a spreadsheet format that sustains predictive modelling. This model of representation of data is built with columns that correspond to *features* or *variables* and each row is a new instance in the dataset. The cells are filed with values that characterize the instances according to the variables which have meaning as

this is necessary for the data mining processes to make sense (Sebastiani, 2002).

There are many options and paths one can follow in order to achieve this same result. Starting with representation options, in which one decides both what the features are going to be and what will fill the cells with. Normally the features are either the words in the document collection or some concepts derived from them. The cells are then filled with a form of frequency counts, that can be just simple counts, binary values to discern the presence of a word or some weighting schemes. These options and their relevance are going to be discussed first in this section (Berry and Castellanos, 2007).

The second problem that preprocessing of text tries to resolve is feature selection. If the representation chosen covers all the words that appear in the document collection, one can deduce that the number of features can be extremely large. This can become very dramatic when the text documents are for instance messages or social media posts since all the random misspellings of words would be considered features as well. Moreover, the resulting datasets would be very sparse, which can potentially be exacerbated if the average size of the documents is small. Consider the example of *tweets*, that have a maximum of 140 characters and one outcome of this is that people end up using many abbreviations and technically misspelling of words. The consequence would be a feature space of all possible words in a language plus all the misspellings/abbreviations/made-up words which frequency is recorded in texts of 140 characters. Whereas, if each example is a book from a collection of books, with normalized spelling and with a virtually unlimited amount of words per example, it is normal that there are more repetitions and the spreadsheets will not be so sparse (Jansen et al., 2009; Chen et al., 2011).

Thus, it is necessary to *clean* the documents before entering them into classification procedures and the techniques used for this purpose are presented summarily in the next sub-sections. We start by presenting options for representation of documents in the structured tabular format. Then we present in detail techniques used

specifically for dimension reduction in TC, such as *sparsity correction*, *stemming* and *stop-word removal*. We also present a typical preprocessing task for classification problems, information gain feature selection and then we give a short overview of other promising preprocessing techniques.

Representation

The need to choose a representation format comes from the fact that the input data for text classification is text documents. In a data mining perspective, this means the data is in an *unstructured* form of data since it does not abide to the spreadsheet structure. Since the data mining process requires the data to be in a *structured* format, text needs to be converted into tables (Feldman and Sanger, 2006).

The typical way a document is represented is through the so called *bag-of-words* also referred to as *vector space model* or even *document-term matrix*. In this representation, we have *unique terms* as features that characterize the text document. This entails that the ensemble of the features or single terms becomes the *dictionary* of the document collection (Weiss et al., 2010). Some studies have tried more complex constructions for the features, namely the use of *phrases*, which would have a greater syntactical value such as *N-Grams* (Cavnar et al., 1994). However the test results for these have not been encouraging which reinforces the use of terms instead (Weiss et al., 2010). Still in the document-term matrix, each row of the spreadsheet consists in a new example, which means that the number of rows is the sample size. The cells will then be filled by a statement about frequency of the term in the particular document.

Tokenization The first step consists of creating a framework for the machine to understand how to recognize a single word: when does it start and end. This means to recognize word delimiters. This process is known as *tokenization* for consisting of

teaching a machine to break streams of characters into *tokens* (Weiss et al., 2010). The intention is to permit the grouping of single terms. Usually and for the sake of simplicity, all `whitespaces` are considered delimiters. This concept includes `space`, `tab` and `new line`. It is defined this way, so that whenever a `whitespace` is detected, a new word is considered present. However, it is possible to point out already a big problem in this decision: multi-word expressions. Saying *white* and *house* should be different from saying *White House*, these expressions mean completely different things and treating them the same is incorrect. Another problem that occurs here are homograph and homonym words being treated as if they have the same meaning. Methods like *word sense disambiguation* are applied for solving these problems, even if their complete efficacy is not assured (Hotho et al., 2005).

When identifying word delimiters another important aspect it is the removal of punctuation and numbers. These are mostly considered not to be informative and in fact mark the end of words in conjunction with `whitespace`. The only exception to this are intra-word dashes, which are present in hyphenated words. For instance, the function `removePunctuation` from the `tm` package includes a very self explanatory logical option `preserve_intra_word_dashes` (Feinerer and Hornik, 2012). This may however create other errors, like variations in spelling of the same concept word, since the use of these hyphens are taken less strictly by most people (also spell-checkers do not recognize always this sort of mistake). However, since this is not a problem that affects many expressions, and unless a particularly important term for the document collection has a dash in between, there is less need to correct for it.

Frequency format At this point comes the decision about the weighting scheme to be used. This refers to the weight that is given to a token in a particular document which is derived from the term frequency, either just in that particular document or considering also its frequency in the whole document collection. The weight, w_{jm},

that are attributed to term t_m in document d_j, are set between 1 and 0. From here it can be concluded that for text representation, the datasets do not have negative values and also there are no missing values (Weiss et al., 2010).

The most common options for the term weighting schemes are:

– the *binary* case, in which only 1 is attributed as weights, if the token *appears* in the document, and zero if it doesn't,

– the *tf* case, in which the weighting comes from the frequency of the term in the document, and

– the most popular *tf-idf*, that is the term frequency modified by a scale factor for the importance of the word, which is called *inverse document frequency* (Weiss et al., 2010).

The *tf-idf* weighting scheme is described in the equation 2.1 (Salton and Buckley, 1988):

$$tfidf\left(t_m, d_j\right) = \#\left(t_m, d_j\right) \times \log\left(\frac{|D|}{\#D\left(t_m\right)}\right) \tag{2.1}$$

in which $\#\left(t_m, d_j\right)$ is the number of times t_m occurs in d_j, and $\#D\left(t_m\right)$ is the number of documents in D in which t_m occurs (Sebastiani, 2002). The logic behind this equation is that the more frequent a term is in a document, the more representative of that document it is, but that should be countered by how frequent that term is in the whole document collection. If the term is very frequent throughout, then it's much less relevant (Weiss et al., 2010; Sebastiani, 2002).

Advances in the area of representation were made in papers like Bloehdorn and Hotho (2004), in which the authors challenged the *bag-of-words* paradigm, integrating higher semantic level features and using *boosting* for the TC task. The results were more promising than past attempts. Also Cai and Hofmann (2003), made a very similar study, using a concept-based document representation to aid word- or phrase-based features. Their approach also achieved promising results.

Feature Selection

As expressed before, a spreadsheet representation of a text ends up having a very large number of features, making dimensionality reduction critical. Furthermore, many of those features are not informative which means they should be dropped as they are shown to create a bigger risk of *over-fitting* the classifying task (Sebastiani, 2002). This occurs when a model generated instead of learning to generalize from the trend, starts to memorize the training data. This will result in much worse results when the model tries to predict new instances.

Hence, there are a lot of techniques that enable filtering the terms that should be part of the dataset. Each technique aims at finding the features that are irrelevant for one reason or another. Next, we will present the feature selection or extraction techniques that are considered consistently as appropriate for the TC task.

Sparsity correction This technique enforces that for a term to be considered as a feature for the dataset, it needs to be present on a minimum percentage of documents across the document collection. The minimum threshold can be set by the user and their needs. In `R` this method can be applied through the `removeSparseTerms` function in the `tm` package (Feinerer and Hornik, 2012). According to the documentation of this package, if `sparse` is set for 0.98 or 98%, we are removing the words (features) that are present in only 2% or less of the documents. So by performing *sparsity correction*, we are effectively filtering based on how rare the words are throughout the dataset.

This constitutes one of the first steps in feature selection, based on the grounds that extremely rare words are unlikely to be present to aid in future classifications (Forman, 2003). In Yang and Pedersen (1997), this form of dimension reduction achieved very competitive results in comparison with methods that use Information Gain and χ^2 test. Being one of the simplest and computationally cheap preprocessing tasks makes this method a reliable alternative when time is of the

essence. These results suggest a contradiction with an established Information Retrieval law that states that the terms with low-to-medium document frequency are the most informative ones (Sriram et al., 2010), as Sebastiani (2002) points out. However that is not the case here, since the number of features can be very large and each single document will have a small percentage of all the possible words in its collection. From this one can conclude that by taking out the words that appear at most 2% of the documents, we are not talking about the low-to-medium frequent terms, these are the extremely sparse terms (Sebastiani, 2002; Feldman and Sanger, 2006).

Stemming Stemming refers to reducing terms in the *bag-of-words* into their *stem* or root. This comes from the rationale that words which are drawn from the same base word, have the same *concept*, the same *semantic meaning*, so they can be merged together forming a single feature. This will reduce the words *look*, *looking*, *looked* and so on, into the single *concept*: *look*. This will effectively reduce the dimension of the possible words in a document collection and inflate the frequency of the *stem words* (Aggarwal and Zhai, 2012).

This task is done through the use of stemming algorithms (or *stemmers*) that are mostly developed in the context of Information Retrieval. There, stemming is used to raise the ability of a system to match a query and document vocabulary because it will reduce word variability (Xu and Croft, 1998). The most popular stemming algorithm for TC tasks is the Porter stemming algorithm (Porter, 1980) and is based on suffix removal. This type of *stemmer*, however, works without the aid of a dictionary, which makes it too aggressive at times and prone to errors. For instance, the words *general*, *generous* and *generic* are going to be conflated with the Porter algorithm, but words like *recognize* and *recognition* that actually should be merged since they are related, will remain separate (Cardoso-Cachopo, 2007). Actually, some experimental results seem to suggest that stemming may not be too

beneficial for text classification task and the reason for its continued use has to do with it's simplicity and the dimension reduction (Weiss et al., 2010; Sebastiani, 2002). Another variation of stemming has achieved better results than the Porter algorithm for information retrieval. This stemmer merges words based on corpus co-occurrence statistics, instead of relying only on morphological rules (Xu and Croft, 1998).

Some other variations to the normal usage of stemming algorithms can be pointed. *Inflectional stemming* is a much less aggressive kind of stemming in which terms are only merged if they are actually variations of the same word. Thus, only words that are grammatical variants will be conflated, like singular/plural and present/past. This type of stemming seems to make more sense for the English language, in which the irregular verbs for instance tend to make it impossible to cover them in a rule. Thus, this stemmer integrates a lexical database which makes sure that the words that end up together are merged correctly, which is an advantage that may compensate for the decreased ability to reduce dimension (Weiss et al., 2010).

Finally, *stemming to a root* is very aggressive in comparison to the other variants. In this case, not only the suffixes will be removed to get to the root but also the prefixes. In this variant, the word *uncomfortable* would be reduced to just *comfort*. The reasoning behind this type of stemmer is that by reducing the number of single tokens in a text collection, will make distributional statistics more reliable (Weiss et al., 2010).

Stop-word removal Stop-words removal or *stopping* refers to the act of discarding words that even though may be very frequent, are very frequent throughout the whole document collection and therefore have no information value. This stems from the fact that the aim in TC problems is to be able to *separate* or *distinguish* between categories. *Stop-words* are the most frequent words in texts for a given

language however, they are used as connectors in sentences and *Stopping* involves the words that are frequent for a language however, their use is related to semantics and grammar. These are the connecting terms in a sentence, like prepositions, subjects and adverbs, such as *the, she, a, which, that, because,...* These words allow us to connect ideas in sentences but besides that, they do not carry any information for the classification task, since they are frequent throughout the language as a whole (Hotho et al., 2005; Weiss et al., 2010).

There are words also that are monotonously frequent in a specific document collection or domain and therefore should be removed as well. For instance, when the document collection is of reviews of video games, words like *game, play* should be removed as well. These are considered domain specific stop-words.

Feature selection using information gain This method is commonly used for feature selection in any type of data mining tas tasks. It consists of filtering the features that only contribute up to a minimum threshold for *information gain*. This constitutes a way to attribute a *rating* to terms based on their *separating* ability. This *rating* is measured by how correlated a certain term t_m is with a particular class c_i. The logic behind this procedure is that by rating the features in terms of how useful they are for the classification task, one can just drop the ones that the algorithm deems not useful. The selection can be of a certain number of the highest rated terms or by establishing a threshold value for the score, below which the terms will be rejected (Guyon and Elisseeff, 2003).

This technique can be adapted to text classification, where we consider the terms as features and where the dimension is a great problem. With P_{c_i} as the global probability of label c_i, and $p_{c_i}(t_m)$ as the probability of class i, given that the document contains the term t_m. Let $F(t_m)$ be the fraction of the documents containing the token t_m. The information gain measure $Ig(t_m)$ for term t_m is given

by equation 2.2 (Aggarwal and Zhai, 2012):

$$\begin{aligned} Ig(t_m) = -\sum_{c_i=1}^{p} P_{c_i} \times \log(P_{c_i}) + F(t_m) \times \sum_{c_i=1}^{p} p_{c_i}(t_m) \\ \times \log(p_{c_i}(t_m)) + (1 - F(t_m)) \times \sum_{c_i=1}^{p} \times \log(1 - p_{c_i}(t_m)) \end{aligned} \tag{2.2}$$

In this way, the greater the value of information gain $Ig(t_m)$, the greater the discriminatory power of the token t_m.

Other preprocessing methods Other preprocessing methods for text classification include *word sense disambiguation* (*WSD*) techniques, that use the semantic properties of the words, i. e. their meaning, to create more appropriate features. Works like of Bloehdorn and Hotho (2004) and Cai and Hofmann (2003) hint on the fact that promising results can be achieved for text classification when using phrases or concepts without undermining the statistical qualities of the models. Also, in Forman (2003), a novel feature selection criteria was introduced, the Bi-Normal Separation and in that study achieved the better performance than other measures in most situations, specially when there is skewness in the class distribution.

Another approach that holds a lot of potential is *topic modelling* (Papadimitriou et al., 1998). This consists of an adaptation of the clustering analysis on to topics in text classification tasks. The logic behind these models is that documents cover a small number of topics and that topics use a small number of words (Blei et al., 2003). It consists of a way to find the topics occurring in a collection of documents. Thus constitutes in itself an unsupervised learning task, quite similar to document clustering. An adaptation of this tasks to supervised classification would be that topic model learns topics within the classes, as in Mcauliffe and Blei (2008).

2.2 Metalearning

Metalearning arises from the need to exploit knowledge gathered through past experience to make learning systems more efficient (Brazdil et al., 2008). It is the study of machine learning tasks, contextualized as part of a learning system to find better ways to solve machine learning problems. In this way, it consists in machine learning using itself to classify problems in the best way to fix them (Giraud-Carrier, 2008).

Base learning To understand metalearning and its importance it is necessary to define the distinction between base (level) learning from meta (level) learning. Base learning consists in the typical inductive learning scenario: the application of machine learning methods and tools to a set of data in order solve a particular problem. The scope of this application is usually limited to the specific task and data resulting in not much insight ever being transferred to other tasks or domains (Brazdil et al., 2008). This means the application of a typical learning system does not usually yield any benefit for application of learning systems in new data and/or other tasks. This is the point metalearning tries to give a response to.

Goal Metalearning concerns itself with two major aspects:

1. serve as a guideline for users to select the best models, contextualizing the problems by providing a mapping from tasks to learners (Bensusan et al., 2000), and
2. how to profit from the repetitive use of a predictive model over similar tasks – this area is called *learning to learn* and involves the search for patterns across tasks (Brazdil et al., 2008).

One goal of metalearning will be to learn what causes algorithm α to perform better than other algorithms in certain types of learning tasks. Two perspectives can be

taken to solve this problem: understand what properties these learning tasks share that make algorithm α more efficient at learning them, and/or understand what components of algorithm α contribute for the success in its performance on some learning tasks (Vilalta and Drissi, 2002).

Learning bias According to Rendell et al. (1987), a metalearning system should be *capable of learning through experience when different bias are appropriate for a particular problem*. Even though this idea may minimize the role of metaknowledge (Brazdil et al., 2008), it introduces the important aspect of bias selection that metalearning studies how to navigate dynamically.

Gordon and Desjardins (1995) offer a broad interpretation for learning bias as anything that *influences the definition or selection of inductive hypothesis*. The learning bias is at the core of learning algorithms and explains why algorithms perform differently on the same data — they are formulated to find specific kind of regularities (Vanschoren, 2010). For instance, a linear regression model (LRM) is designed to predict dependent variables through a linear relationship to its explanatory (*independent*) variables. If the relationship between these variables is not linear, the model will not be good at predicting future values for the independent variable. This does not mean, however, that the dependent variable is not explained by the independent ones, but just that the way LRM does its inductive leap to predict new instances is not appropriate for the data at hand.

The learning bias can be separated in two components: *representational* bias and *procedural* bias. On one hand, the representational (or declarative or language) bias is what defines the representation of the search space of hypothesis and affects the size of the search space (Brazdil et al., 2008; Gordon and Desjardins, 1995). This encompasses that each learning algorithm is optimized to adequately capture regularities inside of a search space defined by the model itself. The increasing number of observations allow for more refined instances but still limited

by the search space (Vanschoren, 2010). On the other hand, the procedural bias (also called algorithm bias) refers to how the order in which the inductive hypothesis are navigated through are defined by the algorithm (Gordon and Desjardins, 1995). Meaning, how the algorithm decides in its iterations which refinements to the hypothesis are better than the other and so how it should traverse in the search (Vanschoren, 2010).

Transfer knowledge Metalearning aims at transferring knowledge from one learning task to another, regardless of domain (Lemke et al., 2015). This is achieved by exploiting what some authors call *metaknowledge*. As the name suggests, metaknowledge is knowledge about knowledge (Gadomski, 1997). In the context of metalearning, it refers to analysis and interpretation of the data generated by applying machine learning techniques (Giraud-Carrier, 2008). This data, also called *metadata*, can have different forms but it is from it that metaknowledge is gained and then exploited in order to build a more effective way of searching through the space of alternatives (Brazdil et al., 2008). Metadata can include historical information about the performance of a set of algorithms on different tasks/data, datasets and metrics available to compute dataset similarity (*metafeatures*), information that can be useful to perform dynamic bias selection, functions and algorithms to get more information (*metamodels*) (Brazdil et al., 2008). For instance, metafeatures are computed from the dataset are measurable properties of the data that use as input the number of classes, the distribution of training examples for each class (how balanced the training set is), the measure of correlation between the features and target concept and average class entropy, etc. (Brazdil et al., 2008). The main goal of these features is to shed some light on the connection between the learning algorithms and the characteristics of the data (Brazdil et al., 2008). This way, we are able to recognize on which datasets, a specific learning algorithm should work better.

Definition In conclusion, a definition of metalearning can be offered as the science of discovering relationships in metadata, generating metaknowledge. This metaknowledge enlightens the connections between the metafeatures and both the learning bias and the empirical performance data. By enabling the discovery of such patterns, metalearning promotes a better understanding of the determinants of the behaviour of algorithms on different types of data and allows informed recommendations on how to solve new learning tasks (Vanschoren, 2010; Brazdil et al., 2008).

2.2.1 Algorithm selection problem

The No Free Lunch theorem for machine learning (also known as *conservation law of generalization performance*) affirms that if all possible data distributions are equally likely, any pair of learning algorithms will perform the same on average, or even that a gain in performance for one algorithm on one class of tasks will correspond to the same loss on another class of tasks (Schaffer, 1994; Wolpert, 2002). The consequence of these statements is that it is not possible to create a *universal learning algorithm* that would be the best at solving all learning problems and for that reason metalearning presents itself as a natural alternative.

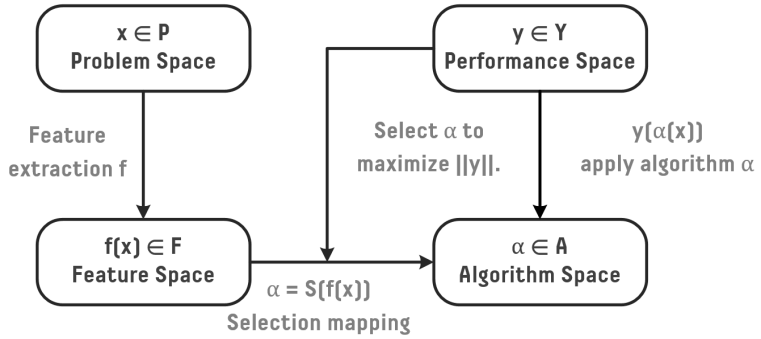

Figure 2.2: Schematic diagram of Rice's (Rice, 1976) algorithm selection problem framework. The objective is to determine the selection mapping S that returns the best algorithm α. Adapted from Smith-Miles (2009)

The framework for the algorithm selection problem was first put forward back in 1976 by John Rice when trying to answer the question: *With so many available algorithms, which one is likely to perform best on my problem and specific instance*? The first configuration of the model proposed in Rice (1976) is depicted in Figure 2.2 and has the following essential components:

- the problem space *P* represents the set of instances of a problem class;
- the feature space *F* contains measurable characteristics of the instances generated by a computational feature extraction process applied to *P*;
- the algorithm space *A* is the set of all considered algorithms for tackling the problem; and
- the performance space *Y* represents the mapping of each algorithm to a set of performance metrics.

This way, the algorithm selection problem can be formalized as:

For a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $y(\alpha(x)) \in Y$.

According to this model, choosing the features must obey a certain number of rules. Their choice must expose the varying complexities of the problem instances, must capture any known structural properties of the problems as well as any known advantages and limitations of the different algorithms must be related to these features (Rice (1976) as cited in Smith-Miles (2009)).

Vanschoren (2010) proposes an updated framework for the algorithm selection problem that expands upon the one proposed by Smith-Miles (2009) and its schema is shown in Figure 2.3. In Smith-Miles (2009) framework, the metadata obtained by the characterization of problems and evaluation of learning algorithms in those problems is then used to improve on existing algorithms but also build models of learning behaviour which is then directed to automatically recommend good

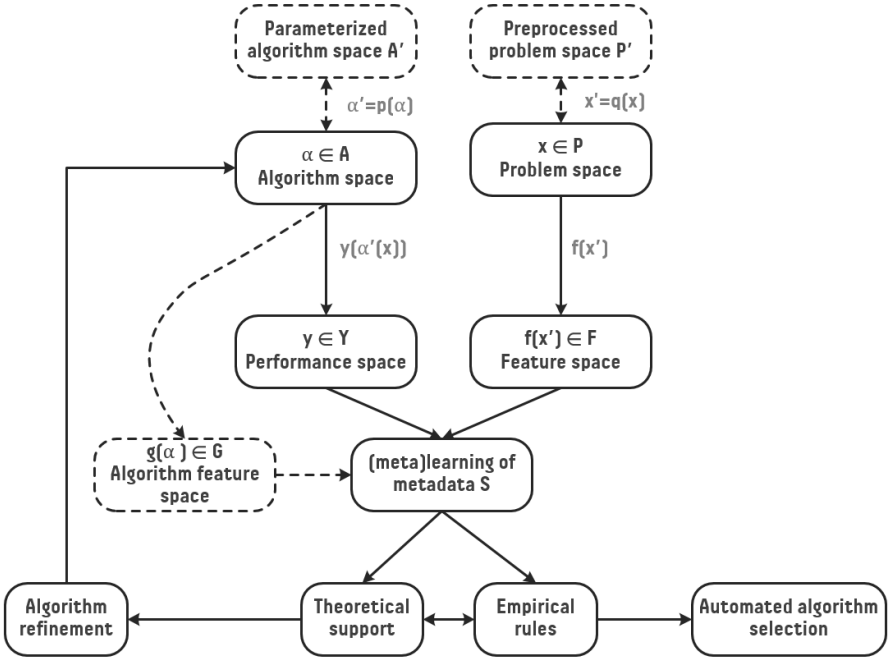

Figure 2.3: An updated framework proposed in Vanschoren (2010) in which the dashed lines are the extensions to the metalearning framework in Smith-Miles (2009) (here in full lines).

algorithms for a particular problem. This updated framework (Figure 2.3) adds dimensions to this problem, namely the parametrisation algorithms (A') which stems from the understanding that different parameter settings of the same algorithm make up for distinct models all together (Soares and Brazdil, 2006). The preprocessing problem space (P') also is added, from the idea that a dataset preprocessed in a certain way will result in different performances for the algorithms as well. And finally, the properties of the algorithms (G) are also considered as it is the aim of this model to be able to generalize about learning algorithms to find the patterns in the features of the algorithms.

An algorithm recommendation system must be able to support the users through the experimental phase of the KDP (*Knowledge Discovery Process*) in data mining. It achieves this by reducing the number of alternative learning algorithms that should be tried. For a given dataset, it selects the most likely models to achieve the best performance, saving the user time. To be able to accomplish this, it is imperative

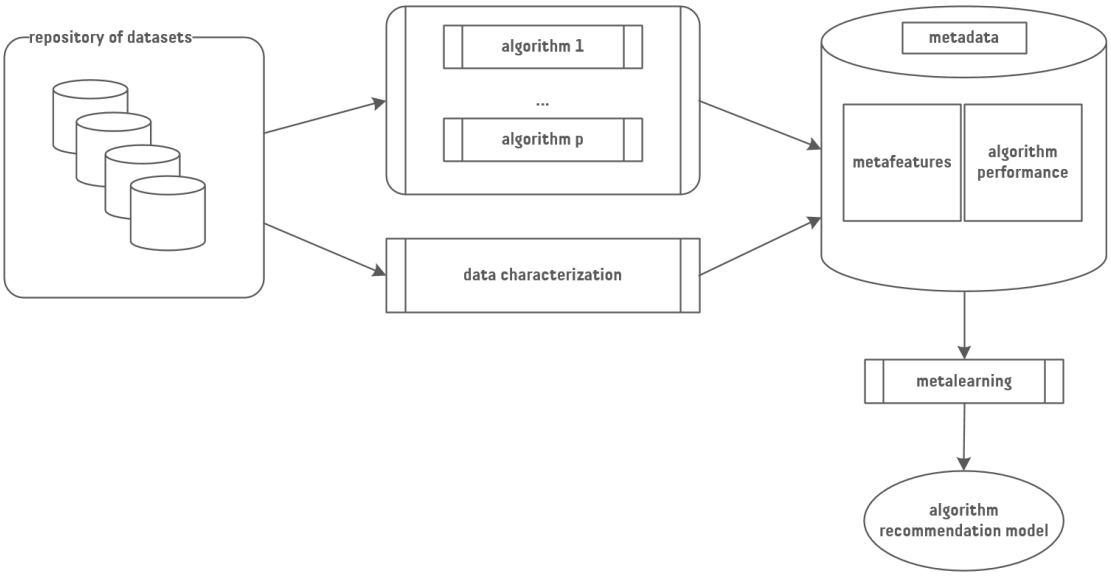

Figure 2.4: Metalearning to obtain metaknowledge for algorithm selection. Adapted from Brazdil et al. (2008)

that this system is able to accurately predict the algorithms *actual performance* as well as their *relative performance*, so it can return precise comparisons. Figure 2.4 represents this process as well as the inputs for such system as they were proposed by Brazdil et al. (2008).

2.2.2 Automated machine learning

Automated machine learning or *autoML* is a field in machine learning that has been gaining attention and that has several conceptual ties with metalearning. Even though not all the authors in this section mention this connection to metalearning in their works, the very implementation of these systems achieve precisely what is the goal and methodologies that fit the metalearning approach.

The goal of the several autoML systems is the same: to design and recommend optimized machine learning pipelines, algorithms and even adequate hyperparameters to specific learning tasks without reliance on user prior knowledge (de Sá et al., 2017; Olson et al., 2016). In this way, *autoML* seeks to make machine learning

tools more accessible by achieving an optimized solution without human intervention. However, autoML is faced with two major problems: first, as was mentioned in the previous section, there is no single algorithm that is best on all machine learning tasks and second, some algorithms performance is highly dependent on the hyperparameter settings used (e.g. non-linear support vector machines) (Feurer et al., 2015).

Auto-WEKA (Kotthoff et al., 2016; Thornton et al., 2013) and Auto-SKLearn (Feurer et al., 2015) are currently some of the most popular autoML systems, in part because of popular *software* in which they are hosted, Python – specifically in scikit-learn (Pedregosa et al., 2011) – in the case of Auto-SKLearn and obviously WEKA for the former. Since WEKA has a very intuitive and easy to use interface, it is favoured by novice users (Kotthoff et al., 2016) and python is currently becoming the most popular machine learning programming language. Both these autoML systems use hierarchical Bayesian method to perform a local search to explore the components of a task and then add some constraints to avoid invalid combinations (de Sá et al., 2017). Auto-WEKA considers all the algorithms available in WEKA and its possible hyperparameter combinations as the hypothesis space in which it navigates to find the most suitable solution: the combination (algorithm & hyperparameter(s)) that minimizes the cross validation loss of the learning task at hand (Kotthoff et al., 2016). Auto-SKLearn looks at 15 classifiers, 14 feature preprocessing methods and 4 data preprocessing methods which add up to 110 possible combinations for the machine learning pipelines that it will explore (Feurer et al., 2015). The Bayesian optimization method that both *Auto-WEKA Auto-SKLearn* use, fits a probabilistic model to capture the relationship between the hyperparameter settings and their measured performance. This model is then used to find the most promising hyperparameter settings, evaluates this configuration and iterates (Feurer et al., 2015). The way it chooses which are the *most promising* hyperparameter settings is by applying a metalearning approach of computing dataset similarity through the use of

metafeatures and evaluating historical performance data. This way, it is possible to connect the current problem with the datasets that are more similar to it and limit the current search space to the more likely settings to achieve better results. This approach may limit discovery by not exploring new areas of the solution space but gains in exploiting the known to be good regions (Feurer et al., 2015). Auto-SKLearn adds to this an automated ensemble construction step, which grants the possibility of using all the classifiers that were found by the Bayesian optimization (Feurer et al., 2015). In a later work, the same research team added neural networks to the mix by creating *Auto-Net*. This tool implements *automatically-tuned feed-forward* neural network clear of human intervention. This tool combined with Auto-SKLearn has shown better results than either one alone (Mendoza et al., 2016).

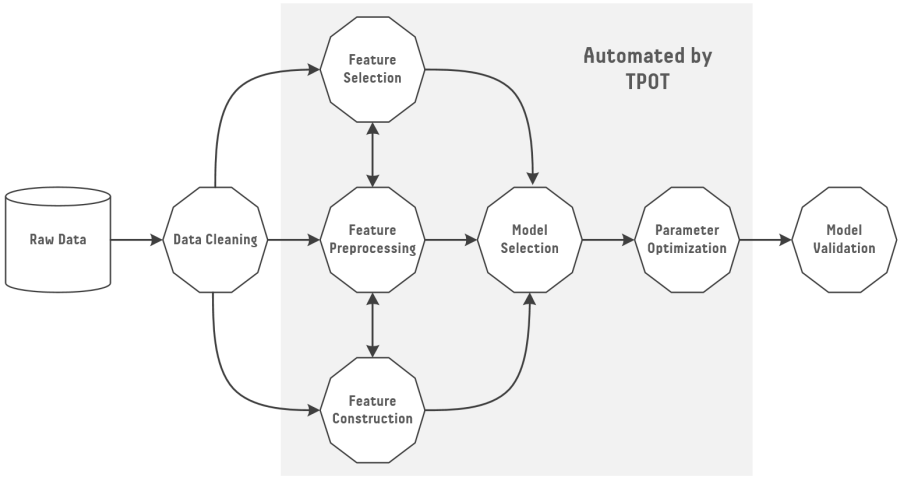

Figure 2.5: Representation of the typical supervised machine learning process and display of the phases of this process the automated machine learning system TPOT aims at automating. Most autoML systems automate these very parts as data cleaning still is, for the most part, too data and domain specific. Adapted from Olson et al. (2016)

Other recent approaches to the autoML problem have shown promising results. *TPOT* (Figure 2.5) is based on genetic programming to optimize a series of feature preprocessing settings and algorithms for classification pipelines. Furthermore, TPOT considers a Pareto selection for multi-objective search: on one hand,

to maximize the final accuracy attained with the model and on the other hand, to minimize the overall complexity of the model (de Sá et al., 2017). Contrary to Auto-SKLearn, where the search space is limited to the most likely to be successful regions, such limitations are not imposed on TPOT (Olson and Moore, 2016) . This way it favours discovery but also has as an disadvantage that it can waste resources trying solutions that are not actually possible (de Sá et al., 2017). *RECIPE* (de Sá et al., 2017) tries to give an answer this problem by working with grammar based genetic programming, which allows to guide the search inside ML pipelines that are similar to the ones that have previously shown successful results. This step permits evading infeasible pipelines and takes advantage of the random component from the genetic algorithm to enlarge the hypothesis search space.

Worth of mention as well is *autoBagging*, an autoML tool designed as a package for the R software which takes advantage of the latest advancements in the metalearning and a learning to rank approach to exploiting metadata (Pinto et al., 2017). And finally, the very recent but most promising *Hyperband* (Li et al., 2016). This method of iteratively tuning algorithms focused on optimizing the runtime of its iterations. Instead of trying several configurations at once, *hyperband* runs only a few iterations to start with in order to ascertain where the data at hand positions itself in terms of performance for those iterations. Then, it takes the best performing tries and runs only those further and iterates again (Zając, 2017). The authors of *hyperband* argue that methods that use Bayesian optimization (like *Auto-WEKA* and *Auto-SKLearn*), only improve on random search by a negligible margin. *Hyperband* is said to not waste resources on *bad* iterations and also takes advantage of random search propensity for discovery.

Chapter 3

Methodology

3.1 Overview of the problem

The study presented in this dissertation has the goal of facilitating workflow selection for classification tasks. We propose a method that provides a solution to the cold start problem in *KDP* (knowledge discovery process) – *which strategy to choose when many strategies are available?* Using a metalearning approach, our method capitalises on past learning experience in order to find the best workflows automatically. Although our study focuses on text mining tasks, the contribution of our method is that it can be applied to virtually any data mining problem.

It is well known that preprocessing strategies can significantly affect the performance of many classification algorithms. It appears that there are certain beneficial pairings between preprocessing methods and classification algorithms and our method is designed to uncover these relationships. This way, we suggest workflow selection, which includes preprocessing strategy (the combination of preprocessing methods) and algorithm selection and configuration (this may include hyperparameter refinement).

Furthermore, it is more useful to offer a list of recommended solutions instead of just one potentially *best* solution. It has been observed that there is no ulti-

mate solution that can fit every problem (*no free lunch theorem* (Giraud-Carrier and Provost, 2005; Schaffer, 1994)). This is even true for problems in the same domain, as the best solution is often not transferable across different tasks. This way, we suggest instead to recommend rankings of solutions – in our case workflows, that the user or the system could follow. This ranking of alternative solutions is constructed following different strategies described in this chapter. The past performance of different workflows plays an important role in the elaboration of the rankings.

We have two goals, on one hand we want to be able to recommend a ranking of workflows for a specific task. On the other hand we want to gain insight about the optimal amalgamations of workflow components that lead to good performance. Thus, we propose a general method for workflow selection problems that is characterized by three main phases. In the first phase, the past learning experience (*metadata*) must be identified and gathered taking into account the problem at hand. This is done either by compilation of the results obtained from other sources or by performing new experiments.

The second phase of our method involves elaboration of rankings. The past results gathered are transformed into rankings using two different measures, either accuracy rate or the A3R measure. The A3R measure combines the accuracy with the runtime taken by the whole process. We have adopted this measure as others have shown that it leads to good results (Abdulrahman, 2017). It is indeed important for a ranking to penalize solutions that take too long to achieve. This measure allows the user to impose their own preference for runtime.

These rankings are then used to determine one that achieved the best performance for a target dataset. This evaluation is performed with the recourse to mean loss-time curves in leave-one-out (LOO) cross-validation mode.

Finally, the resulting rankings of workflows are analysed in order to gain insight about the elements that are most important for achieving good performance. At this

stage, we propose to construct meta-model in the form of a linear regression to aid this analysis. This model relates the workflow elements and certain components of the workflows (e.g. feature selection) as independent or explanatory variables, to the output variable, accuracy rate. The following sections provide more details about the whole process.

3.2 Metadata collection

The first phase of our proposed methodology is defining the scope or extent of the study. Although the methodology can be applied to other domains, our primary focus is classification of text documents.

After deciding the scope of the study, it is then necessary to choose the workflows that will be included in the study. In our case, a workflow is defined by a certain preprocessing strategy to be applied (the combination of the preprocessing setups) and the classification algorithm (may include certain hyperparameter settings). This way, we need to consider which are the possible preprocessing methods that can be used and which settings should be considered. The same should be studied for the classification algorithms and if these require hyperparameter settings, what should be the actual values. These choices should be focused and justified since considering too many options unnecessarily will backlash and increase the number of possible workflows.

Consider the example shown in Table 3.1. There, only two preprocessing methods (stemming and stop-word removal) are considered with two settings each in conjuction with two algorithms (random forests and neural networks), one of which with two hyperparameter settings, will result in: $2 \times 2 \times 3 = 12$ workflows to consider.

Finally, the datasets should be chosen. Ideally, we would select datasets from many different sources in order to capture a representative subset of real-world

Table 3.1: Simple example of workflows construction.

	Preprocessing		
Worflow	Stemming	Stop-word removal	Algorithm
w1	Porter	Applied	random forest
w2	-	Applied	random forest
w3	Porter	-	random forest
w4	-	-	random forest
w5	Porter	Applied	neural network (h1)
w6	-	Applied	neural network (h1)
w7	Porter	-	neural network (h1)
w8	-	-	neural network (h1)
w9	Porter	Applied	neural network (h2)
w10	-	Applied	neural network (h2)
w11	Porter	-	neural network (h2)
w12	-	-	neural network (h2)

problems. However, this is not always very realistically attainable. The results of the study will depend on how well the datasets chosen capture the reality of the problem since the resulting rankings will translate the aspects present in the datasets considered.

Examples like *OpenML* (Vanschoren et al., 2013) make it possible to gather information about past performance of algorithms and even complete workflows. This database includes the results of many diverse experiments carried out by different researchers. However, since we consider *runtime* as an important variable when grading workflows, we did not trust the information available there, as consistent conditions may not have been ensured for the experiments. We have therefore carried out our own experiments. Also, by running our own experiments, we are able to choose exactly which workflows we wish to test. As suggested we combined several preprocessing methods specific to text classification such as *stemming* and *sparsity correction* with different classification algorithms like *random forest*, *neural networks* and *linear discriminant*. We gathered several document classification tasks from three document collections (*20 news groups* (Lang, 1995), *Reuters* (Lewis and Ringuette, 1994) and *ohsumed* (Hersh et al., 1994)), freely available and typically used for text classification experiments. These datasets were processed through the

workflows defined earlier and both accuracy rate and runtime data were collected.

3.3 Ranking methods for workflows

This section describes the ranking methods for workflows proposed for this study. First, we will present the measure used in conjunction with the ranking of the workflows, accuracy rate and the A3R measure Abdulrahman and Brazdil (2014). This is a measure that penalizes the accuracy rate for the time that a workflow needs to generate a result. Next we present both ranking techniques used *average ranking* and *active testing*. Average ranking Brazdil and Soares (2000) constitutes a simple method for ranking, where the ranking of workflows for each dataset is aggregated into a combined *average ranking*.

Active Testing Leite et al. (2012) is a more complex method, that needs an initial workflow to start a tournament, in which a series of *duels* between two different workflows takes place. This method selects the best competitor of the current best solution based on how likely it is that the competing workflow can outperform the current best workflow.

3.3.1 Performance measure used in ranking

Users normally prefer to obtain reasonably good solutions fast than solutions that may be slightly better but would take longer to achieve. This preference should be taken into account. For this reason, the accuracy rate alone does not seem to translate this preference well when creating a ranking of workflows. To respond to this problem, we follow the approach suggested in Abdulrahman and Brazdil (2014), who described a measure that uses both accuracy and runtime, *A3R*, that is used to compare workflows.

The original formulation of the *A3R* measure is unnecessarily complex for meth-

ods that do not use pairwise comparison, therefore here we opted for the simplified formulation proposed in Van Rijn et al. (2015). Let $A_{i,j}$ be the accuracy (success rate) of workflow *j* on dataset *i*, and $T_{i,j}$ be the runtime of the same workflow in this dataset, in seconds. As such, the *A3R* measure is defined in Equation 3.1.

$$A3R_{i,j} = \frac{A_{i,j}}{\left(T_{i,j}\right)^{P}} \tag{3.1}$$

The *P* parameter featured in this formula is basically the weighting parameter for the runtime in this measure. That is, as the parameter decreases, the less importance the *A3R* gives to runtime. If the value for *P* is defined as a fraction ($P = \frac{1}{r}, T^P = \sqrt[r]{T}$), then when *P* is equal to zero, $A3R_{i,j} = A_{i,j}$, that is, runtime has no effect on the overall outcome.

3.3.2 Elaborating average ranking

This method is inspired by the *Friedman's M Statistic* (Neave and Worthington, 1988) and its simplicity is one of its most attractive qualities. As the name suggests, this method consists in aggregating average of ranks the workflows achieved on the datasets.

After gathering the results obtained by each workflow, these are ordered for each dataset in terms of the chosen relative measure. In our case, it can be either accuracy alone or *A3R*. Then, all these ranks are collected for each workflow and aggregated by following the method described in Abdulrahman (2017). This list is then ordered according to the averaged ranks and the result is the average ranking of the workflows.

Table 3.2 displays an example of how an average ranking can be built for six workflows based on their results on three datasets.

As can be seen in our table, there is a tie for the first rank for dataset *d2* between workflows *w4* and *w6*. This problem is resolved in the usual way, by assigning to

Table 3.2: Example on how to construct average ranking of six workflows based on the results on three datasets.

	Datasets			Ranks				
Workflows	d1	d2	d3	d1	d2	d3	Avg. Rank	**Avg. Ranking**
w1	80%	70%	85%	4	4	5	4.33	**5**
w2	88%	74%	87%	3	3	4	3.33	**3**
w3	78%	66%	90%	5	5	1	3.67	**4**
w4	91%	79%	76%	2	1.5	6	3.17	**2**
w5	75%	60%	89%	6	6	2	4.67	**6**
w6	90%	79%	88%	1	1.5	3	1.83	**1**

each tied value the average of the ranks that would have been assigned without ties. This tie breaking solution is the same that is used for the *Friedman's M Statistic* and has little impact on the results. Furthermore, we can see that *w1* has the average rank of *4.33*. This is the result of the average of the ranks this workflow achieved on each dataset, as in $\frac{(4+4+5)}{3} = 4.33$.

3.3.3 Active testing

Active testing was introduced in Leite et al. (2012) and is a method to intelligently select the most promising model that should be tested *next* for a specific problem. By taking advantage of information of the models past performance in *similar* tasks, it is able to choose the *most powerful* adversary to compete with the best solution found so far. It selects this adversary by looking at a history of duels between the current best model and all other models, selecting the competitor which had better results on similar datasets.

The similarity between datasets is defined by the history of results from these duels, since it is assumed that a *similar* dataset is a dataset in which the same two algorithms have analogous outcomes in a *duel*. This way, the datasets are characterized by the performance difference between pairs of solutions that were run on them. It is assumed, for instance, that if *solution A* wins, ties or loses against *solution B* on *dataset 1* and *dataset 2*, the data distribution of these two datasets

is likely to be similar Leite et al. (2012) . This characterization of datasets works as a sort of *relative landmarking* for the solutions. By adding dataset similarity, this method is able to run through a very large set of possible fits for a problem relatively quickly, as it focus only on the most promising solutions.

This method returns an ordered list of workflows that should be followed and are likely to achieve the better results first – a ranking, however it needs a starting point, a starting workflow from which it can iterate. The method used by us Abdulrahman (2017) proposes that the results of average ranking should be used for this starting point, as it is an already good solution that can be obtained quickly. Given this first solution and the history of previous duels as relative landmarks, a tournament is started in which, for each round, the current best solution is compared to the next, most promising challenger. The winner becomes the new current best solution, the loser is disqualified. Each round contributes to further characterize the dataset and more accurately estimate similarity between datasets.

This method can only guarantee that the best solution for each dataset ($w^*_{d_j}$) will be the final winner of the tournament if all solutions are considered. For evaluation purposes, at the end of each round the difference between the performance of the winner and the performance of $w^*_{d_j}$ can be computed. This difference can be expressed in terms of *loss*. The goal will be then to minimize the loss while also minimizing the time it takes to reach the best result.

After running all workflows through all datasets and recording the performance achieved, either in terms of accuracy or *A3R*, the active testing is applied to each dataset in a leave-one-out cross-validation. This means that a tournament is held for each dataset and the performance it achieved with the workflows is kept out when selecting the competitors to ensure no data leakage. Each tournament is defined for each dataset, d_{new} and has four main stages:

1. Compute the average ranking for all datasets – leaving the dataset d_{new} out. Get the highest ranking workflow in average ranking, w_{best} and save the per-

formance achieved by w_{best} on d_{new}.

2. Compute the difference between the performance achieved by each workflow on each dataset with the corresponding performance of the w_{best}. Sum the positive differences for each workflow alternative and select the workflow with the highest sum. This is then the workflow that has historically won against w_{best} for the largest (relative) margin and therefore the most challenging competitor, $w_{competitor}$.
3. Get the performance of $w_{competitor}$ on d_{new} and compare with the performance of w_{best}. Select the winner and remove the losing workflow from the tournament. The winner will become the new w_{best}.
4. Repeat the process, now excluding the losing workflow, and stop once all the workflows have been considered or when the $w^*_{d_{new}}$ is found, whichever happens first.

Dataset similarity. Step 2 is where *relative landmarking* is established and can be further refined if a non-binary similarity variable is introduced. The original paper (Leite et al., 2012) proposed different methods for creating this variable and how to iteratively improve upon it. The most promising alternative presented in that study was a method in which the number of times two datasets agree on the duels winners is counted. This number is later corrected according to the Laplace method and becomes the similarity indicator. Our methodology, however, does not contemplate this dimension, assuming that all datasets are similar to one another, following the approach in Abdulrahman et al. (2017).

3.3.4 Evaluation of ranking methods

Loss-time curves

A good ranking of workflows is one that permits finding good solutions for the problem – low *loss* – in short amounts of time. We have decided to use loss-time curves to evaluate how good the rankings are at achieving this goal. These curves have the advantage of *showing* clearly the improvement obtained in the form of a step function. The shorter and steeper its steps are going down, the better is the ranking at finding the best workflow. These loss-*time* curves are an application of loss curves (first proposed in Abdulrahman et al. (2015)) where runtime evaluation is introduced to these curves. Instead of number of tests used in ordinary loss curves, the x-axis shows the runtime. Following a ranking means running the list of recommended workflows and each workflow takes time to execute. However, not all the solutions improve upon the current best. In the curve this is translated into a horizontal line that goes down vertically as soon as a better workflow has been identified.

Figure 3.1 displays an example of the representation of two loss-time curves corresponding to two different rankings for fictitious data. As we can see, *Ranking2* is a more efficient ranking since it finds better solutions sooner than *Ranking2* and so reaches a reasonable *loss* sooner. Although *Ranking1* ends up finding the best solution first, *Ranking2* performs a more efficient search than *Ranking1*.

Our figure shows the time represented on logarithmic scale. This was done on purpose in order to highlight the loss achieved at the beginning of the curve, corresponding to the initial tests. We assume users would be more interested in obtaining good recommendations as quickly as possible.

Default loss. The concept of *default accuracy* is the accuracy that would be obtained for the dataset if we labelled all instances as the majority class. This is often

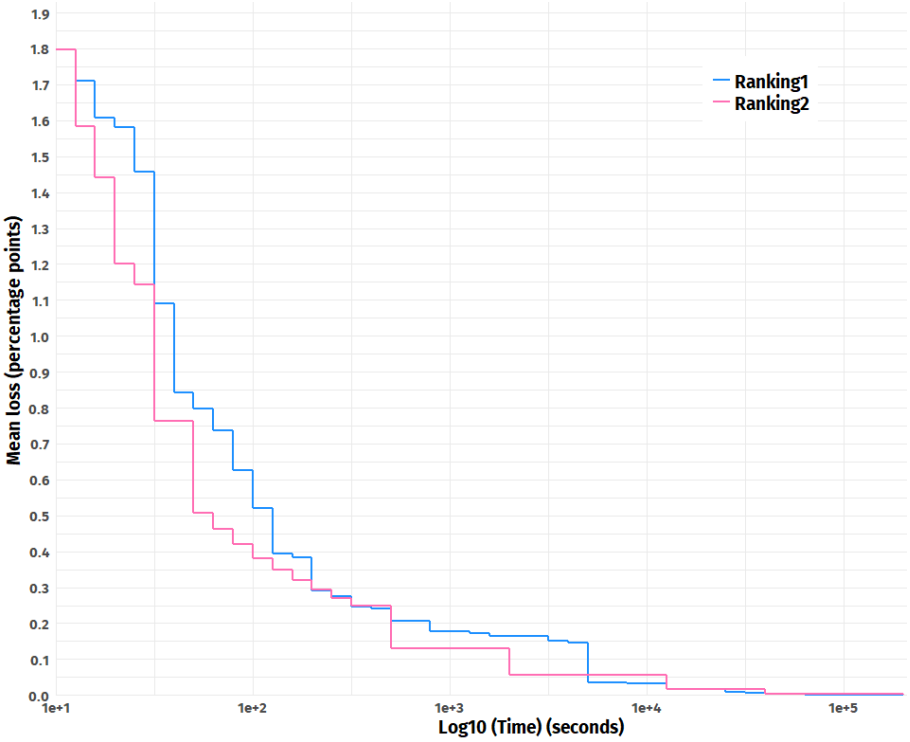

Figure 3.1: Loss-time curve example for two ranking methods.

used as the absolute baseline method[1] and is the accuracy achieved at moment t_0. From this, we can compute the *default loss*, which is the difference between the best possible solution for the dataset (w^*) and its *default accuracy*. This is the loss that can be used to initiate each loss curve.

The loss-time curve will start at the default loss and as each workflow is executed, we obtain its performance. If the recommended workflow achieves better performance than the default loss, the loss-time curve will go down to the new loss point. This will only happen, however, when the workflow finishes its execution. The limit of the x-axis will be the time taken to execute the workflows. This way, the earlier this ranking is able to identify the best workflow, the better.

[1]This classification criteria is slightly better than random choice. Because it uses more information, it is possible to compute.

Mean interval loss A good way to characterize these curves is by computing the mean loss in a given time interval, which corresponds to the area below the loss-time curve (Abdulrahman, 2017). Even though users may have their specific preference for time, they probably have a time budget beyond which they are not interested in running more tests. This way, we focus on the selection process within a given interval. This concept allows us to compare different ranking methods by comparing the *mean interval loss* (*MIL*), which corresponds to the mean area in the interval below the loss-time curve. The smaller the *MIL* value is, the more efficient the ranking method is.

Mean loss-time curves

When considering a group of datasets, however, other steps need to be added when carrying out the evaluation. Following Abdulrahman (2017), we adopt the usage of *mean loss-time curves* for this aim, where all the individual loss-time curves for the datasets are aggregated into a single curve. This makes it possible to have an overview of how different ranking methods compare with each other across a group of datasets, which represents a more comprehensive analysis.

The aggregation of several loss-time curves into a single curve is not trivial. To do this we followed the method proposed by Abdulrahman et al. (2015), in which for every point in time, the loss on each data curve is retrieved then all the values are averaged.

3.4 Analysis of usefulness of workflow constituents

The last phase of our proposed methodology is focused on gaining insight about what makes a good workflow for the problem at hand. This step involves using *metaknowledge* gathered through the experimental phase. This is a type of knowledge that is useful beyond the single learning task, as it involves transfer across

domains. The details concerning this approach are given in the following section.

3.4.1 Meta-regression model

Our analysis of results follows a novel methodology. We propose the use of a meta-regression model, in which the output variable is the accuracy achieved and the explanatory variables are the components of the workflows and some features characterizing the datasets. In this study we have opted for linear regression model for the meta-regression. This model is not meant to be used as a prediction model for accuracy, but rather as a convenient tool for the analysis of the impact of the elements that define a workflow and the task.

The model is constructed by transforming components of the workflow into categorical variables. That is, each preprocessing method and algorithm will become a variable, and the *levels* for these variables constitute the settings of each workflow. This way, the output (accuracy) is associated with what defines the workflow. Furthermore, two variables are proposed for characterizing the dataset/task: number of classes and similarity between classes. Both these variables reflect the *complexity* of the task, which influences the performance achieved on those tasks.

Since categorical variables cannot be entered directly in a linear regression model, our explanatory variables need to be converted into numerical. If the categorical variables only have two possible *levels* (dichotomous), the encoding is immediate and the only aspect that has to be considered is that the encoding needs to be reflected on the interpretation of the regression (e.g. using 0/1 or -1/1). However, when a categorical variable has more than two levels, further transformations have to be made. The most common way to deal with these types of variables is to separate them into their levels.

We would invite the reader to consider the example on Table 3.3. There are shown examples that can define a workflow, with preprocessing methods (*stem-*

ming and *sparsity correction*) and the algorithm options. While both preprocessing methods have only two possible *levels* (dichotomous), the example contemplates three possible algorithms.

Table 3.3: Simplified example for meta-regression.

Stemming (*stem*)	Sparsity correction (*spar*)	Algorithm (*algo*)
setting A (0)	setting C (0)	setting E
setting B (1)	setting D (1)	setting F
		setting G

While *setting A* and *setting C* can be converted in 0 and *setting B* and *setting D* into 1, for the *algorithm* variable it is necessary to perform *dummy codification* as shown in Table 3.4.

Table 3.4: Example of dummy encoding of a three level categorical variable.

algorithm	algoE	algoF
setting E	1	0
setting F	0	1
setting G	0	0

This table shows that the variable *algorithm* is divided in two new variables, *algoE* and *algoF*. When *algorithm* is *setting E*, then variable *algoE* takes the value of 1 and variable *algoF* is 0. The contrary happens when *algorithm* is *setting F*. When *algorithm* is *setting G* however, both these *dummy* variables are equal to 0.

Performing this transformation produces an obvious correlation between the two new dummy variables (*algoE* and *algoF*) and also causes some variation in the results according to the order the variables are entered in the model. This is an aspect we do not consider too troubling since this model, as stated before, is not designed as a prediction model for accuracy and rather an analysis tool of this variable.

Consider $A_{i,j}$ the accuracy of *workflow j* on *dataset i*. We created variables that define both workflows as well as datasets. Let W be the ensemble of variables that

define workflows and Q the variables that describe the dataset. In this way, there can be as many variables for these ensembles as deemed necessary[2]. Then, each W_j is defined in terms of k w_p variables, one for each workflow setting, and that each Q_i is defined by m q_l variables, for each dataset characteristic. All of these variables take either the value 0 or 1 as they are all *dichotomous*.

The formal model equation will be given by the Expression 3.2:

$$A_{i,j} = \beta_0 + \beta_1 \times w_1 + ... + \beta_k \times w_k + \beta_{k+1} \times q_1 + ... + \beta_{k+m} \times q_m + \varepsilon_{i,j} \quad (3.2)$$

where,

- w_p is one variable that defines the workflows (out of k variables),
- q_l is one variable that defines the type of dataset (out of m possible variables) and
- ε_i refers to the regression error for $A_{i,j}$

Using the simple example provided in Table 3.3, in which dataset characterization variables are not considered and the necessary transformations discussed, the meta-regression for this data is given by Expression 3.3

$$A_{i,j} = \beta_0 + \beta_1 \times stem + \beta_2 \times spar + \beta_3 \times algoE + \beta_4 \times algoF + \varepsilon_{i,j} \quad (3.3)$$

The output of this *meta-regression* will consist in the estimations of the values for the model coefficients. The sign of each coefficient estimative is indicates which direction the corresponding variable level influences the accuracy (if $\hat{\beta}_4 > 0$, all else equal, the accuracy increases when algorithm G is used). A t-test is performed for each coefficient, and the corresponding p-value will tell if that particular variable level is statistically significant (for a 5% significance level, we can reject the null hypothesis that the corresponding β is equal to zero if $p - value < 0.05$). The output also includes an F-Statistic, which corresponds to a *overall significance* test,

[2]There are some workflow elements that will have more than one level, and as seen in Table 3.4, this will result in separating the workflow element into variables corresponding to each possible level minus one (*setting G* in the example).

and evaluates the null hypothesis that all coefficients are zero. The *coefficient of determination*, or *adjusted R-squared*[3] should also be noted, as it translated into the fraction of the variance of accuracy that is explained by the model.

Furthermore, we propose also to perform an ANOVA Chi-squared test, the output of which groups the impact of each regression variable when looking at the variance of the target variable,

3.5 Discussion

It is at the first phase of our proposed methodology that generalization can be introduced. It is by defining clearly which metadata to investigate that the scope and application of the study is also determined. This way, any classification problem can use our methodology for discovering the most appropriate workflows. Furthermore, by increasing the variety of domains in the metadata, at the limit considering all possible domains, this methodology can be applied to solve *any* classification problem. However, our methodology benefits from some specification of the classification problem for two reasons: first, we believe that even if we were able to use all the past performances of all classification problems, that data would not be useful for recommendation of workflow for a specific problem. The ranking of workflows would most likely fit better the most performed classification tasks, being very biased toward *popular* domains. Second, we believe it should be offered some liberty in this selection, so the user may decide how wide the scope of their study should be. This method can equally be applied to all classification tasks as well as for instance, credit scoring – both applications are valid and the insights gained could potentially benefit other domains.

Our study consisted of applying this methodology specifically to the text clas-

[3]We advise looking at the *adjusted R-squared* instead of *R-squared* since these models tend to have a lot of independent variables, which tend to increase the *R-squared* albeit in a artificial way. The *adjusted R-squared* is an effort to correct this artificial effect of adding explanatory variables.

sification problem. This type of problem is characterized by a special type of data and distribution – text documents. This results not only in algorithms performing very differently on this type of classification tasks but also, in needing special preprocessing methods.

Applying a workflow selection methodology *specifically* to these type of problems is new[4], as we are studying preprocessing methods that can only be applied to *text data* (e.g. stemming or stop-word removal).

Consider the example of the application of stemming method. The reason for *stemming* words is that words with the same root-word have the same meaning and therefore can be grouped in a single word. However, could it be that the use of these more *nuanced* versions of the root word is actually important for the task of classification of documents? Are there algorithms that would benefit from not performing stemming more than others? We were not able to find such a complete study of text classification workflows in which questions like these are clearly addressed. This served as a motivation for carrying out this study. Next chapter (Chapter 4) will explore the results we obtained in our study.

The second phase of our methodology consists of the application of the methodology proposed by (Abdulrahman, 2017) for algorithm selection problems on to workflow selection for text classification problems. This reformulation consisted of increasing the dimensions of the performance, as now we consider not just the algorithm but also the data treatment applied before running the classifier.

Finally, the last phase of the methodology consists of a tool for analysing the elements of workflows, which is not only informative but also very easy to use. The formulation of linear regression models, where a set of variables is connected to a target variable, computing how each variable influences it in a *ceteris paribus* scenario and performing a significance test consists of a very convenient tool for the

[4]The study performed for the Auto-SKLearn (Feurer et al., 2015) method does include text classification problems, however does not consider specific preprocessing methods for text classification. The author used datasets that are already had been transformed into a structured format.

analysis of the effect of the elements of workflows and dataset characteristics. This analysis can support decisions about which elements should be dropped and which should be further explored in future works and definitely ascertain *what* makes a good workflow.

Chapter 4

Experimental Validation and Results

This chapter describes the experimental set-up used for the experimental validation of the methodology presented in the previous chapter and the results obtained from the experiments. First, we present the experimental set-up.

4.1 Experimental set-up

The experiments were performed on the same computer (for the complete hardware specification refer to Table A.1) over the course of 15 days. The computer was kept *off-line* for this period and had only essential software running in the background. The experiments were done in two parts. First, all the preprocessing strategies were applied on the data resulting in 2,400 preprocessed datasets (50 datasets were built from the 3 document collections and preprocessed in 48 different ways) which were then ran through the 8 classification algorithms. At the end of this process, 19,200 data points were obtained that detailed the time taken to run the complete process (preprocessing strategy time plus algorithm) – runtime, and the predictive accuracy obtained.

4.1.1 Document collections

When preparing the datasets, the difficulty of the classification task was taken into account. We assumed that the classification task will be more complex when the labels are more similar (less distinguishable). We also presuppose that classifying into three classes is more difficult than a task which has only two classes. Table 4.1 shows all the datasets prepared for the experiments, identifying all the classes as well as the corresponding encoding and number of documents present. The last column, *Similar?*, is used to indicate the how closely related the classes of the dataset are. Every document present in the datasets is labelled with one class only (single-class problems).

As can be observed in Table 4.1, three document collections were used to create the datasets used for this study, namely, *20 news groups*, *reuters* and *ohsumed*. The *20 news groups* dataset is a collection of approximately 20,000 newsgroup documents[1], distributed between 20 categories almost evenly. It was downloaded from the website `http://qwone.com/~jason/20Newsgroups/` which consisted of an unaltered version of the documents, still in their email format. From this document collection we created 23 datasets with 2 or 3 classes and varying levels of similarity between them.

The second document collection used here was the *OHSUMED* collection of medical abstracts with the MeSH categories from the year 1991. This document collection was downloaded from `http://disi.unitn.it/moschitti/corpora.htm` and consisted of all 50,216 available abstracts of cardiovascular diseases which were divided in 23 categories (*types of diseases*). The distribution of the documents in categories is not as uniform as it was in the case of the *20 news groups* datasets, therefore we ensured a balanced class distribution when creating the datasets. Since all the documents are abstracts from medical journals, all classes are considered similar in

[1]These consist of messages and notes exchanged about a particular topic posted in something like an on-line bulletin board system. These were the precursors of on-line forums.

Table 4.1: Complete list of datasets, respective sources and class names.

code	source	class 1	class 2	class 3	no. docs	similar?
n1	20 Newsgroups	alt.atheism	comp.graphics		1772	No
n2		alt.atheism	soc.religion.christian		1795	Yes
n3		alt.atheism	soc.religion.christian	talk.religion.misc	2423	Yes
n4		comp.graphics	comp.os.ms-windows.misc		1939	Yes
n5		comp.sys.ibm.pc.hardware	comp.sys.mac.hardware		1945	Yes
n6		comp.sys.ibm.pc.hardware	comp.os.ms-windows.misc	comp.sys.mac.hardware	2911	Yes
n7		comp.windows.x	misc.forsale		1960	No
n8		rec.autos	sci.crypt		1980	No
n9		sci.electronics	comp.graphics		1957	Yes
n10		sci.med	talk.politics.guns		1899	No
n11		talk.politics.guns	talk.politics.mideast	talk.politics.misc	2624	Yes
n12		talk.politics.guns	talk.politics.mideast		1849	Yes
n13		sci.space	rec.sport.baseball		1981	No
n14		rec.motorcycles	rec.autos		1985	Yes
n15		rec.autos	alt.atheism		1788	No
n16		talk.religion.misc	rec.sport.hockey		1627	No
n17		sci.crypt	misc.forsale		1966	No
n18		misc.forsale	rec.sport.baseball		1969	No
n19		comp.windows.x	alt.atheism	sci.med	1765	No
n20		sci.space	talk.religion.misc	misc.forsale	2590	No
n21		rec.sport.baseball	rec.sport.hockey		1993	Yes
n22		sci.electronics	sci.med	sci.space	2961	Yes
n23		misc.forsale	alt.atheism		1774	No
o1	ohsumed	neoplasms	cardiovascular		12429	Yes
o2		nervous_syst	immunologic		6967	Yes
o3		digestive_syst	disorders_environmental		5923	Yes
o4		bacterial_infec_mycoses	respiratory_tract		5129	Yes
o5		bacterial_infec_mycoses	urologic_male_genital		5058	Yes
o6		respiratory_tract	urologic_male_genital		5107	Yes
o7		urologic_male_genital	nutritional_metabolic		4437	Yes
o8		musculoskeletal	female_genital_pregnancy		3301	Yes
o9		female_genital_pregnancy	skin_connective_tissue		3240	Yes
o10		musculoskeletal	female_genital_pregnancy	skin_connective_tissue	4918	Yes
o11		musculoskeletal	skin_connective_tissue		3295	Yes
o12		virus	hemic_and_lymphatic		2448	Yes
o13		virus	neonatal_abnormalities		2257	Yes
o14		eye	neonatal_abnormalities		2084	Yes
o15		eye	endocrine		1863	Yes
o16		otorhinolaryngologic	endocrine		1580	Yes
o17		stomatognathic	animal		1032	Yes
o18		parasitic	stomatognathic		953	Yes
o19		parasitic	animal		933	Yes
o20		parasitic	stomatognathic	animal	1459	Yes
o21		digestive_syst	immunologic		6106	Yes
o22		virus	hemic_and_lymphatic	neonatal_abnormalities	3534	Yes
r1	reuters	earn	acq		6215	Yes
r2		crude	trade		700	No
r3		crude	trade	money-fx	993	No
r4		money-fx	interest		564	Yes
r5		trade	money-fx	interest	890	Yes

relation to the other. From this document collection, 22 datasets were constructed with 2 or 3 classes.

Finally, the *Reuters* dataset is a collection of documents that appeared in the

Reuters news-wire in 1987, which were manually assigned into categories by Reuters staff members. However, these documents were not just assigned to one category as a rule, which excluded many of them from this project. From the documents that were only *single labelled*, a sub-collection of the 8 more frequent labels was selected. This pre-selection of documents was downloaded from `http://ana.cachopo.org/datasets-for-single-label-text-categorization` (Cardoso-Cachopo, 2007) and included some transformations (e.g. lower case of all letters, replacement of new lines with single space, etc.). Due to the fact that this sub-collection is very small and the category distribution is very unbalanced, only 5 datasets were created from this source. Thus, this work used 50 datasets from 3 sources, 11 of which had 3 classes instead of 2, and most of these datasets had similar classes.

4.1.2 Workflows

For this study, 48 preprocessing strategies and 8 classification algorithms were considered forming an hypothesis space with 384 workflows. This section describes the settings studied.

Preprocessing methods

Five preprocessing methods were considered in our study, namely *representation*, *Stemming*, *Sparsity correction*, *Stop-word removal* and *Information gain feature selection*. For all of these methods except for *stop-word removal*, two options were studied. The methods for *representation* considered were *term frequency* (*freq*) and *term frequency-inverse document frequency* (*tf-idf*). The stemming options considered were either *no stemming* (*none*) or the simple Porter stemming algorithm (Porter, 1980) (*porter*).

Due to a limitation of the hardware used for the experiments (not enough random-access memory – RAM), it was necessary to perform some *sparsity correc-*

tion (99%) by default. This way, it was possible to reduce the number of features substantially and it allowed the learning algorithms to run. This could not be ensured when sparsity was not corrected since some datasets would have more than 20,000 features[2].

Three possible settings for *stop-word removal* were considered: *default*, *SMART* and *none*. *Default* stop-word removal refers to the standard 174 words removed by the `tm` package for the English language. The *SMART* setting refers to a much more extensive list (contains 571 words) composed by Chris Buckley and Gerard Salton at Cornell University freely available at `http://www.lextek.com/manuals/onix/stopwords2.html`.

Table 4.2 shows the different preprocessing methods included and corresponding options studied. Appendix B presents the full preprocessing strategies[3] that were used in the experiments and their code. Each dataset was preprocessed in the resulting 48 different ways[4], which culminated in 2,400 preprocessed datasets.

Table 4.2: Preprocessing methods options considered and correspondent codified term.

Method	Option 1	Code	Option 2	Code	Option 3	Code
Representation	Tf-idf	*tf-idf*	Frequency	*freq*		
Stemming	-	*none*	Porter Stemmer	*porter*		
Sparsity correction	At 99%	*0.99*	At 98%	*0.98*		
Stop-word removal	-	*none*	Default tm	*default*	Smart list	*smart*
Information gain FS	-	*none*	More than zero IG	*>0*		

Classification algorithms

Eight algorithms were selected for our experiments that were sufficiently diverse so as to make a representative selection of classifiers for documents. The implementation was conducted using the `caret` `R` package as it provides a consistent interface

[2]Terms.

[3]The combination of preprocessing options.

[4]The resulting preprocessed dataset would be the same every time, save for *information gain feature selection*, in which we ensured the same seed was set.

for all classifiers. The decision was made to use default values for all tuning parameters which were retrieved either from the original libraries or in some cases from WEKA software equivalent default values for these algorithms[5]. Finally, all workflows were run using 4-fold cross validation.

Table 4.3 shows all the classifiers considered, their code names, the original libraries and the default tuning parameters settings chosen for the experiments.

Table 4.3: Models considered. Reference to the R libraries used and default values of the tuning parameters.

Model	Code	Libraries	Tuning parameters
Linear Support Vector Machines	*lsvm*	e1071	cost = 1
Random Forest	*rf*	e1071, ranger	mtry = round down $\sqrt{no.features}$
Neural Networks	*nnet*	nnet	size = 1, decay = 0
C4.5-like Trees	*c4.5*	RWeka	C = 0.25, M = 2
k-Nearest Neighbours	*knn*	knn	k = 1
Rule-Based Classifier	*jrip*	RWeka	NumOpt = 2, NumFolds = 3, MinWeights = 2
Single C5.0 Tree	*c5.0*	C50	-
Linear Discriminant	*ld*	MASS	-

4.2 Exploratory metadata analysis

A short summary of descriptive statistics of the results helps to draw attention to some particular patterns that a ranked analysis cannot do. For this reason, the focus of this section is to provide descriptive statistics of accuracy and runtime by dataset, since the following section will be focused on the workflow distribution of results. This can help obtain understanding of the variance of performance in datasets.

Figure 4.1 refers to the boxplot of accuracy rate obtained per dataset. We can see which are the datasets that achieved the best results and also the ones which had the worst results. We consider the mean of accuracy as a *proxy* measure for the

[5]This was the case with the *neural network* and *k-nearest neighbours* algorithms. These classifiers had no default values set in their original libraries, so the default values of the correspondent algorithms in WEKA were used instead.

difficulty of the task, and this analysis allows us to understand if the assumptions made in the previous section are empirically held.

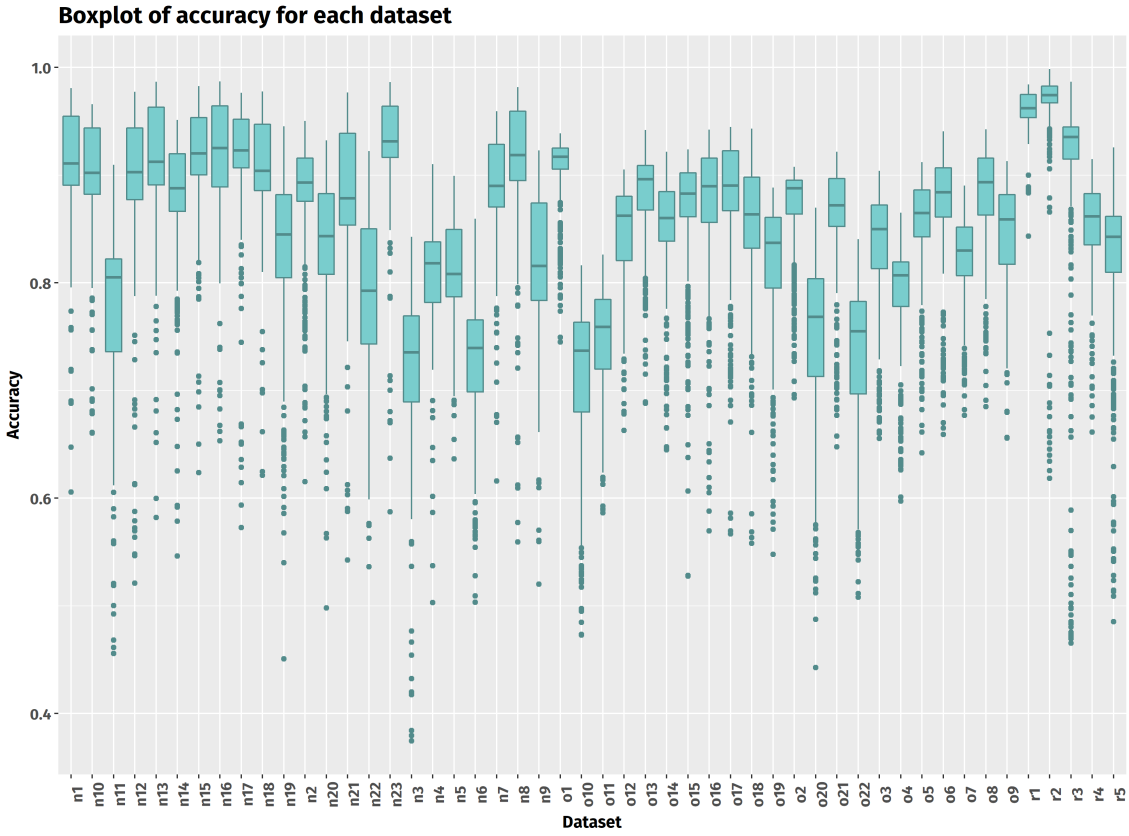

Figure 4.1: Boxplot of accuracy per dataset

The *reuters* dataset *r2* seems to have achieved almost perfect accuracy in one run (in fact Table C.1 shows the maximum result was 99.9%). On the other hand, the dataset *n3* has the worst result, with less than 40% of accuracy. However, the reason for this becomes clear upon consulting Table 4.1 and observing that not only is this a 3 class classification task but also every class covers the same topic, *religion*. Another trend that can be noted from the boxplot is that the tasks from the *ohsumed* dataset seem to be more difficult, which is apparent from the demonstrated tendency of lower values for accuracy for these tasks overall. This indicates more complex tasks, which is in line with the fact that all the datasets assembled

from this document collection have very similar classes (*types of diseases*). On the other hand, the distribution of mean accuracy seems to be more uniform for the *20 news groups* datasets, which in turn indicates that the *difficulty* of the tasks across this group is more uniform.

Figure 4.2 shows the boxplot of the time it took to run each complete workflow on the different datasets. It is possible to see that time variance matches the variance of the number of documents in the datasets, which makes sense since the more documents had to be processed, the longer the algorithm will take to build a classifier. Besides that, time seems to not vary greatly. Table C.2 in the Appendix shows the full set of descriptive statistics for runtime per dataset.

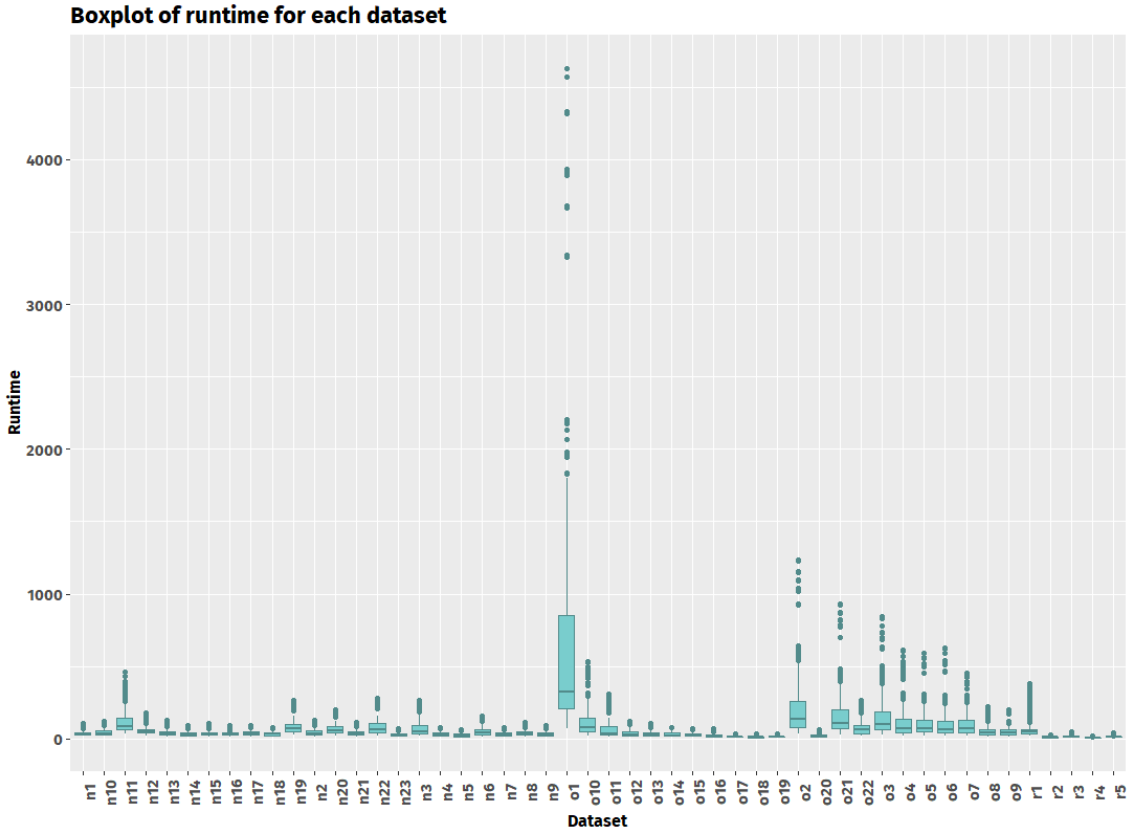

Figure 4.2: Boxplot of runtime per dataset

4.3 Ranking of the best workflows

This section describes the results of the average ranking and the active testing methods with our portfolio of workflows. In this study we use the text classification datasets described earlier. Our aim was to identify the best workflows and also to examine how much time is required to obtain a good solution with recourse to loss-time curves.

First, the final average ranking is presented, both with the A3R measure (AR-A3R) and using accuracy alone (AR-ACC). A section follows where the workflows are discussed in more detail, regarding specifically the relationship between the classifiers and preprocessing strategies that our results indicate. Then, the evaluation of the method in the form of mean loss-time curves is presented and discussed. In order to achieve the best possible results, a grid search was performed for the parameter P from the A3R equation for both average ranking (AR) and active testing (AT), even though this parameter had very little effect on AT as will be shown. This grid search is also featured in a subsection of Section 4.3.2.

4.3.1 Average ranking method

The average rankings constructed in Table 4.4 and 4.5 were done using the full set of data gathered in the experimental phase. Table 4.4 refers to the AR constructed with the A3R measure (AR-A3R) that takes not only accuracy rate, but also time into consideration. The importance of time is measured with parameter P which was set to $\frac{1}{40}$ as a result of the grid search optimization performed and detailed as part of section 4.3.2.

Table 4.4 shows the top 25 workflows include linear discriminant (ld) and random forest (rf), with rf being more common but not taking the first 3 ranks. In relation with the preprocessing methods, and in particular *representation*, both configurations *tf-idf* and *freq* are present. It seems that the *ld* benefits more from *tf-idf*

Table 4.4: Top 25 average ranking workflows considering all the datasets using A3R measure and $P = \frac{1}{40}$ (AR-A3R).

rank	w_id	repr	stop	stem	spar	info	algo
1	w94	tf-idf	default	none	0.98	>0	ld
2	w126	tf-idf	default	porter	0.98	>0	ld
3	w158	tf-idf	smart	none	0.98	>0	ld
4	w320	freq	default	porter	0.98	>0	rf
5	w62	tf-idf	none	porter	0.98	>0	ld
6	w288	freq	default	none	0.98	>0	rf
7	w312	freq	default	porter	0.98	none	rf
8	w30	tf-idf	none	none	0.98	>0	ld
9	w256	freq	none	porter	0.98	>0	rf
10	w128	tf-idf	default	porter	0.98	>0	rf
11	w224	freq	none	none	0.98	>0	rf
12	w190	tf-idf	smart	porter	0.98	>0	ld
13	w384	freq	smart	porter	0.98	>0	rf
14	w174	tf-idf	smart	porter	0.99	>0	ld
15	w110	tf-idf	default	porter	0.99	>0	ld
16	w64	tf-idf	none	porter	0.98	>0	rf
17	w376	freq	smart	porter	0.98	none	rf
18	w96	tf-idf	default	none	0.98	>0	rf
19	w296	freq	default	porter	0.99	none	rf
20	w352	freq	smart	none	0.98	>0	rf
21	w142	tf-idf	smart	none	0.99	>0	ld
22	w248	freq	none	porter	0.98	none	rf
23	w120	tf-idf	default	porter	0.98	none	rf
24	w32	tf-idf	none	none	0.98	>0	rf
25	w304	freq	default	porter	0.99	>0	rf

setting since all the workflows in the top 25, include this combination. Next, our results suggest that *stop-word removal* is better than *none*, with the default setting having some edge over *smart*. In regard to *stemming*, the top 25 seems to suggest a slight lead for the *Porter stemming algorithm*. Moving on to the last two preprocessing methods, *98%* setting on *sparsity correction* seems to greatly benefit the results, and feature selection with *information gain* is also beneficial.

The complete ranking of the workflows obtained with AR-A3R can be found in the Appendix, Table D.1. As a matter of interest, the workflow setting that achieved the worst rank was *w68*, in which the algorithm used is the *K-Nearest Neighbours* classifier, uses *tf-idf representation*, with *default stop-word removal*, *no stemming*

and *sparsity correction* done at *99%*.

Table 4.5: Top 25 average ranking workflows considering all the datasets using accuracy rate alone (AR-ACC).

rank	w_id	repr	stop	stem	spar	info	algo
1	w360	freq	smart	porter	0.99	none	rf
2	w168	tf-idf	smart	porter	0.99	none	rf
3	w296	freq	default	porter	0.99	none	rf
4	w104	tf-idf	default	porter	0.99	none	rf
5	w328	freq	smart	none	0.99	none	rf
6	w264	freq	default	none	0.99	none	rf
7	w232	freq	none	porter	0.99	none	rf
8	w136	tf-idf	smart	none	0.99	none	rf
9	w40	tf-idf	none	porter	0.99	none	rf
10	w72	tf-idf	default	none	0.99	none	rf
11	w176	tf-idf	smart	porter	0.99	>0	rf
12	w312	freq	default	porter	0.98	none	rf
13	w376	freq	smart	porter	0.98	none	rf
14	w112	tf-idf	default	porter	0.99	>0	rf
15	w184	tf-idf	smart	porter	0.98	none	rf
16	w368	freq	smart	porter	0.99	>0	rf
17	w304	freq	default	porter	0.99	>0	rf
18	w120	tf-idf	default	porter	0.98	none	rf
19	w200	freq	none	none	0.99	none	rf
20	w48	tf-idf	none	porter	0.99	>0	rf
21	w336	freq	smart	none	0.99	>0	rf
22	w272	freq	default	none	0.99	>0	rf
23	w174	tf-idf	smart	porter	0.99	>0	ld
24	w144	tf-idf	smart	none	0.99	>0	rf
25	w8	tf-idf	none	none	0.99	none	rf

The top 25 workflows ranked on accuracy alone (AR-ACC) tell a different story, however. Even though *ld* still appears in the 23rd position, this ranking is clearly dominated by the *rf* algorithm which occupied the remaining 24 ranks. Regarding the preprocessing methods, and in particular the *representation* format, a surprising result is that the very top rank is occupied by *freq*, however, both formats are featured in a similar way in this ranking. Regarding the *stop-word removal* method, again the top 25 features *default* and *smart* almost equally, with a slight lead to *smart* this time. The *Porter stemming algorithm* is also quite frequent, more so here than it was in the top 25 of AR-A3R. Finally, regarding *sparsity correction* and fea-

ture selection with *information gain*, surprisingly the very opposite happens with AR-ACC when compared with AR-A3R. Here the settings with *less* preprocessing both clearly win, with *99%* for *sparsity correction* and *no information gain* feature selection. It should be noted again that the difference between AR-A3R and AR-ACC is that the latter does not take runtime into account. Workflows which reduce the number of features will result in shortening of the runtime as well, which accounts for workflows that feature *more* preprocessing being favoured in the AR-A3R in the same measure as the importance that is given to runtime (the *P* paramenter). These results suggest that even though increasing *sparsity correction* and filtering features based on their *information gain* may decrease the runtime, it does not seem to benefit accuracy on its own.

The full ranking can be found also in the Appendix, on Table E.1. In the case of AR-ACC, the worst result comes with workflow *w212,* again with the *knn* algorithm and almost no preprocessing done, besides a *98%* setting on the *sparsity correction* and *representation* of the *frequency* format.

4.3.2 Evaluation of ranking methods

In order to construct a recommendation of workflows for document classification problems, both average ranking and active testing ranking methods were constructed. So far, we presented the average ranking results as the method allows for a ranked list of workflows to be presented. This is not the case with active testing, which chooses the best competitor for one starting workflow and performs duels iteratively. For this reason, the results of this method are only presented in this section, where we will discuss the quality of the recommendations of workflows it returns. Meaning, how fast the user reaches a reasonable amount of loss when following the ranking proposed by these methods.

This evaluation will be performed with the recourse to mean loss-time curves

and calculating the corresponding MIL (*mean interval loss*). These were assembled using a leave-one-out cross validation method. This method allows for an easy and effective way of assessing how effective the ranking of workflows is in identifying the potentially best alternative.

Optimization of the *P* parameter value

The original paper (Abdulrahman et al., 2017) that proposed the A3R measure for evaluating rankings of algorithms performed a grid search optimization in order to find the more beneficial values for the *P* parameter. Both AR-A3R and AT-A3R use the A3R measure, however the AT-A3R method needs a starting point for its iterative search. This means that the A3R measure used for the AR-A3R that jump-starts the AT-A3R method may have a different *P* parameter value from the one used for AT-A3R method. Since this is solely a starting point, it is not considered to be a problem albeit something that warrants explanation.

The grid search optimization of the MIL measure that was performed for the AR-A3R and the AT-A3R methods can be examined on Table 4.6. The *P* parameter seems to have a greater impact on the AR-A3R and just barely any on AT-A3R.

Table 4.6: MIL value for different settings of parameter P.

P (AR)	$^1/_5$	$^1/_{10}$	$^1/_{22}$	$^1/_{32}$	$^1/_{40}$	$^1/_{42}$	$^1/_{45}$	$^1/_{75}$
MIL	5.56	5.63	5.48	5.40	**5.31**	5.68	8.47	12.11
P (AT)	$^1/_1$	$^1/_{13}$	$^1/_{21}$	$^1/_{34}$	$^1/_{55}$	$^1/_{90}$	$^1/_{91}$	$^1/_{144}$
MIL	6.08	5.67	5.51	5.31	5.16	**5.1286**	5.13	5.13

Figure 4.3 reveals the impact of different *P* values on the AR-A3R performance. As it would be expected, if *P* decreases ($P \to 0$), the AR-A3R becomes very similar to the AR-ACC[6] and therefore the loss-time curves associated with each become the same. The best *P* parameter value found was $\frac{1}{40}$ and is depicted in this figure in blue.

[6]AR-ACC is equivalent to AR-A3R in which: $P = 0$

This setting is somewhat different from the one identified by the authors Abdulrahman et al. (2017) ($P = \frac{1}{64}$), however in that study the classification datasets were very different. Here, we are looking solely at text classification and this problem was not considered for that study.

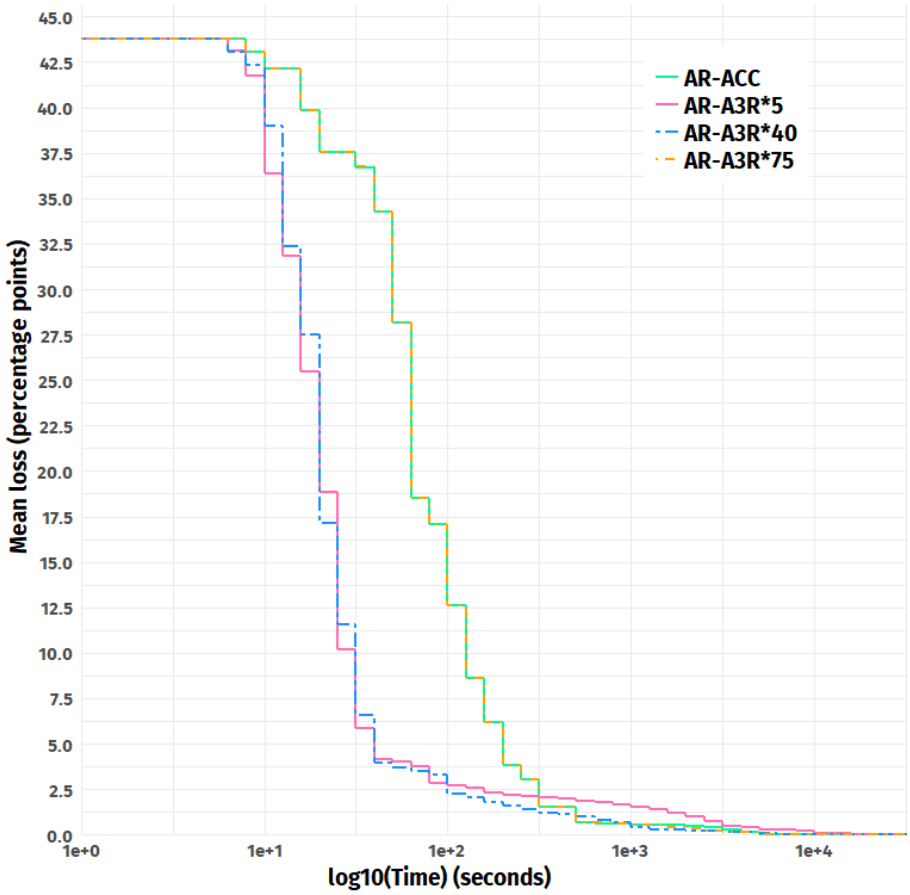

Figure 4.3: Mean loss-time curve study of AR-A3R parameter P. The alternatives considered are $P = \frac{1}{5}$, $P = \frac{1}{40}$ *and* $P = \frac{1}{75}$.

As was mentioned before, the *P* parameter value does not seem to have much impact for this data for the AT-A3R method as we can see in Figure 4.4. Especially when the *P* parameter becomes very small, the impact becomes negligible. When *P* is relatively high, which means that the A3R measure is giving more weight to runtime, the AT-A3R becomes less efficient at searching for the best workflows. The best value of the *P* parameter is in the end quite small ($\frac{1}{90}$) which leads us

to conclude that runtime does not have much impact on the performance of the AT-A3R ranking method.

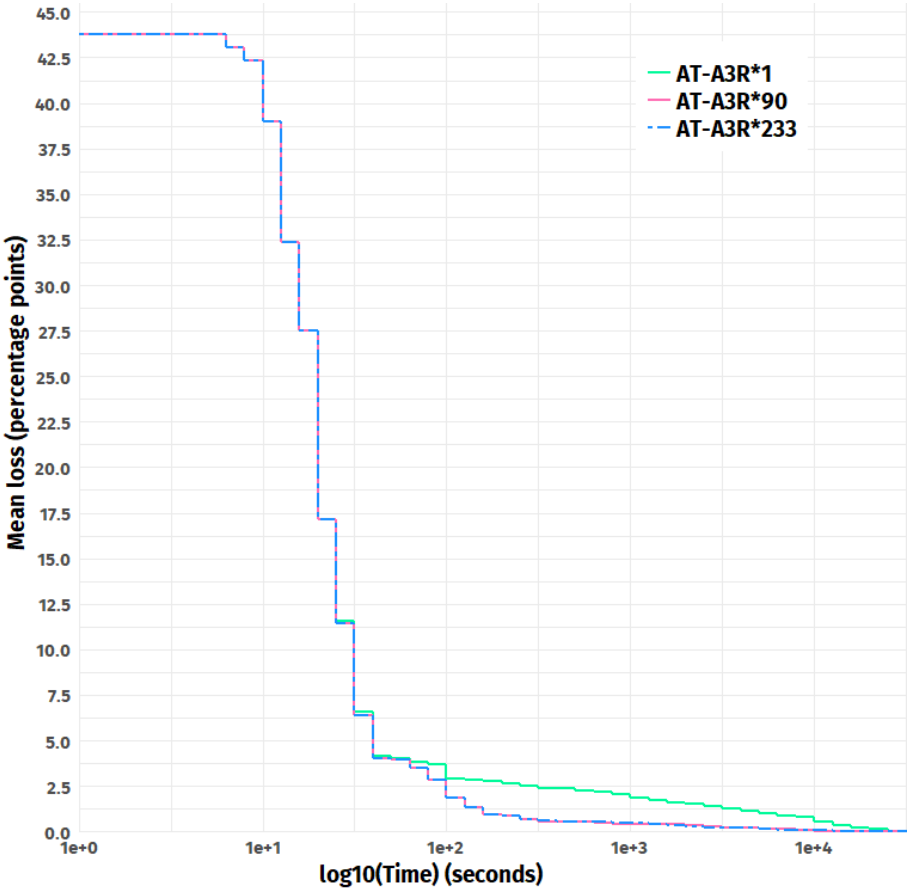

Figure 4.4: Mean loss-time curve study of AT-A3R parameter P. The alternatives considered are $P = 1$*,* $P = \frac{1}{90}$ *and* $P = \frac{1}{233}$*.*

Mean loss-time curves

Figure 4.5 presents the final mean loss-time curves for the average ranking method, AR-ACC and AR-A3R, and the active testing method, AT-A3R, applied to our data. First, this plot evidences the great gain in loss-time from using AR-A3R compared to the AR-ACC, which is quite expectable. As stated previously, this analysis evaluates the method based on how much time it takes the user to obtain a reasonable loss, since the AR-A3R considers runtime in its ordering of workflows, it will always get

an advantage against AR-ACC which solely considers accuracy rate.

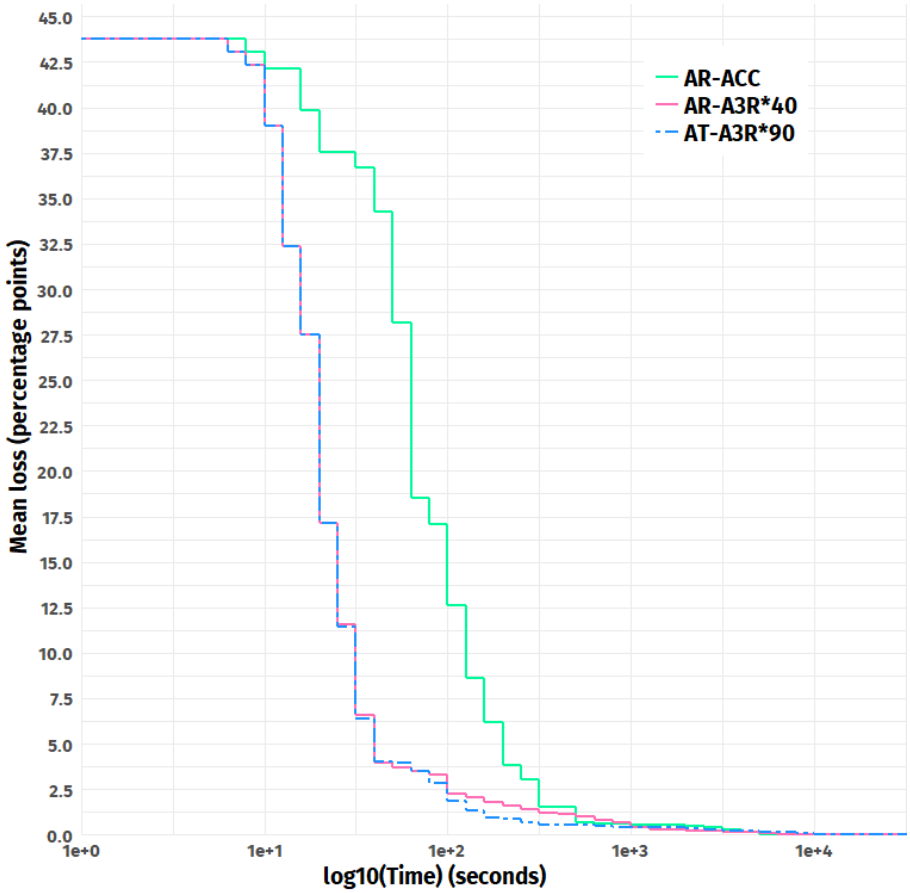

Figure 4.5: Mean loss-time curves for all ranking methods considered: AR-ACC (average ranking using accuracy alone), AR-A3R (average ranking using A3R measure and $P = {}^{1}/_{40}$) and AT-A3R (active testing using A3R measure and $P = {}^{1}/_{90}$).

Table 4.7: MIL measurements for the three rankings: AR-ACC, AR-A3R and AT-A3R.

	AR-ACC	AR-A3R	AT-A3R
MIL	12.136	5.305	**5.129**

Then, comparing just AR-A3R with AT-A3R, we can seen that the latter is slightly more efficient than the former. However, this win is only achieved in later points in time. For instance, at 100 seconds (10^2), the AR-A3R method is actually returning better workflows, for a mean loss of around 2.5pp. This is also identified in Table 4.7 where the MIL measure is better for the AT-A3R method, even if by a very small

difference. This indicated that even a very *plain* method like *average ranking*, with a very fast and simple implementation can be more beneficial. This, however, is also related to the fact that the version of *active testing* used is a simplified variant, in which similarity between datasets is not considered in the iterative search. It is our expectation that this method would benefit from taking similarity between datasets into account.

4.4 Identifying the important elements of the workflows

The first conclusion that can be taken from observing both average rankings depicted in this chapter is that from the learning algorithms that were featured in the experiments, two of them dominated both rankings, with *random forest* being the winner overall. This fact has supported our conviction that the right pairing of preprocessing methods and classifiers is more important than just considering the algorithm selection problem. This section is dedicated to understanding the components that make a good workflow for our data. First, we make a short analysis of the pairings that are more successful for each dataset, by analysing both AR-ACC and AR-A3R. Then, we end this chapter with a linear regression analysis which enables the statistical identification of the impact of the workflow components on the final accuracy achieved by the workflows.

4.4.1 Algorithms and preprocessing pairings

Table 4.8 orders all the algorithms featured in the workflows by their overall ranking position, and displays the highest rank each algorithm achieved in both AR-ACC and AR-A3R.

Observing the complete ranks (Tables E.1 and D.1 found in the Appendix) allows

Table 4.8: Algorithm and its highest rank in both AR-ACC and AR-A3R.

Algorithm	Highest rank AR-ACC	AR-A3R
Random Forest	1st	4th
Linear Discriminant	23rd	1st
Linear Support Vector Machines	50th	53rd
Neural Networks	83rd	80th
Single C5.0 Tree	126th	103rd
C4.5-like Trees	123rd	110th
Rule-Based Classifier	180th	173rd
k-Nearest Neighbours	257th	248th

us to notice which algorithms benefited the most from each preprocessing strategy and to realize the differences between AR-A3R and AR-ACC, which gives insight about the impact of runtime for the AR-A3R method. Looking at the distribution of the preprocessing strategies on the rankings for each algorithm, it is possible to conclude that *feature selection using information gain* is the preprocessing method with most impact across the algorithms. For almost all, the top half of their ranks are mostly characterized by the use of this method. This was so remarkably so in the case of linear discriminant *ld* and linear support vector machines *lsvm*, that the exact (24) top half ranks of both AR-ACC and AR-A3R had the *ig>0* setting. This has also happened in the case of the *neural networks* (*nnet*) but only for the AR-A3R, even though their AR-ACC also benefited greatly from this setting. However, for the *rf* algorithm this was not the case, and more specifically for AR-ACC, the best results were achieved when *ig* feature selection was not applied and this was also the same ranking where *rf* was the clear winner (see Table 4.5).

In fact, the *rf* presents the most atypical pattern in its settings for the ranking of preprocessing methods. *Freq representation* format wins for AR-A3R and the *99% sparsity correction* (our lowest *spar* setting), achieves the highest ranks for AR-ACC. Since this is also the winning algorithm, it leads us to believe that this classifier does not require as much preprocessing and feature selection and has the learning bias that better adapts to this data. We suspect that this is the result of *rf* having

a tuning parameter in which the default value takes the number of features into account (`mtry`).

Furthermore, it is possible to observe that the *ld* classifier, although definitely the second most successful algorithm, does not have a very uniform performance in comparison with *rf*. Indeed, its worst rank is 378^{th} for AR-ACC and 374^{th} for AR-A3R. In fact, a pattern is easily identifiable for the worst workflows with this algorithm: the less preprocessing performed, the worse the results. Finally, the exceptional ranks that these workflows achieve with the AR-A3R do not hold for the AR-ACC because even though this algorithm is remarkably fast, its accuracy rate can be topped by other less efficient algorithms.

The *lsvm* algorithm also seems to suffer drawbacks from using *freq* format of *representation* as most of the bottom ranks for this algorithm are characterized by this setting and *porter stemming* seems to benefit *single c5.0 tree* classifier. The remaining algorithms do not show a consistent preprocessing pairing pattern besides the aforementioned benefit from using *information gain* for feature selection.

4.4.2 Meta-Regression analysis

Our aim was to determine automatically how important different constituents of workflows are. We have decided to use regression analysis and corresponding ANOVA chi-squared test output for this goal, as this method is able to identify how significant each variable is. The conclusions are quite in line with the analysis so far, however a regression model offers statistical significance information and the quantifiable effects of the set-ups. This model is not intended to be used as a prediction tool for accuracy, it is only thought of as a tool for a better understanding of how some variables make accuracy vary.

To construct the linear regression model, the workflow options considered for the experiments were transformed into categorical variables (in `R`, factors). Besides

this, we have used two additional variables to describe the datasets: one describing the number of classes of the dataset and another characterizing the similarity of classes (see Table 4.1). In this way, the linear model has all the workflow *set-up variables* and *dataset descriptor variables* as independent or *explanatory* variables and *accuracy* as the dependent or *explained* variable. The formula that defines our meta-regression model is

$$\begin{aligned} Accuracy = \beta_0 + \beta_1 \times Algorithm + \beta_2 \times Repr + \beta_3 \times Stop + \beta_3 \times Stem \\ + \beta_4 \times No.Class + \beta_5 \times Similar + \beta_6 \times Spar + \beta_7 \times Info + \epsilon \end{aligned} \tag{4.1}$$

The regression model estimated from the data the output shown in Table 4.9[7].

Table 4.9: Output of linear regression model.

```
Call:
lm(formula = accuracy ~ algorithm + repr + stop + stem + no_class
    + similar + spar + info, data = reg_data)

Residuals:
      Min        1Q    Median        3Q       Max
-0.34658  -0.03018   0.03630   0.03727   0.18763

Coefficients:
                Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)    0.9382423   0.0017756  528.406    <2e-16  ***
algo_c4.5     -0.0034122   0.0017308   -1.971   0.04869  *
algo_jrip     -0.0173145   0.0017308  -10.004    <2e-16  ***
algo_knn      -0.0603282   0.0017308  -34.856    <2e-16  ***
algo_ld       -0.0386163   0.0017308  -22.311    <2e-16  ***
algo_nnet     -0.0067444   0.0017308   -3.897  9.79e-05  ***
algo_rf        0.0493420   0.0017308   28.508    <2e-16  ***
algo_lsvm      0.0170972   0.0017308    9.878    <2e-16  ***
repr_tf-idf    0.0094334   0.0008654   10.901    <2e-16  ***
stop_none     -0.0056571   0.0010599   -5.337  9.53e-08  ***
stop_smart     0.0031514   0.0010599    2.973   0.00295  **
stem_porter    0.0061706   0.0008654    7.130  1.04e-12  ***
no.class_3    -0.0893328   0.0010446  -85.519    <2e-16  ***
similar_1     -0.0685716   0.0009637  -71.152    <2e-16  ***
spar_0.99      0.0014016   0.0008654    1.620   0.10533
info_none     -0.0326097   0.0008654  -37.681    <2e-16  ***
—
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '·' 0.1 ' ' 1

Residual standard error: 0.05996 on 19184 degrees of freedom
Multiple R-squared: 0.5019, Adjusted R-squared: 0.5015
F-statistic: 1289 on 15 and 19184 DF, p-value: < 2.2e-16
```

[7]In this output the coefficients are defined as one setting of the workflows to make clear the *dummy encoding* performed of the explanatory variables. For instance, when variable *repr* equals to 0 it corresponds to *frequency* format, and when it is 1 it corresponds to the *tf-idf* format.

The F-statistic obtained shows a very low p-value, which allows us to reject the null hypothesis that all the regression coefficients are zero. Furthermore, almost all the coefficients are statistically significant with very low p-values for their t statistic[8] (in reality only the variable concerning sparsity is not statistically significant). It is interesting to observe the effect of the dependent variables by looking at the signal of the coefficients. For instance, it can be noted that algorithms *rf* and *lsvm* have a positive impact on accuracy, which goes in line with the position workflows with these algorithms have obtained in the general rank, and furthermore that the coefficient of *knn* has a negative sign. The workflows with this classifier obtained consistently bad relative results (Table E.1). Moreover, it is possible to note that preprocessing has a positive impact in accuracy. The coefficient of `stop_none` referring to the situation of not performing any stop-words removal is negative. It is suggested that applying *information gain feature selection* is particularly beneficial. Finally, the dataset described as being more complex (when the number of classes is 3 and the classes are similar to each other) not only has a negative impact on accuracy, but this variable is also accompanied by a larger (absolute value) coefficient - indicating that this factor has a significant effect on the performance.

The adjusted R squared statistic[9] or adjusted *coefficient of determination*, which is a measure of goodness of fit, is just over 50%. A more detailed analysis of variance in this model is presented in the ANOVA output table 4.10.

In this output it is possible to observe which variables have the most impact on the variance of the target variable, that is, accuracy. Specifically, the type of *algorithm* and the *information gain* have the most significant impact on the dependent variable. Only variable *spar* does not have statistical significance for the variance of accuracy. We would speculate that the reason is that the difference between the

[8]Since the t statistic is the result of the coefficients estimates, $\hat{\beta}_i$, divided by their standard errors, $\hat{\sigma}_i$, the p-value is the probability of obtaining a t statistic as high as if the null hypothesis was true, meaning if $\hat{\beta}_i = 0$.

[9]R-squared is the fraction by which the variance of the errors is less than the variance of the dependent variable.

Table 4.10: Output of ANOVA Chi-squared test of linear regression model.

```
> anova(linearReg, test = "Chisq")


Analysis of Variance Table

Response: accuracy

                Df  Sum Sq  Mean Sq  F value  Pr(>|t|)
algorithm        7  18.636   2.6623 740.5811  <2.2e-16 ***
repr             1   0.427   0.4271 118.8230  <2.2e-16 ***
stop             2   0.255   0.1275  35.4655 4.226e-16 ***
stem             1   0.183   0.1828  50.8409 1.037e-12 ***
no.class         1  26.671  26.6707 7419.1885  <2.2e-16 ***
similar          1  18.199  18.1990 5062.5650  <2.2e-16 ***
spar             1   0.009   0.0094  2.6231    0.1053
info             1   5.104   5.1043 1419.8926  <2.2e-16 ***
Residuals    19184  68.963   0.0036
—
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '·' 0.1 ' ' 1
```

two settings for *sparsity correction* is not significant enough to impact accuracy.

Chapter 5

Conclusions and future work

The study presented in this dissertation focused on text classification tasks. Text classification is a method increasingly used by many professionals. It also differs greatly from any other classification task because of the data that it uses, as it requires converting documents into tables representing a structured format. The preprocessing methods required for this conversion are mostly specific to text classification. Also, the performance of algorithms is specific on this type of tasks due to the sparse nature of the data.

For this dissertation we have conducted an empirical study of the performance and runtime of 384 workflows on 50 text classification tasks. The results of this study were analysed in detail and applied to the construction of three rankings of workflows using different ranking methods and according to a measure that combines accuracy and runtime, A3R. In addition, the data from the experiments was also used for the construction of a meta-regression model which related the constituents of the workflow and some dataset characteristics to the accuracy achieved.

We were able to conclude that the *random forest* algorithm was the best algorithm for the text classification tasks considered. The workflows that featured this algorithm had not only exceptionally good accuracy, but also very short runtimes. This is due to the use of a fast implementation of this algorithm, *ranger* (Wright

and Ziegler, 2017). This algorithm was present in most top ranks for any of our ranking methods. Regarding the effects of preprocessing we note that information gain feature selection, which benefited greatly the other algorithms, had a negative impact on workflows featuring the *random forest*.

The *linear discriminant* classification algorithm produced very efficient workflows in the A3R-based ranking of workflows. This leads us to conclude that this algorithm not only has a generally good performance, but is very fast at producing results. This fact brought workflows that featured this algorithm to the the top of the rankings.

Furthermore, we built a meta-regression model which was able to accurately identify the workflow elements which had the largest (and lowest) impact on accuracy. This analysis gave us informed clues as to where the highest potential is when it comes to constructing text classification workflows and also which settings should be used or abandoned in future work. This convenient tool produced conclusions that were in line with the metadata exploratory analysis performed.

The sections will feature a discussion of our results, with reference to the limitation of the study. In the last section we will present future work.

5.1 Limitation of this work

There are some aspects of our study that deserve some discussion. Some of them are potential limitations and in some other cases, we would like to state the reasons for some decisions we made.

Datasets variety. The decision of using only three sources for the text classification tasks considered was a difficult one. These sources are widely used in text classification in literature and are easily accessible which makes our study reproducible in the future. Furthermore, the use of these datasets allows for direct comparisons

of our study with other studies that used the same data. However, this is a very short list of sources and we are afraid that it is not representative enough of many real applications in text classification. The results could be somewhat different if the documents used came from social networks, particularly *Twitter*. This type of text is one of the most exciting applications of text classification nowadays and one of the most challenging, due to the short nature of *tweets*. Ultimately the decision fell on the reproducibility argument and on the fact that we were not able to find third-party sources in which *tweets* were consistently categorized into classes. This would represent extra work, while the time to carry out the experimental work is rather limited.

The algorithms. Our preference was to use a representative variety of classifiers, and we opted to use default values for their hyperparameters. This decision was made for the sake of comparability since hyperparameter tuning effectively changes the algorithm. For this reason some algorithms were clearly in a disadvantage, in particular the *neural networks*, which normally achieve good results when some tuning is carried out. Moreover, the rankings featured always the same two algorithms in the top positions. While this is certainly an indicator that these algorithms are very good fits for the text classification problem, it raises also the question of whether they had an unfair advantage. Specifically in the case of *random forests*. Even though the default settings were used, these settings introduced some adjustment to the specific task by considering the number of features. On one hand this is a sign of very intelligent algorithm design, when the default hyperparameters are able to adhere to the particular dataset. On the other hand, this could indicate that perhaps allowing some tuning of the hyperparameters based on the number of features may be a good strategy. Particularly in the case of *neural networks*, allowing for some tuning based on the number of the features in the task could useful.

Ranking methods used. Even though we are very satisfied with the fact that such a simple method like *average ranking* was able to achieve such good quality, other alternatives could be explored. This method is very biased towards the sample of datasets used for its creation. It would probably have faired worse, had the datasets been more diverse. Additionally, we would have liked to explore the potential of *Active Testing* further. However, the implementation of this method in its original set-up has revealed itself too complex to tackle for this study.

Meta-regression for analysis, not prediction. Finally, as was stated often throughout this dissertation, our meta-model is not meant to be used as a prediction tool for accuracy, rather of analysis of the elements of workflows. There are several reasons for this, the main one being that the construction of the model is not able to guarantee that the necessary assumptions for linear regression model to be used for prediction are met. It is for this reason that we do not correct for correlation when performing the *dummy encoding* explained in Chapter 3. We consider that the limitations of a predictor of this form would actually defeat its purpose, as in order to capture enough workflows to be useful at predicting accuracy, the significance of the model would be seriously impaired. As well as considering the data used to build the model, the independence between the data-points is unlikely, since the source document collections are only 3.

5.2 Future work

Based on the results and conclusions we presented, future work on this study could be focused on the following points.

- ***Drop workflow elements that performed worse***: Drop *sparsity correction* using 99% as the only setting and drop *smart stop-word removal*. From the eight

algorithms, only keep using *random forest*, *linear SVM*, *linear discriminant* and *neural networks*.

- ***Explore other preprocessing methods***: In particular topic modelling.
- ***Increase dataset variety***: Use different document collections and from more distinct sources, like *reddit* or *twitter*.
- ***Carry out some hyperparameter tuning***: Investigate tuning of hyperparameters particularly for *neural networks*. Consider a way to incorporate the number of features (or an indicator of it).
- ***Improvement of Active Testing***: Considering similarity between datasets.
- ***Comparison study to other ranking techniques and autoML methods***: Applying a metalearning approach to autoML has proven itself very useful for algorithm selection and we would like to test how our method would fare in comparison.

References

Abdulrahman, S. M. (2017). Improving algorithm selection methods using meta-learning by considering accuracy and run time. PdD Thesis, Faculdade de Ciencias do Porto, Universidade do Porto (Portugal).

Abdulrahman, S. M. and Brazdil, P. (2014). Measures for combining accuracy and time for meta-learning. In *Proceedings of the 2014 International Conference on Meta-learning and Algorithm Selection-Volume 1201*, pages 49–50. CEUR-WS. org.

Abdulrahman, S. M., Brazdil, P., Van Rijn, J. N., and Vanschoren, J. (2015). Algorithm selection via meta-learning and sample-based active testing. *MetaSel@ PKDD/ECML*, 1455:55–66.

Abdulrahman, S. M., Brazdil, P., van Rijn, J. N., and Vanschoren, J. (2017). Speeding up algorithm selection using average ranking and active testing by introducing runtime. *Machine learning*, Special Issue on Metalearning and Algorithm Selection.

Aggarwal, C. C. and Zhai, C. (2012). *Mining text data*. Springer Science & Business Media.

Agichtein, E., Castillo, C., Donato, D., Gionis, A., and Mishne, G. (2008). Finding high-quality content in social media. In *Proceedings of the 2008 international conference on web search and data mining*, pages 183–194. ACM.

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras, G., and Spyropoulos, C. D. (2000). An evaluation of naive bayesian anti-spam filtering. *arXiv preprint cs/0006013*.

Bensusan, H., Giraud-Carrier, C. G., and Kennedy, C. J. (2000). A higher-order approach to meta-learning. *ILP Work-in-progress reports*, 35.

Berry, M. W. and Castellanos, M. (2007). *Survey of Text Mining: Clustering, Classification, and Retrieval, Second Edition*. Springer.

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

Bloehdorn, S. and Hotho, A. (2004). Text classification by boosting weak learners based on terms and concepts. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 331–334. IEEE.

Brazdil, P., Carrier, C. G., Soares, C., and Vilalta, R. (2008). *Metalearning: Applications to data mining*. Springer Science & Business Media.

Brazdil, P. B. and Soares, C. (2000). A comparison of ranking methods for classification algorithm selection. In *European conference on machine learning*, pages 63–75. Springer.

Cai, L. and Hofmann, T. (2003). Text categorization by boosting automatically extracted concepts. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 182–189. ACM.

Cai, L. and Hofmann, T. (2004). Hierarchical document categorization with support vector machines. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 78–87. ACM.

Cardoso-Cachopo, A. (2007). Improving methods for single-label text categorization. PdD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa (Portugal).

Cavnar, W. B., Trenkle, J. M., et al. (1994). N-gram-based text categorization. *Ann Arbor MI*.

Chen, M., Jin, X., and Shen, D. (2011). Short text classification improved by learning multi-granularity topics. In *IJCAI*, pages 1776–1781.

de Sá, A. G., Pinto, W. J. G., Oliveira, L. O. V., and Pappa, G. L. (2017). Recipe: A grammar-based framework for automatically evolving classification pipelines. In *European Conference on Genetic Programming*, pages 246–261. Springer.

Feinerer, I. and Hornik, K. (2012). tm: Text mining package. *R package version 0.5-7.1*, 1(8).

Feldman, R. and Sanger, J. (2006). *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pages 2962–2970.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305.

Gadomski, A. M. (1997). An approach to meta-knowledge unified framework: a toga meta-theory perspective. `http://erg4146.casaccia.enea.it/Meta-know-1.htm`. Accessed: 2017-08-15.

Giraud-Carrier, C. (2008). Metalearning-a tutorial. In *Tutorial at the 2008 International Conference on Machine Learning and Applications, ICMLA*, pages 11–13.

Giraud-Carrier, C. and Provost, F. (2005). Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper. In *Proceedings of the ICML-2005 Workshop on Meta-learning*, pages 12–19.

Gordon, D. F. and Desjardins, M. (1995). Evaluation and selection of biases in machine learning. *Machine learning*, 20(1):5–22.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

Hersh, W., Buckley, C., Leone, T., and Hickam, D. (1994). Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *SIGIR 94*, pages 192–201. Springer.

Hotho, A., Nürnberger, A., and Paaß, G. (2005). A brief survey of text mining. In *Ldv Forum*, volume 20, pages 19–62.

Jansen, B. J., Zhang, M., Sobel, K., and Chowdury, A. (2009). Twitter power: Tweets as electronic word of mouth. *Journal of the Association for Information Science and Technology*, 60(11):2169–2188.

Joseph, S. (2002). Neurogrid: Semantically routing queries in peer-to-peer networks. *Web Engineering and Peer-to-Peer Computing*, pages 202–214.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. (1997). Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2016). Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, 17:1–5.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

Leite, R., Brazdil, P., and Vanschoren, J. (2012). Selecting classification algorithms with active testing. *Machine Learning and Data Mining in Pattern Recognition*, pages 117–131.

Lemke, C., Budka, M., and Gabrys, B. (2015). Metalearning: a survey of trends and technologies. *Artificial intelligence review*, 44(1):117–130.

Lewis, D. D. and Ringuette, M. (1994). A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*, volume 33, pages 81–93.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2016). Hyperband: Bandit-based configuration evaluation for hyperparameter optimization.

Mcauliffe, J. D. and Blei, D. M. (2008). Supervised topic models. In *Advances in neural information processing systems*, pages 121–128.

Mendoza, H., Klein, A., Feurer, M., Springenberg, J. T., and Hutter, F. (2016). Towards automatically-tuned neural networks. In *Workshop on Automatic Machine Learning*, pages 58–65.

Mitchell, T. M. (1997). Machine learning. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877.

Namburu, S. M., Tu, H., Luo, J., and Pattipati, K. R. (2005). Experiments on supervised learning algorithms for text categorization. In *Aerospace Conference, 2005 IEEE*, pages 1–8. IEEE.

Neave, H. R. and Worthington, P. L. (1988). *Distribution-free tests*. Unwin Hyman.

Olson, R. S., Bartley, N., Urbanowicz, R. J., and Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 485–492. ACM.

Olson, R. S. and Moore, J. H. (2016). Identifying and harnessing the building blocks of machine learning pipelines for sensible initialization of a data science automation tool. *arXiv preprint arXiv:1607.08878*.

Papadimitriou, C. H., Tamaki, H., Raghavan, P., and Vempala, S. (1998). Latent semantic indexing: A probabilistic analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 159–168. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Pinto, F., Cerqueira, V., Soares, C., and Mendes-Moreira, J. (2017). autobagging: Learning to rank bagging workflows with metalearning. *arXiv preprint arXiv:1706.09367*.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

Rehurek, R. (2014). Practical data science in python - jupyter notebook. `https://radimrehurek.com/data_science_python/`. Accessed: 2017-09-10.

Rendell, L. A., Sheshu, R., and Tcheng, D. K. (1987). Layered concept-learning and dynamically variable bias management. In *IJCAI*, pages 308–314.

Rice, J. R. (1976). The algorithm selection problem. *Advances in computers*, 15:65–118.

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

Schaffer, C. (1994). A conservation law for generalization performance. In *Proceedings of the 11th international conference on machine learning*, pages 259–265.

Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Smith-Miles, K. A. (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):6.

Soares, C. and Brazdil, P. B. (2006). Selecting parameters of svm using meta-learning and kernel matrix-based meta-features. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 564–568. ACM.

Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., and Demirbas, M. (2010). Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842. ACM.

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*, pages 847–855. ACM.

Van Rijn, J. N., Abdulrahman, S. M., Brazdil, P., and Vanschoren, J. (2015). Fast

algorithm selection using learning curves. In *International Symposium on Intelligent Data Analysis*, pages 298–309. Springer.

Vanschoren, J. (2010). Understanding machine learning performance with experiment databases. PhD Thesis, Katholieke Universiteit Leuven, Leuven (Belgium).

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2013). Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60.

Vilalta, R. and Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95.

Weiss, S. M., Indurkhya, N., and Zhang, T. (2010). *Fundamentals of predictive text mining*, volume 41. Springer.

Wolpert, D. H. (2002). The supervised learning no-free-lunch theorems. In *Soft Computing and Industry*, pages 25–42. Springer.

Wright, M. and Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in c++ and r. *Journal of Statistical Software, Articles*, 77(1).

Xu, J. and Croft, W. B. (1998). Corpus-based stemming using cooccurrence of word variants. *ACM Transactions on Information Systems (TOIS)*, 16(1):61–81.

Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420.

Zając, Z. (2017). Tuning hyperparams fast with hyperband. `http://fastml.com/tuning-hyperparams-fast-with-hyperband/`. Accessed: 2017-08-18.

Appendix A

Computer specifications

Table A.1: Specification of computer in which the experiments were run.

Desktop computer specifications	
Processor	Intel Core i5 3330 3.0 GHZ 6 Mb
RAM	8 GB
Graphic Card	Asus GTX 660
SSD	2.5" Samsung 850 Evo 500GB TLC SATA

Appendix B

Preprocessing strategies

Table B.1: Full set of preprocessing strategies considered for the experiments.

p_id	repr	stop	stem	spar	info	average no. terms
p1	td-idf	none	none	0.99	none	1613.2
p2	td-idf	none	none	0.99	>0	357.7
p3	td-idf	none	none	0.98	none	846.94
p4	td-idf	none	none	0.98	>0	267.76
p5	td-idf	none	porter	0.99	none	1457.44
p6	td-idf	none	porter	0.99	>0	305.02
p7	td-idf	none	porter	0.98	none	847.02
p8	td-idf	none	porter	0.98	>0	247.38
p9	td-idf	default	none	0.99	none	1519.54
p10	td-idf	default	none	0.99	>0	337.4
p11	td-idf	default	none	0.98	none	761.58
p12	td-idf	default	none	0.98	>0	242.16
p13	td-idf	default	porter	0.99	none	1371.56
p14	td-idf	default	porter	0.99	>0	287.96
p15	td-idf	default	porter	0.98	none	767.98
p16	td-idf	default	porter	0.98	>0	226.44
p17	td-idf	smart	none	0.99	none	1343.52
p18	td-idf	smart	none	0.99	>0	316.52
p19	td-idf	smart	none	0.98	none	630.64
p20	td-idf	smart	none	0.98	>0	215.44
p21	td-idf	smart	porter	0.99	none	1243.12
p22	td-idf	smart	porter	0.99	>0	271.58
p23	td-idf	smart	porter	0.98	none	659.92
p24	td-idf	smart	porter	0.98	>0	205.9

p_id	repr	stop	stem	spar	info	average no. terms
p25	freq	none	none	0.99	none	1613.2
p26	freq	none	none	0.99	>0	315.58
p27	freq	none	none	0.98	none	846.94
p28	freq	none	none	0.98	>0	237.2
p29	freq	none	porter	0.99	none	1457.44
p30	freq	none	porter	0.99	>0	267.72
p31	freq	none	porter	0.98	none	847.02
p32	freq	none	porter	0.98	>0	217.58
p33	freq	default	none	0.99	none	1519.54
p34	freq	default	none	0.99	>0	297.28
p35	freq	default	none	0.98	none	761.58
p36	freq	default	none	0.98	>0	214.92
p37	freq	default	porter	0.99	none	1371.56
p38	freq	default	porter	0.99	>0	252.06
p39	freq	default	porter	0.98	none	767.98
p40	freq	default	porter	0.98	>0	199.12
p41	freq	smart	none	0.99	none	1343.52
p42	freq	smart	none	0.99	>0	279.78
p43	freq	smart	none	0.98	none	630.64
p44	freq	smart	none	0.98	>0	192.78
p45	freq	smart	porter	0.99	none	1243.12
p46	freq	smart	porter	0.99	>0	238.26
p47	freq	smart	porter	0.98	none	659.92
p48	freq	smart	porter	0.98	>0	181.94

Appendix C

Summary statistics by dataset

C.1 Summary statistics of accuracy by dataset

Table C.1: Summary statistics of accuracy by dataset

Data	Mean	**Std. dev.**	**Min.**	**Max.**	**Median**
n1	0.913	0.051	0.606	0.981	0.911
n2	0.882	0.056	0.615	0.950	0.893
n3	**0.724**	0.073	0.375	0.843	0.735
n4	0.811	0.053	0.503	0.910	0.818
n5	0.812	0.046	0.637	0.899	0.808
n6	**0.731**	0.066	0.503	0.859	0.739
n7	0.890	0.050	0.616	0.959	0.890
n8	0.912	0.061	0.559	0.982	0.918
n9	0.820	0.061	0.520	0.923	0.816
n10	0.900	0.054	0.661	0.966	0.902
n11	0.783	0.080	0.456	0.910	0.805
n12	0.897	0.073	0.521	0.977	0.903
n13	0.915	0.059	0.582	0.987	0.912
n14	0.880	0.058	0.546	0.951	0.888
n15	**0.917**	0.049	0.624	0.983	0.920
n16	**0.918**	0.056	0.653	0.987	0.925
n17	0.914	0.059	0.573	0.977	0.923
n18	0.908	0.049	0.621	0.978	0.904
n19	0.833	0.078	0.451	0.945	0.845
n20	0.835	0.068	0.498	0.932	0.843

Continued on next page

Table C.1 – continued from previous page

Data	Mean	Std. dev.	Min.	Max.	Median
n21	0.882	0.065	0.542	0.977	0.879
n22	0.789	0.076	0.536	0.922	0.793
n23	**0.929**	0.051	0.587	0.986	0.931
o1	0.905	0.036	0.745	0.939	0.917
o2	0.868	0.044	0.693	0.908	0.888
o3	0.835	0.054	0.656	0.904	0.850
o4	0.793	0.051	0.597	0.865	0.807
o5	0.853	0.048	0.642	0.912	0.865
o6	0.874	0.052	0.659	0.941	0.884
o7	0.822	0.042	0.677	0.890	0.830
o8	0.883	0.048	0.685	0.942	0.893
o9	0.847	0.047	0.656	0.913	0.859
o10	**0.714**	0.075	0.473	0.816	0.737
o11	0.749	0.049	0.586	0.826	0.759
o12	0.846	0.048	0.663	0.905	0.862
o13	0.885	0.045	0.688	0.942	0.896
o14	0.851	0.051	0.645	0.922	0.860
o15	0.867	0.058	0.528	0.924	0.883
o16	0.875	0.063	0.570	0.942	0.890
o17	0.878	0.063	0.567	0.945	0.890
o18	0.855	0.059	0.558	0.943	0.864
o19	0.821	0.061	0.548	0.889	0.837
o20	**0.747**	0.083	0.443	0.870	0.768
o21	0.862	0.050	0.648	0.922	0.872
o22	**0.734**	0.074	0.508	0.840	0.755
r1	**0.962**	0.017	0.843	0.984	0.962
r2	**0.959**	0.067	0.619	0.999	0.974
r3	0.899	0.113	0.465	0.987	0.936
r4	0.853	0.041	0.661	0.915	0.862
r5	0.822	0.084	0.485	0.926	0.843

C.2 Summary statistics of time by dataset

Table C.2: Summary statistics of time by dataset

Data	Mean	Std. dev.	Min.	Max.	Median
n1	38.76	18.43	16.58	107.54	33.59
n2	44.54	25.05	16.09	130.91	35.89

Continued on next page

Table C.2 – continued from previous page

Data	Mean	Std. dev.	Min.	Max.	Median
n3	70.74	49.94	19.50	263.68	50.47
n4	31.87	15.56	11.35	79.79	26.98
n5	25.69	12.69	9.37	64.25	20.71
n6	49.67	30.22	14.48	159.97	39.30
n7	31.83	13.64	12.52	75.79	29.46
n8	43.23	21.24	16.38	115.78	37.18
n9	32.83	17.39	10.74	91.06	26.71
n10	46.28	24.08	18.16	124.03	38.12
n11	109.60	73.63	32.60	459.17	82.58
n12	60.14	31.95	23.57	181.71	49.68
n13	41.93	23.78	15.11	130.89	34.00
n14	32.67	17.77	10.71	91.67	27.98
n15	37.92	19.08	15.10	110.36	32.35
n16	35.60	16.53	15.44	93.69	31.29
n17	40.32	17.44	16.16	95.95	36.18
n18	33.38	14.39	13.26	78.78	31.02
n19	81.87	46.34	25.49	267.95	67.96
n20	67.67	37.27	21.17	204.20	57.15
n21	39.82	21.41	13.03	114.21	33.37
n22	79.97	54.32	21.02	278.38	62.47
n23	32.43	12.81	12.64	71.51	29.59
o1	654.04	784.12	69.18	4632.06	326.93
o2	203.66	206.72	31.42	1237.87	133.02
o3	151.09	144.97	25.56	842.28	97.72
o4	118.59	119.07	23.13	610.22	69.47
o5	105.70	98.46	22.66	590.94	70.23
o6	101.88	102.49	21.68	625.46	62.43
o7	98.76	80.35	21.23	453.15	72.44
o8	54.71	37.86	14.93	221.63	43.77
o9	51.52	34.85	14.94	203.05	39.57
o10	116.60	107.44	20.22	530.30	75.05
o11	61.34	54.33	13.88	311.94	37.87
o12	37.96	21.84	12.24	123.93	30.32
o13	35.06	18.69	11.80	104.19	28.59
o14	29.61	16.32	10.19	81.43	23.29
o15	26.06	13.40	9.93	70.37	20.92
o16	23.46	11.81	8.58	68.86	19.06
o17	14.00	5.70	6.41	33.98	12.53
o18	12.63	5.47	5.62	32.39	11.28

Continued on next page

Table C.2 – continued from previous page

Data	Mean	**Std. dev.**	**Min.**	**Max.**	**Median**
o19	**13.71**	5.77	6.46	34.45	12.21
o20	22.61	11.33	8.00	62.57	18.67
o21	**164.22**	157.81	28.57	931.37	108.30
o22	73.83	50.53	17.40	269.62	61.89
r1	66.32	63.77	17.84	380.00	48.59
r2	**10.85**	3.83	5.18	25.56	10.21
r3	16.73	7.07	7.85	52.58	15.23
r4	**7.12**	2.61	3.23	19.57	6.61
r5	**13.74**	5.45	6.77	40.00	12.52

Appendix D

Workflow ordered rank - A3R (p=40)

Table D.1: Full workflow ordered rank considering all datasets - A3R (p=40).

	workflow	repr	stop	stem	spar	info	algo
1	w94	td-idf	default	none	0.98	>0	ld
2	w126	td-idf	default	porter	0.98	>0	ld
3	w158	td-idf	smart	none	0.98	>0	ld
4	w320	freq	default	porter	0.98	>0	rf
5	w62	td-idf	none	porter	0.98	>0	ld
6	w288	freq	default	none	0.98	>0	rf
7	w312	freq	default	porter	0.98	none	rf
8	w30	td-idf	none	none	0.98	>0	ld
9	w256	freq	none	porter	0.98	>0	rf
10	w128	td-idf	default	porter	0.98	>0	rf
11	w224	freq	none	none	0.98	>0	rf
12	w190	td-idf	smart	porter	0.98	>0	ld
13	w384	freq	smart	porter	0.98	>0	rf
14	w174	td-idf	smart	porter	0.99	>0	ld
15	w110	td-idf	default	porter	0.99	>0	ld
16	w64	td-idf	none	porter	0.98	>0	rf
17	w376	freq	smart	porter	0.98	none	rf
18	w96	td-idf	default	none	0.98	>0	rf
19	w296	freq	default	porter	0.99	none	rf
20	w352	freq	smart	none	0.98	>0	rf
21	w142	td-idf	smart	none	0.99	>0	ld
22	w248	freq	none	porter	0.98	none	rf
23	w120	td-idf	default	porter	0.98	none	rf

Continued on next page

Table D.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
24	w32	td-idf	none	none	0.98	>0	rf
25	w304	freq	default	porter	0.99	>0	rf
26	w192	td-idf	smart	porter	0.98	>0	rf
27	w280	freq	default	none	0.98	none	rf
28	w112	td-idf	default	porter	0.99	>0	rf
29	w344	freq	smart	none	0.98	none	rf
30	w360	freq	smart	porter	0.99	none	rf
31	w176	td-idf	smart	porter	0.99	>0	rf
32	w184	td-idf	smart	porter	0.98	none	rf
33	w168	td-idf	smart	porter	0.99	none	rf
34	w78	td-idf	default	none	0.99	>0	ld
35	w368	freq	smart	porter	0.99	>0	rf
36	w46	td-idf	none	porter	0.99	>0	ld
37	w104	td-idf	default	porter	0.99	none	rf
38	w240	freq	none	porter	0.99	>0	rf
39	w160	td-idf	smart	none	0.98	>0	rf
40	w336	freq	smart	none	0.99	>0	rf
41	w56	td-idf	none	porter	0.98	none	rf
42	w272	freq	default	none	0.99	>0	rf
43	w48	td-idf	none	porter	0.99	>0	rf
44	w14	td-idf	none	none	0.99	>0	ld
45	w88	td-idf	default	none	0.98	none	rf
46	w232	freq	none	porter	0.99	none	rf
47	w264	freq	default	none	0.99	none	rf
48	w152	td-idf	smart	none	0.98	none	rf
49	w208	freq	none	none	0.99	>0	rf
50	w144	td-idf	smart	none	0.99	>0	rf
51	w80	td-idf	default	none	0.99	>0	rf
52	w40	td-idf	none	porter	0.99	none	rf
53	w127	td-idf	default	porter	0.98	>0	lsvm
54	w216	freq	none	none	0.98	none	rf
55	w63	td-idf	none	porter	0.98	>0	lsvm
56	w16	td-idf	none	none	0.99	>0	rf
57	w31	td-idf	none	none	0.98	>0	lsvm
58	w24	td-idf	none	none	0.98	none	rf
59	w328	freq	smart	none	0.99	none	rf
60	w319	freq	default	porter	0.98	>0	lsvm
61	w136	td-idf	smart	none	0.99	none	rf
62	w95	td-idf	default	none	0.98	>0	lsvm

Continued on next page

Table D.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
63	w223	freq	none	none	0.98	>0	lsvm
64	w200	freq	none	none	0.99	none	rf
65	w255	freq	none	porter	0.98	>0	lsvm
66	w72	td-idf	default	none	0.99	none	rf
67	w287	freq	default	none	0.98	>0	lsvm
68	w351	freq	smart	none	0.98	>0	lsvm
69	w191	td-idf	smart	porter	0.98	>0	lsvm
70	w383	freq	smart	porter	0.98	>0	lsvm
71	w159	td-idf	smart	none	0.98	>0	lsvm
72	w111	td-idf	default	porter	0.99	>0	lsvm
73	w47	td-idf	none	porter	0.99	>0	lsvm
74	w175	td-idf	smart	porter	0.99	>0	lsvm
75	w8	td-idf	none	none	0.99	none	rf
76	w303	freq	default	porter	0.99	>0	lsvm
77	w239	freq	none	porter	0.99	>0	lsvm
78	w367	freq	smart	porter	0.99	>0	lsvm
79	w335	freq	smart	none	0.99	>0	lsvm
80	w25	td-idf	none	none	0.98	>0	nnet
81	w9	td-idf	none	none	0.99	>0	nnet
82	w143	td-idf	smart	none	0.99	>0	lsvm
83	w79	td-idf	default	none	0.99	>0	lsvm
84	w15	td-idf	none	none	0.99	>0	lsvm
85	w89	td-idf	default	none	0.98	>0	nnet
86	w271	freq	default	none	0.99	>0	lsvm
87	w57	td-idf	none	porter	0.98	>0	nnet
88	w207	freq	none	none	0.99	>0	lsvm
89	w73	td-idf	default	none	0.99	>0	nnet
90	w185	td-idf	smart	porter	0.98	>0	nnet
91	w137	td-idf	smart	none	0.99	>0	nnet
92	w121	td-idf	default	porter	0.98	>0	nnet
93	w281	freq	default	none	0.98	>0	nnet
94	w345	freq	smart	none	0.98	>0	nnet
95	w105	td-idf	default	porter	0.99	>0	nnet
96	w153	td-idf	smart	none	0.98	>0	nnet
97	w313	freq	default	porter	0.98	>0	nnet
98	w361	freq	smart	porter	0.99	>0	nnet
99	w169	td-idf	smart	porter	0.99	>0	nnet
100	w377	freq	smart	porter	0.98	>0	nnet
101	w329	freq	smart	none	0.99	>0	nnet

Continued on next page

Table D.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
102	w297	freq	default	porter	0.99	>0	nnet
103	w253	freq	none	porter	0.98	>0	c5.0
104	w41	td-idf	none	porter	0.99	>0	nnet
105	w217	freq	none	none	0.98	>0	nnet
106	w249	freq	none	porter	0.98	>0	nnet
107	w265	freq	default	none	0.99	>0	nnet
108	w222	freq	none	none	0.98	>0	ld
109	w285	freq	default	none	0.98	>0	c5.0
110	w314	freq	default	porter	0.98	>0	c4.5
111	w286	freq	default	none	0.98	>0	ld
112	w233	freq	none	porter	0.99	>0	nnet
113	w317	freq	default	porter	0.98	>0	c5.0
114	w221	freq	none	none	0.98	>0	c5.0
115	w250	freq	none	porter	0.98	>0	c4.5
116	w334	freq	smart	none	0.99	>0	ld
117	w350	freq	smart	none	0.98	>0	ld
118	w381	freq	smart	porter	0.98	>0	c5.0
119	w270	freq	default	none	0.99	>0	ld
120	w254	freq	none	porter	0.98	>0	ld
121	w206	freq	none	none	0.99	>0	ld
122	w378	freq	smart	porter	0.98	>0	c4.5
123	w318	freq	default	porter	0.98	>0	ld
124	w282	freq	default	none	0.98	>0	c4.5
125	w238	freq	none	porter	0.99	>0	ld
126	w302	freq	default	porter	0.99	>0	ld
127	w349	freq	smart	none	0.98	>0	c5.0
128	w366	freq	smart	porter	0.99	>0	ld
129	w382	freq	smart	porter	0.98	>0	ld
130	w201	freq	none	none	0.99	>0	nnet
131	w218	freq	none	none	0.98	>0	c4.5
132	w122	td-idf	default	porter	0.98	>0	c4.5
133	w237	freq	none	porter	0.99	>0	c5.0
134	w301	freq	default	porter	0.99	>0	c5.0
135	w346	freq	smart	none	0.98	>0	c4.5
136	w186	td-idf	smart	porter	0.98	>0	c4.5
137	w298	freq	default	porter	0.99	>0	c4.5
138	w234	freq	none	porter	0.99	>0	c4.5
139	w125	td-idf	default	porter	0.98	>0	c5.0
140	w365	freq	smart	porter	0.99	>0	c5.0

Continued on next page

Table D.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
141	w362	freq	smart	porter	0.99	>0	c4.5
142	w90	td-idf	default	none	0.98	>0	c4.5
143	w189	td-idf	smart	porter	0.98	>0	c5.0
144	w58	td-idf	none	porter	0.98	>0	c4.5
145	w61	td-idf	none	porter	0.98	>0	c5.0
146	w373	freq	smart	porter	0.98	none	c5.0
147	w103	td-idf	default	porter	0.99	none	lsvm
148	w167	td-idf	smart	porter	0.99	none	lsvm
149	w154	td-idf	smart	none	0.98	>0	c4.5
150	w39	td-idf	none	porter	0.99	none	lsvm
151	w26	td-idf	none	none	0.98	>0	c4.5
152	w309	freq	default	porter	0.98	none	c5.0
153	w97	td-idf	default	porter	0.99	none	nnet
154	w245	freq	none	porter	0.98	none	c5.0
155	w106	td-idf	default	porter	0.99	>0	c4.5
156	w93	td-idf	default	none	0.98	>0	c5.0
157	w170	td-idf	smart	porter	0.99	>0	c4.5
158	w205	freq	none	none	0.99	>0	c5.0
159	w1	td-idf	none	none	0.99	none	nnet
160	w269	freq	default	none	0.99	>0	c5.0
161	w49	td-idf	none	porter	0.98	none	nnet
162	w42	td-idf	none	porter	0.99	>0	c4.5
163	w333	freq	smart	none	0.99	>0	c5.0
164	w109	td-idf	default	porter	0.99	>0	c5.0
165	w277	freq	default	none	0.98	none	c5.0
166	w29	td-idf	none	none	0.98	>0	c5.0
167	w266	freq	default	none	0.99	>0	c4.5
168	w173	td-idf	smart	porter	0.99	>0	c5.0
169	w157	td-idf	smart	none	0.98	>0	c5.0
170	w119	td-idf	default	porter	0.98	none	lsvm
171	w213	freq	none	none	0.98	none	c5.0
172	w55	td-idf	none	porter	0.98	none	lsvm
173	w315	freq	default	porter	0.98	>0	jrip
174	w379	freq	smart	porter	0.98	>0	jrip
175	w129	td-idf	smart	none	0.99	none	nnet
176	w177	td-idf	smart	porter	0.98	none	nnet
177	w150	td-idf	smart	none	0.98	none	ld
178	w183	td-idf	smart	porter	0.98	none	lsvm
179	w7	td-idf	none	none	0.99	none	lsvm

Continued on next page

Table D.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
180	w33	td-idf	none	porter	0.99	none	nnet
181	w357	freq	smart	porter	0.99	none	c5.0
182	w113	td-idf	default	porter	0.98	none	nnet
183	w65	td-idf	default	none	0.99	none	nnet
184	w135	td-idf	smart	none	0.99	none	lsvm
185	w182	td-idf	smart	porter	0.98	none	ld
186	w71	td-idf	default	none	0.99	none	lsvm
187	w145	td-idf	smart	none	0.98	none	nnet
188	w251	freq	none	porter	0.98	>0	jrip
189	w330	freq	smart	none	0.99	>0	c4.5
190	w45	td-idf	none	porter	0.99	>0	c5.0
191	w81	td-idf	default	none	0.98	none	nnet
192	w87	td-idf	default	none	0.98	none	lsvm
193	w347	freq	smart	none	0.98	>0	jrip
194	w151	td-idf	smart	none	0.98	none	lsvm
195	w23	td-idf	none	none	0.98	none	lsvm
196	w202	freq	none	none	0.99	>0	c4.5
197	w17	td-idf	none	none	0.98	none	nnet
198	w293	freq	default	porter	0.99	none	c5.0
199	w161	td-idf	smart	porter	0.99	none	nnet
200	w370	freq	smart	porter	0.98	none	c4.5
201	w283	freq	default	none	0.98	>0	jrip
202	w341	freq	smart	none	0.98	none	c5.0
203	w337	freq	smart	none	0.98	none	nnet
204	w229	freq	none	porter	0.99	none	c5.0
205	w369	freq	smart	porter	0.98	none	nnet
206	w299	freq	default	porter	0.99	>0	jrip
207	w74	td-idf	default	none	0.99	>0	c4.5
208	w325	freq	smart	none	0.99	none	c5.0
209	w363	freq	smart	porter	0.99	>0	jrip
210	w338	freq	smart	none	0.98	none	c4.5
211	w219	freq	none	none	0.98	>0	jrip
212	w187	td-idf	smart	porter	0.98	>0	jrip
213	w138	td-idf	smart	none	0.99	>0	c4.5
214	w77	td-idf	default	none	0.99	>0	c5.0
215	w306	freq	default	porter	0.98	none	c4.5
216	w118	td-idf	default	porter	0.98	none	ld
217	w123	td-idf	default	porter	0.98	>0	jrip
218	w178	td-idf	smart	porter	0.98	none	c4.5

Continued on next page

Table D.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
219	w235	freq	none	porter	0.99	>0	jrip
220	w273	freq	default	none	0.98	none	nnet
221	w305	freq	default	porter	0.98	none	nnet
222	w261	freq	default	none	0.99	none	c5.0
223	w353	freq	smart	porter	0.99	none	nnet
224	w141	td-idf	smart	none	0.99	>0	c5.0
225	w289	freq	default	porter	0.99	none	nnet
226	w321	freq	smart	none	0.99	none	nnet
227	w181	td-idf	smart	porter	0.98	none	c5.0
228	w146	td-idf	smart	none	0.98	none	c4.5
229	w197	freq	none	none	0.99	none	c5.0
230	w13	td-idf	none	none	0.99	>0	c5.0
231	w114	td-idf	default	porter	0.98	none	c4.5
232	w242	freq	none	porter	0.98	none	c4.5
233	w10	td-idf	none	none	0.99	>0	c4.5
234	w331	freq	smart	none	0.99	>0	jrip
235	w375	freq	smart	porter	0.98	none	lsvm
236	w343	freq	smart	none	0.98	none	lsvm
237	w86	td-idf	default	none	0.98	none	ld
238	w91	td-idf	default	none	0.98	>0	jrip
239	w311	freq	default	porter	0.98	none	lsvm
240	w117	td-idf	default	porter	0.98	none	c5.0
241	w267	freq	default	none	0.99	>0	jrip
242	w155	td-idf	smart	none	0.98	>0	jrip
243	w247	freq	none	porter	0.98	none	lsvm
244	w359	freq	smart	porter	0.99	none	lsvm
245	w274	freq	default	none	0.98	none	c4.5
246	w231	freq	none	porter	0.99	none	lsvm
247	w279	freq	default	none	0.98	none	lsvm
248	w44	td-idf	none	porter	0.99	>0	knn
249	w348	freq	smart	none	0.98	>0	knn
250	w295	freq	default	porter	0.99	none	lsvm
251	w380	freq	smart	porter	0.98	>0	knn
252	w59	td-idf	none	porter	0.98	>0	jrip
253	w149	td-idf	smart	none	0.98	none	c5.0
254	w50	td-idf	none	porter	0.98	none	c4.5
255	w215	freq	none	none	0.98	none	lsvm
256	w364	freq	smart	porter	0.99	>0	knn
257	w354	freq	smart	porter	0.99	none	c4.5

Continued on next page

Table D.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
258	w54	td-idf	none	porter	0.98	none	ld
259	w82	td-idf	default	none	0.98	none	c4.5
260	w60	td-idf	none	porter	0.98	>0	knn
261	w171	td-idf	smart	porter	0.99	>0	jrip
262	w257	freq	default	none	0.99	none	nnet
263	w327	freq	smart	none	0.99	none	lsvm
264	w203	freq	none	none	0.99	>0	jrip
265	w107	td-idf	default	porter	0.99	>0	jrip
266	w108	td-idf	default	porter	0.99	>0	knn
267	w172	td-idf	smart	porter	0.99	>0	knn
268	w210	freq	none	none	0.98	none	c4.5
269	w53	td-idf	none	porter	0.98	none	c5.0
270	w18	td-idf	none	none	0.98	none	c4.5
271	w332	freq	smart	none	0.99	>0	knn
272	w188	td-idf	smart	porter	0.98	>0	knn
273	w371	freq	smart	porter	0.98	none	jrip
274	w85	td-idf	default	none	0.98	none	c5.0
275	w28	td-idf	none	none	0.98	>0	knn
276	w263	freq	default	none	0.99	none	lsvm
277	w300	freq	default	porter	0.99	>0	knn
278	w199	freq	none	none	0.99	none	lsvm
279	w162	td-idf	smart	porter	0.99	none	c4.5
280	w316	freq	default	porter	0.98	>0	knn
281	w12	td-idf	none	none	0.99	>0	knn
282	w124	td-idf	default	porter	0.98	>0	knn
283	w22	td-idf	none	none	0.98	none	ld
284	w284	freq	default	none	0.98	>0	knn
285	w165	td-idf	smart	porter	0.99	none	c5.0
286	w290	freq	default	porter	0.99	none	c4.5
287	w43	td-idf	none	porter	0.99	>0	jrip
288	w27	td-idf	none	none	0.98	>0	jrip
289	w140	td-idf	smart	none	0.99	>0	knn
290	w156	td-idf	smart	none	0.98	>0	knn
291	w241	freq	none	porter	0.98	none	nnet
292	w21	td-idf	none	none	0.98	none	c5.0
293	w307	freq	default	porter	0.98	none	jrip
294	w339	freq	smart	none	0.98	none	jrip
295	w139	td-idf	smart	none	0.99	>0	jrip
296	w98	td-idf	default	porter	0.99	none	c4.5

Continued on next page

Table D.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
297	w76	td-idf	default	none	0.99	>0	knn
298	w179	td-idf	smart	porter	0.98	none	jrip
299	w92	td-idf	default	none	0.98	>0	knn
300	w101	td-idf	default	porter	0.99	none	c5.0
301	w322	freq	smart	none	0.99	none	c4.5
302	w268	freq	default	none	0.99	>0	knn
303	w226	freq	none	porter	0.99	none	c4.5
304	w75	td-idf	default	none	0.99	>0	jrip
305	w209	freq	none	none	0.98	none	nnet
306	w275	freq	default	none	0.98	none	jrip
307	w37	td-idf	none	porter	0.99	none	c5.0
308	w147	td-idf	smart	none	0.98	none	jrip
309	w130	td-idf	smart	none	0.99	none	c4.5
310	w243	freq	none	porter	0.98	none	jrip
311	w115	td-idf	default	porter	0.98	none	jrip
312	w133	td-idf	smart	none	0.99	none	c5.0
313	w225	freq	none	porter	0.99	none	nnet
314	w34	td-idf	none	porter	0.99	none	c4.5
315	w193	freq	none	none	0.99	none	nnet
316	w69	td-idf	default	none	0.99	none	c5.0
317	w11	td-idf	none	none	0.99	>0	jrip
318	w258	freq	default	none	0.99	none	c4.5
319	w66	td-idf	default	none	0.99	none	c4.5
320	w5	td-idf	none	none	0.99	none	c5.0
321	w355	freq	smart	porter	0.99	none	jrip
322	w83	td-idf	default	none	0.98	none	jrip
323	w194	freq	none	none	0.99	none	c4.5
324	w291	freq	default	porter	0.99	none	jrip
325	w342	freq	smart	none	0.98	none	ld
326	w374	freq	smart	porter	0.98	none	ld
327	w211	freq	none	none	0.98	none	jrip
328	w2	td-idf	none	none	0.99	none	c4.5
329	w51	td-idf	none	porter	0.98	none	jrip
330	w166	td-idf	smart	porter	0.99	none	ld
331	w163	td-idf	smart	porter	0.99	none	jrip
332	w323	freq	smart	none	0.99	none	jrip
333	w236	freq	none	porter	0.99	>0	knn
334	w310	freq	default	porter	0.98	none	ld
335	w227	freq	none	porter	0.99	none	jrip

Continued on next page

Table D.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
336	w99	td-idf	default	porter	0.99	none	jrip
337	w259	freq	default	none	0.99	none	jrip
338	w278	freq	default	none	0.98	none	ld
339	w252	freq	none	porter	0.98	>0	knn
340	w246	freq	none	porter	0.98	none	ld
341	w19	td-idf	none	none	0.98	none	jrip
342	w131	td-idf	smart	none	0.99	none	jrip
343	w102	td-idf	default	porter	0.99	none	ld
344	w134	td-idf	smart	none	0.99	none	ld
345	w35	td-idf	none	porter	0.99	none	jrip
346	w214	freq	none	none	0.98	none	ld
347	w67	td-idf	default	none	0.99	none	jrip
348	w195	freq	none	none	0.99	none	jrip
349	w220	freq	none	none	0.98	>0	knn
350	w204	freq	none	none	0.99	>0	knn
351	w372	freq	smart	porter	0.98	none	knn
352	w38	td-idf	none	porter	0.99	none	ld
353	w340	freq	smart	none	0.98	none	knn
354	w70	td-idf	default	none	0.99	none	ld
355	w3	td-idf	none	none	0.99	none	jrip
356	w308	freq	default	porter	0.98	none	knn
357	w356	freq	smart	porter	0.99	none	knn
358	w358	freq	smart	porter	0.99	none	ld
359	w180	td-idf	smart	porter	0.98	none	knn
360	w6	td-idf	none	none	0.99	none	ld
361	w294	freq	default	porter	0.99	none	ld
362	w52	td-idf	none	porter	0.98	none	knn
363	w324	freq	smart	none	0.99	none	knn
364	w276	freq	default	none	0.98	none	knn
365	w116	td-idf	default	porter	0.98	none	knn
366	w148	td-idf	smart	none	0.98	none	knn
367	w292	freq	default	porter	0.99	none	knn
368	w326	freq	smart	none	0.99	none	ld
369	w230	freq	none	porter	0.99	none	ld
370	w20	td-idf	none	none	0.98	none	knn
371	w262	freq	default	none	0.99	none	ld
372	w164	td-idf	smart	porter	0.99	none	knn
373	w84	td-idf	default	none	0.98	none	knn
374	w198	freq	none	none	0.99	none	ld

Continued on next page

Table D.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
375	w260	freq	default	none	0.99	none	knn
376	w244	freq	none	porter	0.98	none	knn
377	w36	td-idf	none	porter	0.99	none	knn
378	w100	td-idf	default	porter	0.99	none	knn
379	w228	freq	none	porter	0.99	none	knn
380	w132	td-idf	smart	none	0.99	none	knn
381	w212	freq	none	none	0.98	none	knn
382	w196	freq	none	none	0.99	none	knn
383	w4	td-idf	none	none	0.99	none	knn
384	w68	td-idf	default	none	0.99	none	knn

Appendix E

Workflow ordered rank - Accuracy

Table E.1: Full workflow ordered rank considering all datasets - Accuracy only.

	workflow	repr	stop	stem	spar	info	algo
1	w360	freq	smart	porter	0.99	none	rf
2	w168	td-idf	smart	porter	0.99	none	rf
3	w296	freq	default	porter	0.99	none	rf
4	w104	td-idf	default	porter	0.99	none	rf
5	w328	freq	smart	none	0.99	none	rf
6	w264	freq	default	none	0.99	none	rf
7	w232	freq	none	porter	0.99	none	rf
8	w136	td-idf	smart	none	0.99	none	rf
9	w40	td-idf	none	porter	0.99	none	rf
10	w72	td-idf	default	none	0.99	none	rf
11	w176	td-idf	smart	porter	0.99	>0	rf
12	w312	freq	default	porter	0.98	none	rf
13	w376	freq	smart	porter	0.98	none	rf
14	w112	td-idf	default	porter	0.99	>0	rf
15	w184	td-idf	smart	porter	0.98	none	rf
16	w368	freq	smart	porter	0.99	>0	rf
17	w304	freq	default	porter	0.99	>0	rf
18	w120	td-idf	default	porter	0.98	none	rf
19	w200	freq	none	none	0.99	none	rf
20	w48	td-idf	none	porter	0.99	>0	rf
21	w336	freq	smart	none	0.99	>0	rf
22	w272	freq	default	none	0.99	>0	rf
23	w174	td-idf	smart	porter	0.99	>0	ld

Continued on next page

Table E.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
24	w144	td-idf	smart	none	0.99	>0	rf
25	w8	td-idf	none	none	0.99	none	rf
26	w240	freq	none	porter	0.99	>0	rf
27	w248	freq	none	porter	0.98	none	rf
28	w80	td-idf	default	none	0.99	>0	rf
29	w56	td-idf	none	porter	0.98	none	rf
30	w192	td-idf	smart	porter	0.98	>0	rf
31	w110	td-idf	default	porter	0.99	>0	ld
32	w384	freq	smart	porter	0.98	>0	rf
33	w128	td-idf	default	porter	0.98	>0	rf
34	w344	freq	smart	none	0.98	none	rf
35	w320	freq	default	porter	0.98	>0	rf
36	w16	td-idf	none	none	0.99	>0	rf
37	w280	freq	default	none	0.98	none	rf
38	w142	td-idf	smart	none	0.99	>0	ld
39	w152	td-idf	smart	none	0.98	none	rf
40	w88	td-idf	default	none	0.98	none	rf
41	w208	freq	none	none	0.99	>0	rf
42	w64	td-idf	none	porter	0.98	>0	rf
43	w46	td-idf	none	porter	0.99	>0	ld
44	w256	freq	none	porter	0.98	>0	rf
45	w78	td-idf	default	none	0.99	>0	ld
46	w24	td-idf	none	none	0.98	none	rf
47	w352	freq	smart	none	0.98	>0	rf
48	w216	freq	none	none	0.98	none	rf
49	w190	td-idf	smart	porter	0.98	>0	ld
50	w175	td-idf	smart	porter	0.99	>0	lsvm
51	w288	freq	default	none	0.98	>0	rf
52	w126	td-idf	default	porter	0.98	>0	ld
53	w96	td-idf	default	none	0.98	>0	rf
54	w111	td-idf	default	porter	0.99	>0	lsvm
55	w14	td-idf	none	none	0.99	>0	ld
56	w47	td-idf	none	porter	0.99	>0	lsvm
57	w160	td-idf	smart	none	0.98	>0	rf
58	w367	freq	smart	porter	0.99	>0	lsvm
59	w32	td-idf	none	none	0.98	>0	rf
60	w303	freq	default	porter	0.99	>0	lsvm
61	w224	freq	none	none	0.98	>0	rf
62	w62	td-idf	none	porter	0.98	>0	ld

Continued on next page

Table E.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
63	w335	freq	smart	none	0.99	>0	lsvm
64	w127	td-idf	default	porter	0.98	>0	lsvm
65	w239	freq	none	porter	0.99	>0	lsvm
66	w143	td-idf	smart	none	0.99	>0	lsvm
67	w158	td-idf	smart	none	0.98	>0	ld
68	w79	td-idf	default	none	0.99	>0	lsvm
69	w191	td-idf	smart	porter	0.98	>0	lsvm
70	w15	td-idf	none	none	0.99	>0	lsvm
71	w63	td-idf	none	porter	0.98	>0	lsvm
72	w94	td-idf	default	none	0.98	>0	ld
73	w319	freq	default	porter	0.98	>0	lsvm
74	w383	freq	smart	porter	0.98	>0	lsvm
75	w271	freq	default	none	0.99	>0	lsvm
76	w207	freq	none	none	0.99	>0	lsvm
77	w255	freq	none	porter	0.98	>0	lsvm
78	w30	td-idf	none	none	0.98	>0	ld
79	w31	td-idf	none	none	0.98	>0	lsvm
80	w159	td-idf	smart	none	0.98	>0	lsvm
81	w351	freq	smart	none	0.98	>0	lsvm
82	w95	td-idf	default	none	0.98	>0	lsvm
83	w9	td-idf	none	none	0.99	>0	nnet
84	w223	freq	none	none	0.98	>0	lsvm
85	w287	freq	default	none	0.98	>0	lsvm
86	w137	td-idf	smart	none	0.99	>0	nnet
87	w361	freq	smart	porter	0.99	>0	nnet
88	w73	td-idf	default	none	0.99	>0	nnet
89	w105	td-idf	default	porter	0.99	>0	nnet
90	w329	freq	smart	none	0.99	>0	nnet
91	w297	freq	default	porter	0.99	>0	nnet
92	w169	td-idf	smart	porter	0.99	>0	nnet
93	w185	td-idf	smart	porter	0.98	>0	nnet
94	w57	td-idf	none	porter	0.98	>0	nnet
95	w41	td-idf	none	porter	0.99	>0	nnet
96	w265	freq	default	none	0.99	>0	nnet
97	w121	td-idf	default	porter	0.98	>0	nnet
98	w103	td-idf	default	porter	0.99	none	lsvm
99	w39	td-idf	none	porter	0.99	none	lsvm
100	w25	td-idf	none	none	0.98	>0	nnet
101	w89	td-idf	default	none	0.98	>0	nnet

Continued on next page

Table E.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
102	w233	freq	none	porter	0.99	>0	nnet
103	w167	td-idf	smart	porter	0.99	none	lsvm
104	w377	freq	smart	porter	0.98	>0	nnet
105	w1	td-idf	none	none	0.99	none	nnet
106	w313	freq	default	porter	0.98	>0	nnet
107	w97	td-idf	default	porter	0.99	none	nnet
108	w366	freq	smart	porter	0.99	>0	ld
109	w345	freq	smart	none	0.98	>0	nnet
110	w334	freq	smart	none	0.99	>0	ld
111	w7	td-idf	none	none	0.99	none	lsvm
112	w302	freq	default	porter	0.99	>0	ld
113	w270	freq	default	none	0.99	>0	ld
114	w281	freq	default	none	0.98	>0	nnet
115	w238	freq	none	porter	0.99	>0	ld
116	w153	td-idf	smart	none	0.98	>0	nnet
117	w71	td-idf	default	none	0.99	none	lsvm
118	w206	freq	none	none	0.99	>0	ld
119	w135	td-idf	smart	none	0.99	none	lsvm
120	w249	freq	none	porter	0.98	>0	nnet
121	w65	td-idf	default	none	0.99	none	nnet
122	w201	freq	none	none	0.99	>0	nnet
123	w362	freq	smart	porter	0.99	>0	c4.5
124	w129	td-idf	smart	none	0.99	none	nnet
125	w33	td-idf	none	porter	0.99	none	nnet
126	w237	freq	none	porter	0.99	>0	c5.0
127	w298	freq	default	porter	0.99	>0	c4.5
128	w301	freq	default	porter	0.99	>0	c5.0
129	w365	freq	smart	porter	0.99	>0	c5.0
130	w234	freq	none	porter	0.99	>0	c4.5
131	w378	freq	smart	porter	0.98	>0	c4.5
132	w173	td-idf	smart	porter	0.99	>0	c5.0
133	w109	td-idf	default	porter	0.99	>0	c5.0
134	w317	freq	default	porter	0.98	>0	c5.0
135	w353	freq	smart	porter	0.99	none	nnet
136	w217	freq	none	none	0.98	>0	nnet
137	w381	freq	smart	porter	0.98	>0	c5.0
138	w314	freq	default	porter	0.98	>0	c4.5
139	w357	freq	smart	porter	0.99	none	c5.0
140	w170	td-idf	smart	porter	0.99	>0	c4.5

Continued on next page

Table E.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
141	w253	freq	none	porter	0.98	>0	c5.0
142	w125	td-idf	default	porter	0.98	>0	c5.0
143	w189	td-idf	smart	porter	0.98	>0	c5.0
144	w382	freq	smart	porter	0.98	>0	ld
145	w321	freq	smart	none	0.99	none	nnet
146	w106	td-idf	default	porter	0.99	>0	c4.5
147	w289	freq	default	porter	0.99	none	nnet
148	w161	td-idf	smart	porter	0.99	none	nnet
149	w318	freq	default	porter	0.98	>0	ld
150	w250	freq	none	porter	0.98	>0	c4.5
151	w45	td-idf	none	porter	0.99	>0	c5.0
152	w183	td-idf	smart	porter	0.98	none	lsvm
153	w254	freq	none	porter	0.98	>0	ld
154	w186	td-idf	smart	porter	0.98	>0	c4.5
155	w119	td-idf	default	porter	0.98	none	lsvm
156	w165	td-idf	smart	porter	0.99	none	c5.0
157	w55	td-idf	none	porter	0.98	none	lsvm
158	w350	freq	smart	none	0.98	>0	ld
159	w122	td-idf	default	porter	0.98	>0	c4.5
160	w293	freq	default	porter	0.99	none	c5.0
161	w42	td-idf	none	porter	0.99	>0	c4.5
162	w373	freq	smart	porter	0.98	none	c5.0
163	w61	td-idf	none	porter	0.98	>0	c5.0
164	w286	freq	default	none	0.98	>0	ld
165	w101	td-idf	default	porter	0.99	none	c5.0
166	w229	freq	none	porter	0.99	none	c5.0
167	w266	freq	default	none	0.99	>0	c4.5
168	w181	td-idf	smart	porter	0.98	none	c5.0
169	w49	td-idf	none	porter	0.98	none	nnet
170	w257	freq	default	none	0.99	none	nnet
171	w269	freq	default	none	0.99	>0	c5.0
172	w222	freq	none	none	0.98	>0	ld
173	w333	freq	smart	none	0.99	>0	c5.0
174	w354	freq	smart	porter	0.99	none	c4.5
175	w205	freq	none	none	0.99	>0	c5.0
176	w330	freq	smart	none	0.99	>0	c4.5
177	w285	freq	default	none	0.98	>0	c5.0
178	w349	freq	smart	none	0.98	>0	c5.0
179	w325	freq	smart	none	0.99	none	c5.0

Continued on next page

Table E.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
180	w363	freq	smart	porter	0.99	>0	jrip
181	w282	freq	default	none	0.98	>0	c4.5
182	w37	td-idf	none	porter	0.99	none	c5.0
183	w87	td-idf	default	none	0.98	none	lsvm
184	w309	freq	default	porter	0.98	none	c5.0
185	w151	td-idf	smart	none	0.98	none	lsvm
186	w23	td-idf	none	none	0.98	none	lsvm
187	w369	freq	smart	porter	0.98	none	nnet
188	w113	td-idf	default	porter	0.98	none	nnet
189	w299	freq	default	porter	0.99	>0	jrip
190	w221	freq	none	none	0.98	>0	c5.0
191	w379	freq	smart	porter	0.98	>0	jrip
192	w77	td-idf	default	none	0.99	>0	c5.0
193	w117	td-idf	default	porter	0.98	none	c5.0
194	w245	freq	none	porter	0.98	none	c5.0
195	w177	td-idf	smart	porter	0.98	none	nnet
196	w346	freq	smart	none	0.98	>0	c4.5
197	w58	td-idf	none	porter	0.98	>0	c4.5
198	w182	td-idf	smart	porter	0.98	none	ld
199	w141	td-idf	smart	none	0.99	>0	c5.0
200	w202	freq	none	none	0.99	>0	c4.5
201	w370	freq	smart	porter	0.98	none	c4.5
202	w162	td-idf	smart	porter	0.99	none	c4.5
203	w290	freq	default	porter	0.99	none	c4.5
204	w235	freq	none	porter	0.99	>0	jrip
205	w93	td-idf	default	none	0.98	>0	c5.0
206	w359	freq	smart	porter	0.99	none	lsvm
207	w331	freq	smart	none	0.99	>0	jrip
208	w261	freq	default	none	0.99	none	c5.0
209	w74	td-idf	default	none	0.99	>0	c4.5
210	w315	freq	default	porter	0.98	>0	jrip
211	w53	td-idf	none	porter	0.98	none	c5.0
212	w337	freq	smart	none	0.98	none	nnet
213	w13	td-idf	none	none	0.99	>0	c5.0
214	w157	td-idf	smart	none	0.98	>0	c5.0
215	w295	freq	default	porter	0.99	none	lsvm
216	w305	freq	default	porter	0.98	none	nnet
217	w231	freq	none	porter	0.99	none	lsvm
218	w133	td-idf	smart	none	0.99	none	c5.0

Continued on next page

Table E.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
219	w138	td-idf	smart	none	0.99	>0	c4.5
220	w145	td-idf	smart	none	0.98	none	nnet
221	w81	td-idf	default	none	0.98	none	nnet
222	w341	freq	smart	none	0.98	none	c5.0
223	w218	freq	none	none	0.98	>0	c4.5
224	w197	freq	none	none	0.99	none	c5.0
225	w291	freq	default	porter	0.99	none	jrip
226	w347	freq	smart	none	0.98	>0	jrip
227	w98	td-idf	default	porter	0.99	none	c4.5
228	w17	td-idf	none	none	0.98	none	nnet
229	w226	freq	none	porter	0.99	none	c4.5
230	w375	freq	smart	porter	0.98	none	lsvm
231	w150	td-idf	smart	none	0.98	none	ld
232	w154	td-idf	smart	none	0.98	>0	c4.5
233	w355	freq	smart	porter	0.99	none	jrip
234	w29	td-idf	none	none	0.98	>0	c5.0
235	w273	freq	default	none	0.98	none	nnet
236	w251	freq	none	porter	0.98	>0	jrip
237	w90	td-idf	default	none	0.98	>0	c4.5
238	w267	freq	default	none	0.99	>0	jrip
239	w69	td-idf	default	none	0.99	none	c5.0
240	w371	freq	smart	porter	0.98	none	jrip
241	w327	freq	smart	none	0.99	none	lsvm
242	w322	freq	smart	none	0.99	none	c4.5
243	w178	td-idf	smart	porter	0.98	none	c4.5
244	w306	freq	default	porter	0.98	none	c4.5
245	w199	freq	none	none	0.99	none	lsvm
246	w263	freq	default	none	0.99	none	lsvm
247	w149	td-idf	smart	none	0.98	none	c5.0
248	w225	freq	none	porter	0.99	none	nnet
249	w187	td-idf	smart	porter	0.98	>0	jrip
250	w5	td-idf	none	none	0.99	none	c5.0
251	w277	freq	default	none	0.98	none	c5.0
252	w242	freq	none	porter	0.98	none	c4.5
253	w10	td-idf	none	none	0.99	>0	c4.5
254	w171	td-idf	smart	porter	0.99	>0	jrip
255	w311	freq	default	porter	0.98	none	lsvm
256	w85	td-idf	default	none	0.98	none	c5.0
257	w364	freq	smart	porter	0.99	>0	knn

Continued on next page

Table E.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
258	w343	freq	smart	none	0.98	none	lsvm
259	w44	td-idf	none	porter	0.99	>0	knn
260	w34	td-idf	none	porter	0.99	none	c4.5
261	w323	freq	smart	none	0.99	none	jrip
262	w114	td-idf	default	porter	0.98	none	c4.5
263	w247	freq	none	porter	0.98	none	lsvm
264	w107	td-idf	default	porter	0.99	>0	jrip
265	w26	td-idf	none	none	0.98	>0	c4.5
266	w283	freq	default	none	0.98	>0	jrip
267	w130	td-idf	smart	none	0.99	none	c4.5
268	w307	freq	default	porter	0.98	none	jrip
269	w213	freq	none	none	0.98	none	c5.0
270	w203	freq	none	none	0.99	>0	jrip
271	w118	td-idf	default	porter	0.98	none	ld
272	w338	freq	smart	none	0.98	none	c4.5
273	w172	td-idf	smart	porter	0.99	>0	knn
274	w193	freq	none	none	0.99	none	nnet
275	w332	freq	smart	none	0.99	>0	knn
276	w108	td-idf	default	porter	0.99	>0	knn
277	w163	td-idf	smart	porter	0.99	none	jrip
278	w259	freq	default	none	0.99	none	jrip
279	w50	td-idf	none	porter	0.98	none	c4.5
280	w123	td-idf	default	porter	0.98	>0	jrip
281	w179	td-idf	smart	porter	0.98	none	jrip
282	w279	freq	default	none	0.98	none	lsvm
283	w146	td-idf	smart	none	0.98	none	c4.5
284	w21	td-idf	none	none	0.98	none	c5.0
285	w227	freq	none	porter	0.99	none	jrip
286	w258	freq	default	none	0.99	none	c4.5
287	w300	freq	default	porter	0.99	>0	knn
288	w66	td-idf	default	none	0.99	none	c4.5
289	w12	td-idf	none	none	0.99	>0	knn
290	w241	freq	none	porter	0.98	none	nnet
291	w215	freq	none	none	0.98	none	lsvm
292	w339	freq	smart	none	0.98	none	jrip
293	w243	freq	none	porter	0.98	none	jrip
294	w43	td-idf	none	porter	0.99	>0	jrip
295	w380	freq	smart	porter	0.98	>0	knn
296	w139	td-idf	smart	none	0.99	>0	jrip

Continued on next page

Table E.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
297	w99	td-idf	default	porter	0.99	none	jrip
298	w2	td-idf	none	none	0.99	none	c4.5
299	w194	freq	none	none	0.99	none	c4.5
300	w219	freq	none	none	0.98	>0	jrip
301	w140	td-idf	smart	none	0.99	>0	knn
302	w274	freq	default	none	0.98	none	c4.5
303	w275	freq	default	none	0.98	none	jrip
304	w86	td-idf	default	none	0.98	none	ld
305	w115	td-idf	default	porter	0.98	none	jrip
306	w76	td-idf	default	none	0.99	>0	knn
307	w268	freq	default	none	0.99	>0	knn
308	w75	td-idf	default	none	0.99	>0	jrip
309	w54	td-idf	none	porter	0.98	none	ld
310	w60	td-idf	none	porter	0.98	>0	knn
311	w18	td-idf	none	none	0.98	none	c4.5
312	w82	td-idf	default	none	0.98	none	c4.5
313	w188	td-idf	smart	porter	0.98	>0	knn
314	w348	freq	smart	none	0.98	>0	knn
315	w210	freq	none	none	0.98	none	c4.5
316	w59	td-idf	none	porter	0.98	>0	jrip
317	w209	freq	none	none	0.98	none	nnet
318	w147	td-idf	smart	none	0.98	none	jrip
319	w91	td-idf	default	none	0.98	>0	jrip
320	w155	td-idf	smart	none	0.98	>0	jrip
321	w195	freq	none	none	0.99	none	jrip
322	w131	td-idf	smart	none	0.99	none	jrip
323	w67	td-idf	default	none	0.99	none	jrip
324	w316	freq	default	porter	0.98	>0	knn
325	w28	td-idf	none	none	0.98	>0	knn
326	w124	td-idf	default	porter	0.98	>0	knn
327	w35	td-idf	none	porter	0.99	none	jrip
328	w284	freq	default	none	0.98	>0	knn
329	w156	td-idf	smart	none	0.98	>0	knn
330	w211	freq	none	none	0.98	none	jrip
331	w83	td-idf	default	none	0.98	none	jrip
332	w22	td-idf	none	none	0.98	none	ld
333	w11	td-idf	none	none	0.99	>0	jrip
334	w51	td-idf	none	porter	0.98	none	jrip
335	w166	td-idf	smart	porter	0.99	none	ld

Continued on next page

Table E.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
336	w92	td-idf	default	none	0.98	>0	knn
337	w27	td-idf	none	none	0.98	>0	jrip
338	w3	td-idf	none	none	0.99	none	jrip
339	w236	freq	none	porter	0.99	>0	knn
340	w374	freq	smart	porter	0.98	none	ld
341	w19	td-idf	none	none	0.98	none	jrip
342	w102	td-idf	default	porter	0.99	none	ld
343	w134	td-idf	smart	none	0.99	none	ld
344	w342	freq	smart	none	0.98	none	ld
345	w372	freq	smart	porter	0.98	none	knn
346	w310	freq	default	porter	0.98	none	ld
347	w252	freq	none	porter	0.98	>0	knn
348	w356	freq	smart	porter	0.99	none	knn
349	w38	td-idf	none	porter	0.99	none	ld
350	w204	freq	none	none	0.99	>0	knn
351	w70	td-idf	default	none	0.99	none	ld
352	w340	freq	smart	none	0.98	none	knn
353	w246	freq	none	porter	0.98	none	ld
354	w308	freq	default	porter	0.98	none	knn
355	w324	freq	smart	none	0.99	none	knn
356	w278	freq	default	none	0.98	none	ld
357	w292	freq	default	porter	0.99	none	knn
358	w180	td-idf	smart	porter	0.98	none	knn
359	w358	freq	smart	porter	0.99	none	ld
360	w6	td-idf	none	none	0.99	none	ld
361	w220	freq	none	none	0.98	>0	knn
362	w52	td-idf	none	porter	0.98	none	knn
363	w214	freq	none	none	0.98	none	ld
364	w116	td-idf	default	porter	0.98	none	knn
365	w260	freq	default	none	0.99	none	knn
366	w294	freq	default	porter	0.99	none	ld
367	w164	td-idf	smart	porter	0.99	none	knn
368	w276	freq	default	none	0.98	none	knn
369	w326	freq	smart	none	0.99	none	ld
370	w148	td-idf	smart	none	0.98	none	knn
371	w36	td-idf	none	porter	0.99	none	knn
372	w230	freq	none	porter	0.99	none	ld
373	w20	td-idf	none	none	0.98	none	knn
374	w100	td-idf	default	porter	0.99	none	knn

Continued on next page

Table E.1 – continued from previous page

	workflow	repr	stop	stem	spar	info	algo
375	w262	freq	default	none	0.99	none	ld
376	w228	freq	none	porter	0.99	none	knn
377	w84	td-idf	default	none	0.98	none	knn
378	w198	freq	none	none	0.99	none	ld
379	w132	td-idf	smart	none	0.99	none	knn
380	w244	freq	none	porter	0.98	none	knn
381	w4	td-idf	none	none	0.99	none	knn
382	w68	td-idf	default	none	0.99	none	knn
383	w196	freq	none	none	0.99	none	knn
384	w212	freq	none	none	0.98	none	knn

{huuuuuuuuuuuuug}