Workflow Selection and Hyper-Parameters Tuning using Meta-Learning

by

Anthi Papadopoulou

For the degree of Master of Science, Data Analytics

Supervised by

Pavel Brazdil Professor Catedrático Jubilado

Faculdade de Economia

Universidade do Porto

2019

Acknowledgments

To my supervisor Professor Pavel Brazdil who introduced me to the world of text mining in the classes of MADSAD. From back then and until the finalization of this project, he has been always attentive, supportive and patient. To my family that still doubts that I will ever finalize this dissertation. Special thanks to Fernando Duarte and Maria Ferreira and the community of *stack overflow* for their support to complete this research.

Resumo

Durante a década passada, Machine Learning (ML) revelou bons modelos nos sistemas de aprendizagem que desempenharam eficientemente sem intervenção humana. O modo como as decisões são tomadas, desde tarefas simples como escolher a melhor rota para alcançar um destino até tarefas complexas como desenhar resultado de modelo de rede de fornecimento, é um combinação de teorias de inteligência artificial e ferramentas que os humanos aplicam para obter a decisão. Sem minimizar a importância do fator humano, as máquinas têm provado que conseguem superar os homens em tarefas específicas e dar apoio útil em qualquer área científica.

Este progresso tem trazido, naturalmente, uma abundância de diferentes metodologias para solucionar problemas. Dado que regras de ouro não existem, selecionar o algoritmo de aprendizagem correto para um problema particular é tão complicado quanto essencial. A teoria de meta-aprendizagem foi desenvolvida para ultrapassar esta deficiência e dar uma resposta robusta a uma simples questão: *qual é a melhor forma de treinar os meus dados?*

Meta-aprendizagem traz vantagens em grande variedade de problemas de aprendizagem, contudo, aqui estamos interessados no seu poder preditivo em tarefas de classificação de texto (TC). TC tem ganho muito interesse desde o "boom" da Internet que tornou a informação textual disponível a qualquer pessoa, em qualquer lugar, a qualquer hora.

No geral, nesta investigação, nós focamo-nos em explorar quatro modelos de classificação populares usando um conjunto de parâmetros e hiper-parâmetros adequados em cada treino. No total, nós treinamos 204 pipelines de fluxo de trabalho em *20newsgroups* dataset. Os algoritmos seguem diferentes métodos de pré-processamento e variam na complexidade computacional. Por fim, aplicamos uma medida de meta-aprendizagem, A3R, de forma a ordenar e selecionar os algoritmos mais eficientes e efetivos.

Palavras-Chave: meta-learning, workflow selection problem, text classification, machine learning

Abstract

During the last decade, Machine Learning (ML) has shown astonishing paradigms of learning systems that perform accurately and efficiently without any human intervention. The way decisions are made, from a simple task as choosing the best path for reaching a destination to more tasks, such as designing a supply chain model, is a combination of artificial intelligence theories and tools that humans apply to reach decision. Without minimizing the importance of the human factor, machines have proven that can outperform humans in specific tasks and provide assistance in any scientific field.

This progress has naturally brought an abundance of different methodologies for solving problems. As golden rules do not exist, selecting the right learning algorithm for a particular problem is as tricky as essential. Meta-learning theory was developed in order to tackle this deficiency and give a strong answer to the simplest question: *which is the best way to train my data?*

Meta-learning brings advantages in a wide range of learning problems, however here, we are interested in its predictive power over Text Classification (TC) tasks. TC has gained lots of interest since the boom of internet that made textual information available to anyone, anywhere, at anytime.

Overall, in this particular research, we focused on exploring four popular classification models using an adequate set of hyper-parameters in training. In total, we have trained 204 workflow pipelines on *20newsgroups* dataset. The algorithms follow different pre-processing methods and vary in computational complexity. Last, we applied a meta-learning measure to rank and select the most efficient and effective algorithm.

Keywords: meta-learning, workflow selection problem, text classification, machine learning

Index

List of Tables

List of Figures

Acronyms

ML Machine Learning

TC Text Classification

A3R Average 3 Ranking

NLP Natural Language Processing

LR Linear Regression

OLS Ordinary Least Squares

RF Random Forest

SVM Support Vector Machine

DT Decision Trees

tf-idf term frequency-inverse document frequency matrix

BoW Bag of Words

PMI Pointwise Mutual Information

HAL Hyperspace Analogue to Language model

SVD Singular Value Decomposition

COALS Correlated Occurrence Analogue to Lexical Semantics

LSA Latent Semantic Analysis

GloVe Global Vectors for Word Representation

IG -Information Gain

FS -Feature Selection

AT Active Testing

AR Average Ranking

DL Deep Learning

ANN Artificial Neural Networks

CNN Convolutional Neural Network

RNN Recurrent Neural Network

Chapter 1

Introduction

In Machine Learning (ML), new methods and algorithms are proposed regularly. Favoured and widely used methods that have been in the field for many years are being challenged by modern more complex models. The increased computational capabilities of computers, the abundance of information and the high demand for automatization, has brought new dynamics to the field (Socher, 2014).

Deep Learning (DL), has become the new panacea for any problem and it is considered the hottest topic in any artificial intelligence discussion (Lohr, 2019). However, as promising as DL models can be, there is still no comprehensive theoretical understanding of the learning process involved (Shwartz-Ziv and Tishby, 2017). More than that, by their natural computational composition, they demand large datasets in order to learn and consequently require a lot of time to train. On the other hand, classical methods, such as Random Forest (RF), can often achieve satisfying results while avoiding the previous limitations.

In ML problems are divided into*classification*, when the predicted value is a discrete label, and *regression*, when the output is a numeric quantity. In this particular work, we are exploring classification methods on text. Text classification is the process of handling unstructured text in order to categorize it into predefined classes.

Overall, the issue that we are interested to investigate: given a specific text the question is which algorithm will be the best in terms of accuracy and runtime. To answer this, we have adopted the meta-learning approach that casts the algorithm selection problem as a learning problem.

1.1 Objectives and contribution of this thesis

In this thesis, our objectives can be summarized below:

- Conduct a profound study and gather extensive knowledge of state-of-the-art pre-processing techniques and classification models for unstructured text.

- Create workflows based on the previous studies and use them in a given set of classification tasks.
- Compare different workflows using the methodology of ranking and meta-learning.
- Generate a ranking of potential useful workflows.

To achieve these objectives, a number of pre-selected models with different architectures and workflows were trained on raw text datasets. Finally all the models will be compared in terms of accuraacy and runtime.

1.2 Overview of the thesis

In the following chapters, we discuss the methodology used and describe our experimental study. More specifically: Chapter 2 dedicated to reviewing related work; Chapter 3 describes to the methodology followed and the description of the experiments conducted; Chapter 4 provides the summary of the results and concludes our work.

Chapter 2

Literature Review

In this chapter we cover the bibliography that our work was based on. In the following sections, we explain the fundamentals of text classification and representation and the classification models used in our work. The forms of meta-learning and algorithm selection theory is also covered as they are used in our work.

2.1 Text Classification

On the core of our research approach lays text classification. Text Classification (TC), also referred to as *text categorization* or *topic attribution*, is defined the activity of automatically classifying natural language text to thematic categories from a predefined set (Sebastiani, 2002). The labels usually represent the main topic of the document (Aggarwal and Zhai, 2012), but also could represent different level of sentiment of the documents (Melville et al., 2009) or spam versus non-spam (Sakkis et al., 2001). TC has been widely used in real-world applications, such as spam detection (Jindal and Liu, 2007), fraud detection (Airoldi and Malin, 2004), emergency response (Caragea et al., 2016), question answering (Madabushi and Lee, 2016), legal discovery (Roitblat et al., 2010) etc.

We can define the classification problem as a set of training records

$$D = \{X_1, X_2, ...X_N\} \tag{2.1}$$

such that each record is labeled with a class value drawn from a set of K different discrete values indexed by *{1...k}* (Aggarwal and Zhai, 2012). The classification model, fed by the training data, captures the relations between features and assigns the class labels:

$$\phi = D \times \ K \rightarrow \{1, 0\} \tag{2.2}$$

where ϕ is the classifying function (classifier). The labels could be represented by binary numbers, 1 for the correct label and 0 for the incorrect are for each instance.

For assigning labels to an unknown set, the 2.2 equation is transformed to:

$$\hat{\Phi} = \hat{D} \times \ C \rightarrow \{1, 0\} \tag{2.3}$$

2.2 Text Representation

Starting point of TC is the representation of the text into a format that is compatible with the model specifications. The most common practice is the representation of a corpus by a *term–document* matrix, where rows correspond to terms and columns correspond to documents (Turney and Pantel, 2010). In *document-term* matrix, the columns and rows are interchanged. In its basic form, rows include a set of unordered words, commonly known as bag-of-words Bag of Words (BoW). A special case of BoW are n-grams where the sequence of n elements follow the sequence of the words in the text (Sidorov et al., 2014). Other schemes include chunks of text, such as phrases.

Assume X to be a *term–document matrix* of a collection with n documents and m unique terms. The matrix X will then have m rows (one row for each unique term in the vocabulary) and n columns (one column for each document). Let x_i be the *i-th* term in the vocabulary and d_j be the *j-th* document in the collection. The component x_{ij} in X represents the frequency of the *i-th* term in the *j-th* document d_j. The matrix depicts the frequency of each term to every document, providing a measure of relevance between the documents (Salton et al., 1975). In general, the value of most of the elements in X will be zero (the matrix is usually sparse), since most documents will use only a small fraction of the whole vocabulary. If we randomly choose a term w_i and a document d_j, it is likely that w_i does not occur anywhere in d_j, and therefore x_{ij} equals 0.

A direct way to improve the performance of a system based on *tf* is by adding a weight to the frequencies. Spärck Jones (1972) proposed a novel approach of creating the matrix, term frequency-inverse document frequency matrix (tf-idf). The main idea is that the frequency of a term in a document should be inversely proportional to its frequency in other documents in the corpus. High weights are attributed to terms with high *tf* and low *df*. The tf-idf equation is summarized below:

$$tf - idf = tf_{m,i,j} \times \log \frac{\mid D \mid}{D(t_m)} \tag{2.4}$$

2.3 Text pre-processing methods

Pre-processing is the first step in text classification. As pre-processing we define the procedure of cleansing and preparing raw text into a form that contains all the useful information of the given text and can also serve as input to a predictive

model. The challenge rises as real world text data are usually incomplete, noisy and inconsistent. Incomplete data are those that may lack relevant words. Noisy data comprise of errors and outliers, while inconsistent contain discrepancies in codes or names (Nayak, 2016).

According to Fayyad et al. (Fayyad et al., 1996), the total percentage of noise in datasets may reach 40, a fact that causes problem to in machine learning algorithms. In order to tackle these issues different methods have been developed. Symeonidis et al. (Symeonidis et al., 2018) investigated the interactions among pre-processing methods via ablation and identified the sixteen most useful and popular methods. Table 2.1 summarizes them:

Pre-processing Techniques
Basic (Remove Unicode strings and noise)
Other (Replace URLs and user mentions)
Replace Slang and Abbreviations
Replace Contractions
Remove Numbers
Replace Repetitions of Punctuation
Replace Negations with Antonyms
Remove Punctuation
Handling Capitalized Words
Lowercase
Remove Stopwords
Replace Elongated Words
Spelling Correction
Part of Speech Tagging
Lemmatizing
Stemming
Handling Negations

Table 2.1: Table of the most common Pre-processing techniques (Symeonidis et al., 2018).

Tokenization Although not included in the table, tokenization is considered to be the first step of text pre-processing. Albert and Atkinson (Albert and Atkinson, 2005) defined tokenization as a kind of lexical analysis that breaks a stream of text up into words, phrases, symbols, or other meaningful elements called *tokens*. In its standard version, a token represents a single "word", where a word is considered to be a string of characters that is separated by special characters, such as space, tab or line. Repeated words can be grouped into unique tokens matrix. However, multi-word expressions represent a problem. The description of a house in the form of *white house* is different from *White House*, as the second term represents a *named*

entity. Another problem that can occur when homograph and homonym are treated as if they had the same meaning.

Remove text elements Remove not useful information A classic technique in information retrieval and data mining is to remove any character or word that has a low impact on the outcome. For this reason, we remove non-alphabetic characters, unicode strings, numbers, punctuation symbols, and stopwords. Stopwords are function words with high frequencies of presence across all sentences. They are considered dispensable as they do not contain much useful information for further analysis. The set of these words is predefined, but can be changed by removing or adding more works to it, depending on the application. The lists vary depending on the language and the application used. Some of the most popular are the SMART system that contains 571 English words, proposed by Squire et al. (Squire et al., 2000) and grammatical words (Fox, 2009) that include 421 words. DIALOG information service (Harter, 1986) includes nine items ("an," "and," "by," "for," "from," "of," "the," "to, "with").

Capitalized words Another method is to transform all words that contain any upper case letter to lower. By doing so, the same words are merged and the dimensionality of the problem is reduced. For example, the text *I spilled milk all up in my Macbook.* is transformed after this pre-processing step to *i spill milk all up in my macbook.*

Stemming Is the procedure aiming to reduce all words with the same stem (or, if pre-fixes are left untouched, the same stem) to a common form (Lovins, 1968). To give an example consider the following phrase: *Saying good morning to everyone. Another gorgeous morning in Surrey. Wishing you all well.* After applying stemming (in particular Porter Stemmer it is transformed into *Say good morn to everyon. Anoth gorgeou morn in Surrey. Wish you all well.*

By its computational nature, a stemming algorithm has inherent limitations. The routine handles individual words: it has no access to information about their grammatical and semantic relations with one another. In fact, it is based on the assumption of close agreement of meaning between words with the same root. This assumption, while workable in most cases, in English represents an approximation at best.

The most popular stemming algorithm for TC tasks is Porter's original English stemming algorithm. Other common implementations are Snowball algorithm (Agichtein and Gravano, 2000) that supports a number of languages such as French, German, Hungarian, Italian etc., Lovins' English stemmer, Kraaij-Pohlmann Dutch stemmer, and a variation of the German stemmer which normalises umlauts.

Feature selection - (IG) *Feature Selection* is often used in text categorization

to reduce the size of the dimension of the initial corpora. It is a selection process of extracting the most important features in order to reach faster performance without affecting accuracy (Liu et al., 2005). A number of feature selection methods are successfully used in a wide range of text categorizations. Yang and Pedersen (1997) compared five feature selection methods for text categorization including IG, statistic document frequency, term strength, and mutual information concluding that IG excels the others. Other approaches inspired by optimization theory, genetic algorithm and ant colony optimization (Asghari and Aghdam, 2010).

IG is one of the popular approaches employed as a term importance criterion in the text document data (Joachims, 1998b). The idea is based on information theory (Hammond and Mitchell, 1997). The information gain of term t is defined in Eq. (4)

$$IG(t) = -\sum_{i=1}^{|C|} P(c_i) logP(c_i)) + P(t) \sum_{i=1}^{|C|} P(c_i) logP(c_i|t) + P(\bar{t}) \sum_{i=1}^{|C|} P(c_i|\bar{t}) logP(c_i|\bar{t}) \tag{2.5}$$

where c_i represents the i_{th} category, $P(c_i)$ is the probability of the i_{th} category, *P(t)* and $P(c_i)$ the probabilities that the term t appears or not in the documents, respectively, $P(c_i/t)$ is the conditional probability of the i_{th} category given that term t appeared, and $P(c_i/t)$ is the conditional probability of the i_{th} category given that term t does not appeared.

In this study, before dimension reduction, each term within the text is ranked depending on their importance for the classification in decreasing order using the IG method. Thereby, in the process of text categorization, terms of less importance are ignored, and dimension reduction methods are applied to the terms of highest importance.

Sparsity correction There have been several studies that attempted to overcome the data sparseness to get a better (semantic) similarity. Sparsity correction is applied in big word matrices in order to remove rare terms. This method ensures that each term of the dataset is present in a minimum percentage of documents across the document collection. The minimum threshold can be set by the user according to their needs. In *R* this method can be applied through the *removeSparseTerms* function in the *tm* package (Feinerer and Hornik, 2012). Near the other extreme, if *sparse = .01*, then only terms that appear in (nearly) every document will be retained. Of course this depends on the number of terms and the number of documents, and in natural language, common words like "the" are likely to occur in every document and hence never be "sparse".

2.4 Machine Learning Algorithms

In ML classification is a supervised learning approach in which the algorithm learns from data to predict a label it. There are many different types of classification algorithms and an extensive bibliography that covers them. For practical reasons we are going to focus on the ones that we have applied in this work.

R

2.4.1 Linear Regression

As the name implies, Linear Regression (LR) is a *linear* model that tries to identify linear relationships between the explanatory (dependent) variables and the predicted (independent) label (Seber and Lee, 2012). The predictive function of the model is a *linear* function that measures the accuracy by its squared *residual*. *Residual* is the difference between the observed value and the estimated value, where in this case this difference equals to distance.

Suppose that x_i is the dependent variable, y_i is the independent variable, for *i=1,...,n* samples, a LR model would be formed as:

$$y_i = a + bx_i + e_i \tag{2.6}$$

where a and b are the parameters that the linear function tries to fit to the initial data and e is the *residual*. As our goal is to minimize e, we could transform the equation 2.16 to a minimization problem (Montgomery et al., 2012):

$$forQ(a,b) = \sum_{i}^{n} \hat{e}_i^2 = \sum_{i}^{n} (y_i - a - bx_i)^2 \tag{2.7}$$

The *square* calculation of the *e residual* is based on the principles of Ordinary Least Squares (OLS). It is a common estimator that tries to minimize the square difference between the real and predicted values. Other variations of OLS inlcude, *least absolute deviations* or introduce a *cost* function (Montgomery et al., 2012).

2.4.2 Random Forest

RF is an ensemble method of Decision Trees (DT) generated on a random split of the initial dataset. In order to understand it, we will start by explaining how a DT works. A DT is a decision support algorithm that uses a tree-like graph that contains conditional control statements (Quinlan, 1986). The algorithm splits the initial data into smaller and smaller subsets that form a decision tree with *nodes* and *leaves*. By the term *nodes* we refer to two or more branches while *leaves* represent the decision between the predictive classes. The node at the crest of the tree corresponds to the best predictor called root node (see Fig. 2.1).

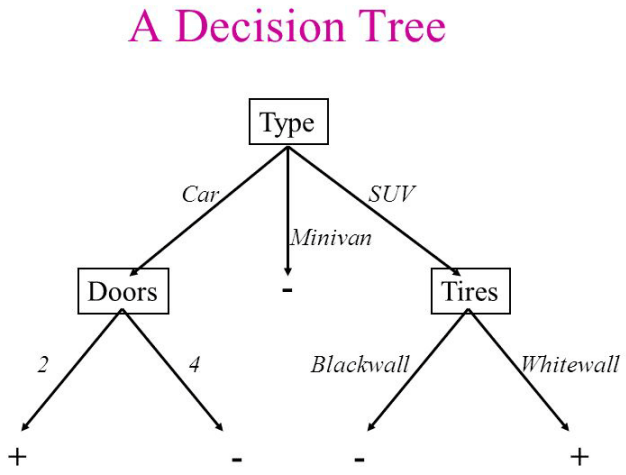

Figure 2.1: Decision Tree graph.

Instead of training one tree and getting its predictions, RF creates random trees and performs a vote for each predicted result. The final predicted class is the one that will prevail in the majority of the trees (Pal, 2005). In this way, the predictive value of the final output is more powerful. Cases of datasets with skewed distributions or missing data are handled in an efficient way. Another advantage is that RF can be applied to both regression and classification problems.

2.4.3 SVM

Support Vector Machine (SVM) is another algorithm that is applicable to both regression and classification problems. In SVM, the initial data are represented as points in space (see Fig. 2.2), mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible (Joachims, 1998a). In other words, the larger the margin, the lower the error. These mappings schemes are designed to ensure that dot products of the input data may be computed based on the variables in the original space, by defining them in terms of a kernel function *k(x,y)* (Joachims, 1998a).

For a linear 2-dimention space, we could define the mathematical formula of the *hyperplane* as:

$$b_0 + b_1 * x_1 + b_2 * x_2 = 0 \tag{2.8}$$

where $b_i x_i$ is the dot product of the vector b_i with the dependent variable. The equation equals to 0 as the objective is to find the margin between the two classes.

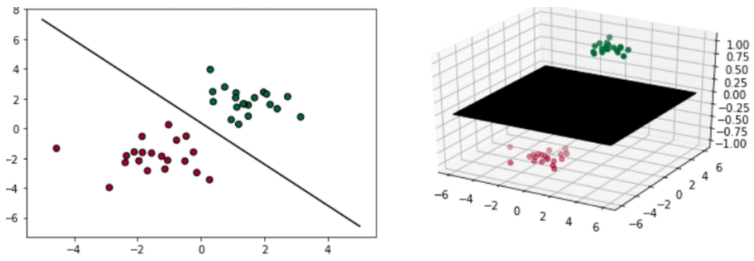

Figure 2.2: Hyperplane line for 2-dimention space (left). Hyperplane plane for 3-dimention space (right).

In case, the classes of the independent variables were y=*{-1,1}*, the equation (2.18) would be formulated as:

$$b_0 + b_1 * x_1 + b_2 * x_2 = 1 \tag{2.9}$$

for any value on or above this boundary for y=*1*, and

$$b_0 + b_1 * x_1 + b_2 * x_2 = -1 \tag{2.10}$$

for y=*-1*.

2.4.4 Deep Learning

DL has become the most favored model for solving any machine learning problem. Although the idea of neural networks started from the 50's, it was until 2012 when Alex Krizhevsky won *ImageNet* Large Scale Visual Recognition Challenge with his model of Convolutional Neural Network (CNN) named *AlexNet* (Krizhevsky et al., 2012). After that, a boom in academic research and in business applications proved that deep learning models can achieve higher accuracy in language modeling (Graves et al., 2013) (Pascanu et al., 2013) (Le and Mikolov, 2014) (Sutskever et al., 2011), learning word embeddings (Mikolov et al., 2013), online handwritten recognition (Graves et al., 2008), speech recognition (Graves et al., 2013), objection recognition (Krizhevsky et al., 2012).

In our research effort three different architectures of DL models have been applied. For a better understanding of DL theory and applications, we will cover the mathematical origins of artificial neural networks and the most prominent architectures.

Artificial Neural Networks (ANN), although inspired by the cyclical connectivity of neurons in the brain, cannot reach the complexity of the human brain (McCullough et al., 2002). There are two key similarities between biological and artificial neural networks. First, the building units of both networks are highly intercon-

nected and second, the connections between the neurons determine the function of the network (Bengio et al., 2013).

Similar to any other machine-learning model, DL aims at predicting the output of a given a problem. The process works as follows: given an initial data-set (in this case, we will refer to it as input layer), the model will transform it and learn patterns (in the hidden layers) in order to produce the final result (output layer). Learning, in the context of ML and consequently in DL , refers to the automated process of data transformations in order to obtain better results (Chollet et al., 2018). However in the case of the latter the learning is a process of successive layers of transformations where at each stage the overall performance is increased.

ANN In its simplest form a ANN consists of one feed-forward neuron input (single-input neuron). As the figure below shows2.3, input p is multiplied by the weight w resulting the product wp. After adding the bias, the created net input layer will pass to the activation function (hidden layer). A is the final output produced by the function. The process is stated as feed-forward as it starts from the input and ends at the output moving on only one direction as the arrows show.

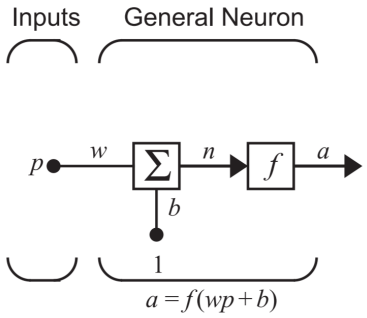

Figure 2.3: Single-Input Neuron (Demuth et al., 2014).

The above description could be summarized into the following equation:

$$a = f(wp + b) \tag{2.11}$$

where w , p and b are scalars.

Transfer functions are chosen based on the specificities of the problem given. A variety of functions exists, that could be clustered into three main categories: *linear*, *sigmoid* and *hard limist*. Linear functions output is equal to its input, meaning that a=n. The hard limit transfer function sets the output of the neuron to 0 if the function argument is less than 0, or 1 if its argument is greater than or equal to 0. The *log-sigmoid* transfer function takes the input (which may have any value between plus and minus infinity) and squashes the output into the range 0 to 1. The a here is defined as follows:

$$a = \frac{1}{1 + e^{-n}} \tag{2.12}$$

It is commonly used in multi-layer networks that are trained using the back-propagation algorithm, in part because this function is differentiable. On the table below, we can see how different functions work for a n input and a output.

A neuron with more than one input is considered a multiple-input neuron. The image below (Fig. 2.4) sums up a multiple-input neuron:

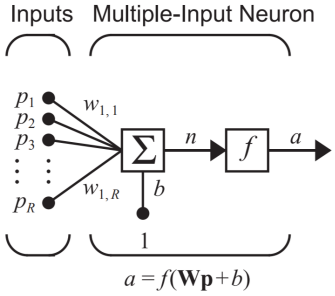

Figure 2.4: Multiple-Input Neuron (Demuth et al., 2014).

In this case the weighted net input is now defined as:

$$n = w_{1,1}p_1 + w_{1,2}p_2 + ... + w_{1,R}p_R + b \tag{2.13}$$

$$n = \mathbf{Wp} + \mathbf{b} \tag{2.14}$$

where W now stands for a matrix, p for a vector and b for a scalar, and the final output equals:

$$a = f(\mathbf{Wp} + b) \tag{2.15}$$

The complexity grows when instead of training one hidden layer at a time, train multiple layers together. Each layer is trying to learn different aspects about the data by minimizing an error or cost function. The updated model looks now (Fig. 2.5):

Convolutional Neural Network **CNN** is an algorithm based on the architecture of deep neural networks. The term "convolution" refers to the linear operations that the algorithm performs instead of general matrix multiplication between the layers (Pascanu et al., 2013). The first CNN models were trained to classify image (Cireşan et al., 2012) and since then they are used in various other problems such as text classification (Yang et al., 2016) (Zhou et al., 2016).

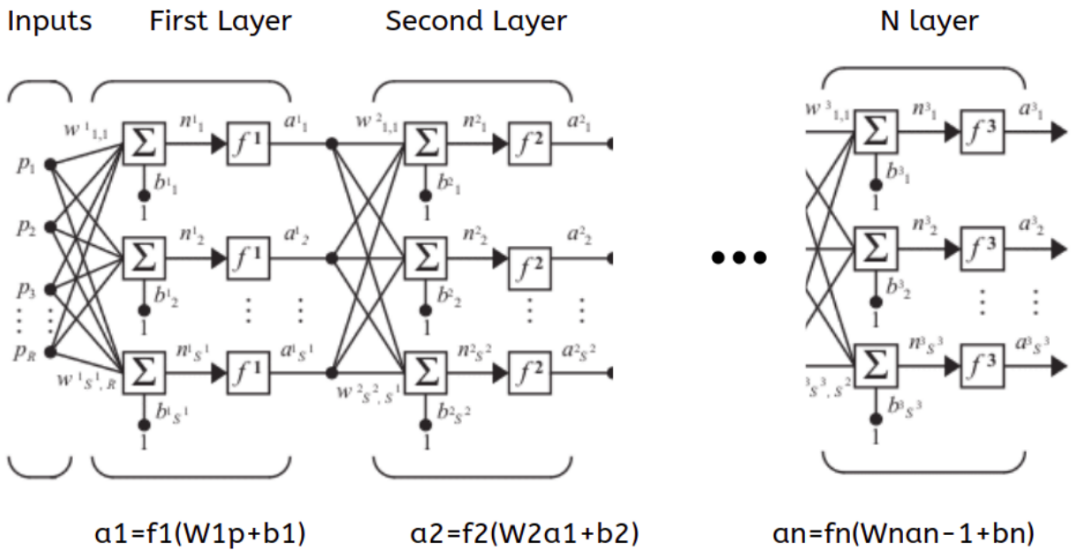

Figure 2.5: n-Layer Network (Demuth et al., 2014).

Recurrent Neural Network **RNN** is another class of deep learning that was first introduced by Rumelhart and McClelland (1986). Their main characteristic is that, instead of feeding each layer with the result of previous layer, each layer receives the previous result and the previous input. A new parameter is added in the model time. The decision a current network reaches at time t_{-1} is determined by the decision at time t_0.

The process we have just described could be defined as the following learning function:

$$h_t = \phi(Wx_t + Uh_{t-1}) \tag{2.16}$$

where h_t is the hidden state at time t_0. Then x_t is multiplied by a weight matrix W and summed-up with the product of U hidden state h_{t-1} at time t_{-1}.

2.5 Algorithm Selection Problem

In 1976, John Rice explored the idea that the problems that algorithms tackle and the corresponding models follow the same rules. He advocated that although exploitation of a specific problem and domain would always be necessary, the selection of the right algorithm and parameters should be based on the previous knowledge gained (meta-knowledge) from approximately similar problems (Rice, 1976). Rice explored the selection problem as a problem per-se, by analyzing the applicability of the approximation theory to the algorithm selection problem. Figure 2.6 illustrates

this process.

Later, Wolpert et al. (1997) proposed the *"No Free Lunch Theorems for Optimization"* that rejects the idea that a universal optimal learning algorithm is possible. They advocated that if any pair of algorithms perform on average equally for a specific domain, then for any similar domain the results will be also similar.

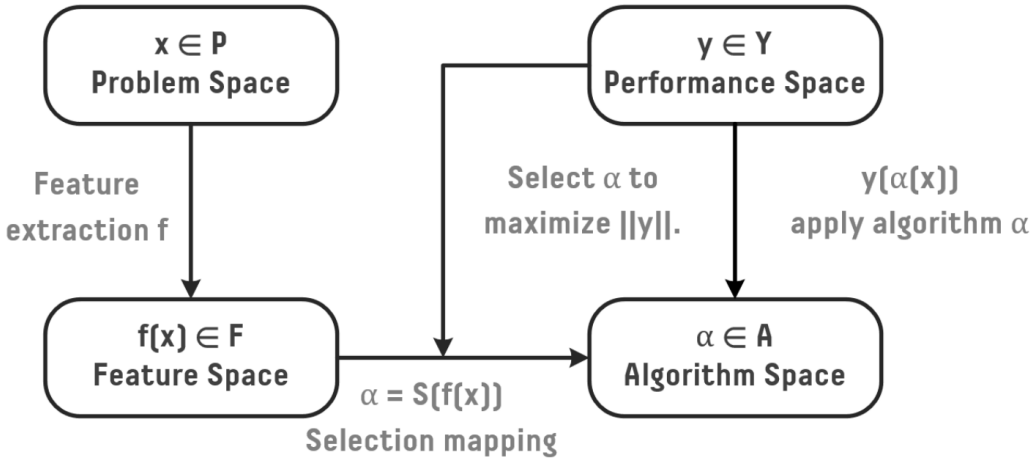

Figure 2.6: Schematic diagram of Rice's algorithm selection problem framework (Rice, 1976). The objective is to determine the selection mapping S that returns the best algorithm. Adapted from Smith-Miles (2009).

Overall, the algorithm selection problem addresses the importance of the decision which learning algorithms to use and how to tune them (Maron and Moore, 1997). Selecting the right predictive model for a given problem can be challenging considering the vast number of predictive models that exist. Training a data-set using all possible classifiers can be timely and computationally exhausting, let alone applying different sets of parameters at each model. Meta-learning studies how the knowledge gained from previous problems could serve in solving new ones (Brazdil et al., 2008).

Abdulrahman et al. Abdulrahman et al. (2017) introduced a multi-objective measure, (A3R), for identifying the best algorithm. The authors explored two techniques, Average Ranking (AR) and Active Testing (AT), that select algorithms based on rankings over efficiency and effectiveness. As its name implies,AR (Brazdil et al., 2003) ranks the algorithms based on the average accuracy of their performance over any prior domain. Average 3 Ranking (A3R) takes into account not only accuracy, but also the computational time of each algorithm.

Introduced by Leite et al. (Leite and Brazdil, 2010), AT aims at selecting those algorithms that have greater probability to outperform the performance of previously

tested algorithms by using the concept of estimates of performance gain (Abdulrahman et al., 2017). The final selection is based on the relative estimate that one algorithm will be better than the previous selected. A3R measure is formulated as:

$$A3R^{d_i}_{a_{ref},a_j} = \frac{\frac{SR^{d_i}_{a_j}}{SR^{d_i}_{a_{ref}}}}{(T^{d_i}_{a_j}/T^{d_i}_{a_{ref}})^P} \tag{2.17}$$

where a_j is the algorithm to be tested, a_{ref} is the current algorithm, d_i is the training domain for both algorithms, SR is the accuracy score (success rate) and T is the computational time counted in seconds. Finally, time is raised in power P, a number close to $1/40$, in order to balance the trade-off time-accuracy. In this scenario, there are two important parameters. First, the loss function L that measures the difference between the real and predicted labels and second the bias that is the measure of the assumptions that the learner creates when predicts for instances that are different or inexistent in the training domain (Mitchell, 1980).

Chapter 3

Methodology, Experiments and Results

3.1 Problem Overview

Our research was focused on exploring text pre-processing techniques and classification algorithms, creating various workflows that could be compared and ranked. The work was divided into three parts: first, applying a number of text pre-processing methods and tune them. Second, follow the same process and extend the existing pipelines with text classification methods, paying special attention to deep neural networks exploration. Finally, create rankings by comparing the resulting pipelines using meta-learning.

As the number of values can be infinite, we focused on creating ranges around the default number of the best algorithms proposed in the existing literature. OpenML (Van Rijn et al., 2013) is a platform that inspired us in our approach to create workflows. It is a research platform where anyone can train and test workflows, publish their configuration and results. The database serves as a base for comparing the results of different methodologies on popular domains. Therefore, it could be considered as a source of meta-data and meta-knowledge information.

We have followed the methodology of Maria Ferreira (2017). This researcher applied workflow recommendation for text classification using meta-learning. As our approach is very similar, we followed the process of tuning the parameters of the text pre-processing (except word embeddings) and some classification algorithms.

3.2 Datasets

The *20 newsgroups* dataset is comprised of 18,846 news documents that are evenly divided into 20 classes. It is an open free source dataset that was created by Ken Lang and it can be retrieved from (Rennie, 2008). Each document contains a text

of approximately 1700 words in English and belongs to the given class. The subject of some of the classes are closely related, which increases difficulty of the correct classification.

Reuters-21578 dataset (Lewis, 1997) is the second document collection that we used. In its original version, it contains 21,578 newspaper articles in the form of text documents. The collection is considered highly skewed as it contains categories with only few documents belong to some of them. In our case, we used the split of *Mod-Apte* that divides it into 90 categories.

From the above dataset collections, we decided to use the complete *20 newsgroups* dataset in conjunction with the 11 most frequent categories of the *Reuters-21578* dataset. The final set is a compilation of 6 different groups (aka *groups*), each containing 5 classes. Later, when we create our workflows, each group is trained separately. Table 3.1 summarizes the documents.

Table 3.1: Six groups of documents used in the study.

code	**source**	**topic**	**no. docs**
group 1	*20 newsgroups*	alt.atheism	1000
	20 newsgroups	talk.religion.misc	1000
	20 newsgroups	soc.religion.christian	1000
	Reuters-21578	acq	1650
	Reuters-21578	grain	433
group 2	*20 newsgroups*	comp.graphics	1000
	20 newsgroups	comp.os.ms-windows.misc	1000
	20 newsgroups	comp.sys.ibm.pc.hardware	1000
	20 newsgroups	comp.sys.mac.hardware	1000
	20 newsgroups	comp.windows.x	1000
group 3	*20 newsgroups*	rec.autos	1000
	20 newsgroups	rec.motorcycles	1000
	20 newsgroups	rec.sport.baseball	1000
	20 newsgroups	rec.sport.hockey	1000
	Reuters-21578	ship	197
group 4	*20 newsgroups*	sci.crypt	1000
	20 newsgroups	sci.electronics	1000
	20 newsgroups	sci.med	1000
	20 newsgroups	sci.space	1000
	Reuters-21578	dlr	131
group 5	*20 newsgroups*	talk.politics.misc	1000
	20 newsgroups	talk.politics.guns	1000
	20 newsgroups	talk.politics.mideast	1000

Continued on next page

Table 3.1 – continued from previous page

code	source	topic	no. docs
	Reuters-21578	crude	389
	Reuters-21578	gold	94
group 6	*20 newsgroups*	ms.forsale	1000
	Reuters-21578	earn	2877
	Reuters-21578	money-fx	538
	Reuters-21578	interest	347
	Reuters-21578	money-supply	140

3.3 Experimental Set-up

The experiments conducted were carried out on a pc, where the specifications are in Table A.1) across the period of three months. First, we applied the pre-processing experiments, creating 60 different workflows (pipelines). After, we fed the results to the *four classification algorithms* that we had pre-selected, resulting to *194 workflows*. The run-time of the two separated processes was added up and the final accuracy of each workflow was stored. These values were finally used to elaborate the final rankings.

Text pre-processing

The first step to process the text was to convert all capital letters to lowercase and other transformations including *representation choice, Stemming, Sparsity correction, Stop-word removal and Information gain feature selection*. Table 3.2 summarizes the pre-processing steps used.

Table 3.2: Pre-processing methods with the correspondent codified term.

Method	Options	Code
Representation	*tf-idf*	*tf-idf*
	frequency	*freq*
	1000 word embbeding	*emb1*
	2000 word embbeding	*emb2*
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

As DL models incorporate in their modeling the text representation, applying *sparsity correction* and *information gain* to *word embeddings* is considered out of the scope of this text representation method. As a result, *word embeddings* were combined with *Stop-word removal and Stemming* and used only for training the CNN model.

Table 3.3: Pre-processing strategies applied on the experiments.

pre-proc.	**repr**	**stop**	**stem**	**spar**	**info**
p1	tf-idf	none	none	0,99	none
p2	tf-idf	none	none	0,99	>0
p3	tf-idf	none	none	0,98	none
p4	tf-idf	none	none	0,98	>0
p5	tf-idf	none	porter	0,99	none
p6	tf-idf	none	porter	0,99	>0
p7	tf-idf	none	porter	0,98	none
p8	tf-idf	none	porter	0,98	>0
p9	tf-idf	default	none	0,99	none
p10	tf-idf	default	none	0,99	>0
p11	tf-idf	default	none	0,98	none
p12	tf-idf	default	none	0,98	>0

Continued on next page

Table 3.3 – continued from previous page

pre-proc.	repr	stop	stem	spar	info
p13	tf-idf	default	porter	0,99	none
p14	tf-idf	default	porter	0,99	>0
p15	tf-idf	default	porter	0,98	none
p16	tf-idf	default	porter	0,98	>0
p17	tf-idf	smart	none	0,99	none
p18	tf-idf	smart	none	0,99	>0
p19	tf-idf	smart	none	0,98	none
p20	tf-idf	smart	none	0,98	>0
p21	tf-idf	smart	porter	0,99	none
p22	tf-idf	smart	porter	0,99	>0
p23	tf-idf	smart	porter	0,98	none
p24	tf-idf	smart	porter	0,98	>0
p25	freq	none	none	0,99	none
p26	freq	none	none	0,99	>0
p27	freq	none	none	0,98	none
p28	freq	none	none	0,98	>0
p29	freq	none	porter	0,99	none
p30	freq	none	porter	0,99	>0
p31	freq	none	porter	0,98	none
p32	freq	none	porter	0,98	>0
p33	freq	default	none	0,99	none
p34	freq	default	none	0,99	>0
p35	freq	default	none	0,98	none
p36	freq	default	none	0,98	>0
p37	freq	default	porter	0,99	none
p38	freq	default	porter	0,99	>0
p39	freq	default	porter	0,98	none
p40	freq	default	porter	0,98	>0
p41	freq	smart	none	0,99	none
p42	freq	smart	none	0,99	>0
p43	freq	smart	none	0,98	none
p44	freq	smart	none	0,98	>0
p45	freq	smart	porter	0,99	none
p46	freq	smart	porter	0,99	>0
p47	freq	smart	porter	0,98	none
p48	freq	smart	porter	0,98	>0
p49	emb1	none	none	none	none
p50	emb1	none	porter	none	none
p51	emb1	default	none	none	none

Continued on next page

Table 3.3 – continued from previous page

pre-proc.	repr	stop	stem	spar	info
p52	emb1	default	porter	none	none
p53	emb1	smart	none	none	none
p54	emb1	smart	porter	none	none
p55	emb2	none	none	none	none
p56	emb2	none	porter	none	none
p57	emb2	default	none	none	none
p58	emb2	default	porter	none	none
p59	emb2	smart	none	none	none
p60	emb2	smart	porter	none	none

```
num_words <- min(MAX_NUM_WORDS, length(word_index) + 1)
prepare_embedding_matrix <- function() {
  embedding_matrix <- matrix(0L, nrow = num_words, ncol = EMBEDDING_DIM)
  for (word in names(word_index)) {
    index <- word_index[[word]]
    if (index >= MAX_NUM_WORDS)
      next
    embedding_vector <- embeddings_index[[word]]
    if (!is.null(embedding_vector)) {
      embedding_matrix[index,] <- embedding_vector
    }
  }
  embedding_matrix
}

embedding_matrix <- prepare_embedding_matrix()
```

Figure 3.1: Code for the training of the embedding word matrix in R.

The packages that we used were: *tm, NLP, FSelector, Rweka, caret and keras* in *R* language.

Classification Algorithms

We applied five classification models, namely: *Linear Regression, Quadratic Regression, Random Forest, Support Vector Machine* and from the DL theory CNN. Table 3.4 is the summary of the algorithms used and the parameters that we used to tune them. As the CNN model a bigger number of parameters, we created a separate Table 3.5 that includes them.

The packages that we used were: *MASS, e1071, randomForest and keras* in *R* language.

Model	Parameters	Code
Linear Regression	-	*ld*
Quadratic Regression	-	*ld*
Random Forest	mtry=round down no.features	*rf*
SVM	*cost=1*	*svm*
CNN	*Conv2D, epochs=10*	*cnn*

Table 3.4: Classification models with the correspondent parameters and codified term.

Parameter	Size
Maximum sequence length	1000
Embedding dimension	150
Filter size	[3,4,5]
No filters	600
Drop	0,5
Maximum pool	Maximum sequence length[i] - filter size[i]
Batch size	30

Table 3.5: Specification of CNN model parameters.

Training & Testing sets

As explained in Chapter 3, we applied our workflows on the documents of *20newsgroups* and *reuters*. We created 6 dataset groups that each one contains five classes with similar topic. In order to ensure the accuracy of our models, we run 10-fold cross validation with *leave-one-out*. That means that each dataset group was split into 10 random samples, from which the 9 were used as training set and the tenth as testing. The final output of the prediction is the class with the highest mean probability. The size of the split was in *0,2%*. As in pre-processing phase, we didn't apply cross-validation in the CNN model, but similarly to it, the split size of the CNN was *0,2%* and the training epochs 10. By this technique we limit the probability of over-fitting and also overpass the problem of a skewed distribution on the dataset.

The packages that we used were: *cvTools* and *keras* in *R* language.

3.4 Ranking Workflows

Based on the work of Abdulrahman et al. (2017), we followed their approach to rank our meta-data collection. The researches propose that accuracy should be combined with run-time, providing a ranking of algorithms that are both accurate and efficient.

The original equation of A3R (see *equation 2.15*) measure unnecessarily complex for methods that do not use pairwise comparison. For this reason, we have adopted the simpler version of Van Rijn et al. (2013):

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

The P parameter featured in this formula is basically the weighting parameter for the run-time in this measure. In our work, *P=1/40*.

The next step after obtaining the results of the A3R measure, is to rank them. For that we applied AR method as described in (Abdulrahman et al., 2017). That means that each workflow was ranked based on every group separately. In our case, we calculated 6 different ranking sets as we had 6 groups. These 6 rankings were after averaged based on AR in order to get the final complete ranking. In case of a tie, we assign to each tied value the average of the ranks that would have been assigned without ties.

The table 3.6 shows how the ranking between the best workflow can be altered. CNN models exceed in accuracy but finally over-passed by the RF model that needs only few minutes to train.

workflow	accuracy	A3R
w193	1	4
w195	2	6
w197	3	8
w202	4	9
w194	5	10
w198	6	11
w152	7	1
w167	8	2
w156	9	3
w81	10	7
w187	11	15

Table 3.6: Workflow and its ranks based on accuracy and A3R.

3.5 Results

From the learning algorithms that we trained, we could conclude that DL models result in high accuracy. By looking at the accuracy and run-time of the CNN model, we could result that DL models need a lot of more time to train in comparison to other models. RF that has similar accuracy rates needs almost 10 times less to conclude the process. Also, in the groups that the distribution is skewed DL have

lower accuracy from these trained with RF. RF was the classification algorithm that had high average A3R rate in all groups. SVM and linear regression followed while quadratic regression had the lowest average results.

In terms of pre-processing techniques and scoring, word embeddings of 1000 words exceed those of 2000 words both is terms of accuracy and run-time. However, could not overpass the combination of *tf-idf, stopwords, stemming and sparcity correction* in the *random forest* algorithm. Although CNN with 1000 words exceed in accuracy all other workflows, it was hold back as its score on A3R was lower because of the high run-time.

tf-idf gave also good predictions. Observing the results on the table **??**, we could conclude that *information gain* showed also promising results when applied. Another interesting point was that 0,98% *sparsity correction* achieved better results than 0,99% which was also better than none. *Stemming* and *Stopwords* removal seemed to have a positive impact in all algorithms but the CNN. This may results from the fact that these words are must have been removed by the CNN algorithm during the training. The results in case of CNN the workflow with no text preprocessing had almost equal results to those when these methods were applied.

Finally we can detect differences between the best performing workflows and the dataset groups. Group *g1* scored first in all classification algorithms. We may deduce that the even distribution of this dataset was the reason for the good performance of all classsifiers. Groups *g2,g3* & *g4* showed similar behaviour while the *g5, g6* came last ones. The relative low numbers of documents in comparison to the other datasets may be reason for this behaviour.

Table 3.7: Ranking of the first 20 workflows with the correspondent run-time, accuracy and A3R score.

rank	workF	repr	stop	stem	spar	info	algo	r-t	acc	A3R
1	w152	tf-idf	smart	porter	0,98	none	rf	2,62	0,71	0,70
2	w167	tf-idf	none	porter	0,98	>0	rf	2,58	0,71	0,69
3	w156	freq	smart	none	0,98	none	rf	1,63	0,69	0,69
4	w193	emb1	none	none	none	none	cnn	66,32	0,765	0,69
5	w187	tf-idf	default	none	0,98	>0	rf	1,61	0,70	0,69
6	w195	emb1	none	porter	none	none	cnn	75,32	0,76	0,68
7	w81	tf-idf	smart	porter	0,99	>0	rf	2,83	0,70	0,68
8	w197	emb1	smart	none	none	none	cnn	74,29	0,76	0,68
9	w202	emb2	none	none	none	none	cnn	105,67	0,76	0,67
10	w194	emb1	default	none	none	none	cnn	73,32	0,75	0,67
11	w198	emb1	default	porter	none	none	cnn	83,00	0,73	0,66
12	w200	emb2	none	porter	none	none	cnn	117,82	0,729	0,65
13	w204	emb2	default	porter	none	none	cnn	121,89	0,728	0,65
14	w196	emb1	smart	porter	none	none	cnn	88,56	0,72	0,65
15	w203	emb2	smart	porter	none	none	cnn	129,99	0,72	0,64
16	w199	emb2	default	none	none	none	cnn	132,37	0,72	0,63
17	w201	emb2	smart	none	none	none	cnn	128,79	0,72	0,63
18	w166	tf-idf	smart	porter	0,99	>0	rf	2,83	0,649	0,63
19	w148	tf-idf	none	none	0,98	>0	rf	2,74	0,65	0,63
20	w159	tf-idf	default	porter	0,98	none	rf	3,41	0,64	0,62

Chapter 4

Conclusions and Future Work

Revising the initial objectives of this thesis, we could assume that meta-learning approaches provide us with useful information about workflow selection and tuning parameters. A3R measure and AR technique helped us to explore different methods and order their results.

On the other hand, DL models proved to have high accuracy rates in dataset groups with low number of samples and unbalanced classes. Though their complexity may harden their implementation, the final outcome is satisfying. Classic algorithms proved to perform well and gave high results, that were correlated to the text-preprocessing. Feature selection and parameters tuning can be definitive in their case.

4.1 Limitations of this work

This work was subject to some limitations. First of all, the tuning of the parameters was manual. In case optimization methods were applied, the resulting ranking could be different. Another limitation was the efficiency of the computer device that we used. A large server would probably be more efficient in the collection of meta-data and probably would result on higher scores. Finally, the group that we used included large corpora that were evenly distributed, for the case of 4 out of 6 dataset groups. These conditions are usually not met in real world data. Therefore, we would expect different results in such a case.

4.2 Future Work

Based on the conclusions we ended, this study could be enhanced in the following aspects a future work:

- **Explore different architectures of DL models:** There is a vast bibliog-

raphy of DL models with different architecture such as Recursice, Recurrent, LSTM etc. Another popular technique is the training of the word embedding matrices based on large, cross-domain, knowledge graphs such as DBpedia, Freebase, OpenCyc, Wikidata, YAGO etc.

- **Tune the parameters based on optimization techniques:** Greedy search, Genetic algorithms are some of the optimization algorithms that could be explored in order to tune the parameters more efficiently.

- **Explore different values on the parameters of A3R:** In our work, we calculated our rankings based on a A3R measure with the P parameter equal to $1/40$. This parameter could be set on a range of values and explre its impact on the final rankings obtained.

- **Apply workflow selection based on meta-learning approaches on real-world domains:** The collection of the documents we used are articles were grammatical mistakes or misspelling are missing. The text in overall does contain characters or words that do not belong to english language. However, in reality, this is hardly the case. Training the workflows on text such as sms or tweets would result to different conclusions.

Bibliography

Abdulrahman, S. M., Cachada, M. V., and Brazdil, P. (2017). Impact of feature selection on average ranking method via metalearning. In *European Congress on Computational Methods in Applied Sciences and Engineering*, pages 1091–1101. Springer.

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

Agichtein, E. and Gravano, L. (2000). Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94. ACM.

Airoldi, E. and Malin, B. (2004). Data mining challenges for electronic safety: the case of fraudulent intent detection in e-mails. In *Proceedings of the workshop on privacy and security aspects of data mining*, pages 57–66.

Albert, M. H. and Atkinson, M. D. (2005). Simple permutations and pattern restricted permutations. *Discrete Mathematics*, 300(1-3):1–15.

Asghari, M. and Aghdam, M. S. (2010). Impact of salicylic acid on post-harvest physiology of horticultural crops. *Trends in Food Science & Technology*, 21(10):502–509.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

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

Brazdil, P. B., Soares, C., and Da Costa, J. P. (2003). Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277.

Caragea, C., Silvescu, A., and Tapia, A. H. (2016). Identifying informative messages in disaster events using convolutional neural networks. In *International Conference on Information Systems for Crisis Response and Management*, pages 137–147.

Chollet, F. et al. (2018). Keras: The python deep learning library. *Astrophysics Source Code Library*.

Cireşan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*.

Demuth, H. B., Beale, M. H., De Jess, O., and Hagan, M. T. (2014). *Neural network design*. Martin Hagan.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., et al. (1996). *Advances in knowledge discovery and data mining*, volume 21. AAAI press Menlo Park.

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

Fox, E. (2009). The role of reader characteristics in processing and learning from informational text. *Review of Educational Research*, 79(1):197–261.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2008). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Hammond, C. and Mitchell, E. (1997). Library instruction for the professions: Information needs and libraries. *Reference Services Review*, 25(2):79–87.

Harter, S. P. (1986). *Online information retrieval: concepts, principles, and techniques*. Academic Press Orlando, FL.

Jindal, N. and Liu, B. (2007). Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, pages 1189–1190. ACM.

Joachims, T. (1998a). Making large-scale svm learning practical. Technical report, Technical report, SFB 475: Komplexitätsreduktion in Multivariaten ....

Joachims, T. (1998b). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

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

Leite, R. and Brazdil, P. (2010). Active testing strategy to predict the best classification algorithm via sampling and metalearning. In *ECAI*, pages 309–314.

Lewis, D. (1997). Reuters-21578 dataset. *URL= http: www. research. att. com lewisreuters21578. html.*

Liu, X.-x., Zhou, L., and Du, X.-Y. (2005). A method of sensor management based on target priority and information gain. *Dianzi Xuebao(Acta Electronica Sinica)*, 33(9):1683–1687.

Lohr, S. (2019). Is there a smarter path to artificial intelligence? some experts hope so. Last visited 23-Aug-2019.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics*, 11(1-2):22–31.

Madabushi, H. T. and Lee, M. (2016). High accuracy rule-based question classification using question syntax and semantics. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1220–1230.

Maron, O. and Moore, A. W. (1997). The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11(1-5):193–225.

McCullough, M. E., Emmons, R. A., and Tsang, J.-A. (2002). The grateful disposition: A conceptual and empirical topography. *Journal of personality and social psychology*, 82(1):112.

Melville, P., Gryc, W., and Lawrence, R. D. (2009). Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1275–1284. ACM.

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

Mitchell, T. M. (1980). *The need for biases in learning generalizations*. Citeseer.

Montgomery, D. C., Peck, E. A., and Vining, G. G. (2012). *Introduction to linear regression analysis*, volume 821. John Wiley & Sons.

Nayak, A. (2016). *Race, place and globalization: Youth cultures in a changing world.* Bloomsbury Publishing.

Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Rennie, J. (2008). The 20 newsgroups data set.

Roitblat, H. L., Kershaw, A., and Oot, P. (2010). Document categorization in legal electronic discovery: computer classification vs. manual review. *Journal of the American Society for Information Science and Technology*, 61(1):70–80.

Rumelhart, D. E. and McClelland, J. L. (1986). On learning the past tenses of english verbs.

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. D., and Stamatopoulos, P. (2001). Stacking classifiers for anti-spam filtering of e-mail. *arXiv preprint cs/0106040*.

Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

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

Seber, G. A. and Lee, A. J. (2012). *Linear regression analysis*, volume 329. John Wiley & Sons.

Shwartz-Ziv, R. and Tishby, N. (2017). Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., and Chanona-Hernández, L. (2014). Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 41(3):853–860.

Socher, R. (2014). *Recursive deep learning for natural language processing and computer vision.* PhD thesis, Citeseer.

Squire, D. M., Müller, W., Müller, H., and Pun, T. (2000). Content-based query of image databases: inspirations from text retrieval. *Pattern Recognition Letters*, 21(13-14):1193–1198.

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Symeonidis, S., Effrosynidis, D., and Arampatzis, A. (2018). A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. *Expert Systems with Applications*, 110:298–310.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Van Rijn, J. N., Bischl, B., Torgo, L., Gao, B., Umaashankar, V., Fischer, S., Winter, P., Wiswedel, B., Berthold, M. R., and Vanschoren, J. (2013). Openml: A collaborative science platform. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 645–649. Springer.

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

Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Icml*, volume 97, page 35.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., and Xu, B. (2016). Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.

Appendices

Appendix A

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

Computer Specifications	
Processor	Intel Core i7-4712HQ CPU @ 2.30GHz x 8
RAM	15,6 GiB
Graphic Card	GeForce GT 750M/PCIe/SSE2
Disk	305,1 GB

Table A.2: Final workflow rankings based on A3R.

rank	**workflow**	**repr**	**stop**	**stem**	**spar**	**info**	**algo**
1	w152	tf-idf	smart	porter	0,98	none	rf
2	w167	tf-idf	none	porter	0,98	>0	rf
3	w156	freq	smart	none	0,98	none	rf
4	w193	emb1	none	none	none	none	cnn
5	w187	tf-idf	default	none	0,98	>0	rf
6	w195	emb1	none	porter	none	none	cnn
7	w81	tf-idf	smart	porter	0,99	>0	rf
8	w197	emb1	smart	none	none	none	cnn
9	w202	emb2	none	none	none	none	cnn
10	w194	emb1	default	none	none	none	cnn
11	w198	emb1	default	porter	none	none	cnn
12	w200	emb2	none	porter	none	none	cnn
13	w204	emb2	default	porter	none	none	cnn
14	w196	emb1	smart	porter	none	none	cnn
15	w203	emb2	smart	porter	none	none	cnn
16	w199	emb2	default	none	none	none	cnn

Continued on next page

Table A.2 – continued from previous page

rank	workflow	repr	stop	stem	spar	info	algo
17	w201	emb2	smart	none	none	none	cnn
18	w166	tf-idf	smart	porter	0,99	>0	rf
19	w148	tf-idf	none	none	0,98	>0	rf
20	w159	tf-idf	default	porter	0,98	none	rf
21	w163	tf-idf	smart	none	0,98	none	rf
22	w170	freq	none	none	0,99	>0	rf
23	w177	freq	default	none	0,99	none	rf
24	w180	freq	default	none	0,98	>0	rf
25	w160	tf-idf	default	porter	0,98	>0	rf
26	w190	freq	smart	porter	0,99	>0	rf
27	w183	freq	default	porter	0,98	none	rf
28	w184	freq	default	porter	0,98	>0	rf
29	w149	tf-idf	none	porter	0,99	none	rf
30	w153	tf-idf	default	none	0,99	none	rf
31	w173	freq	none	porter	0,99	none	rf
32	w161	tf-idf	smart	none	0,99	none	rf
33	w185	freq	smart	none	0,99	none	rf
34	w171	freq	none	none	0,98	none	rf
35	w120	tf-idf	smart	porter	0,98	>0	svm
36	w100	tf-idf	none	none	0,98	>0	svm
37	w145	tf-idf	none	none	0,99	none	rf
38	w117	tf-idf	smart	porter	0,99	none	svm
39	w172	freq	none	none	0,98	>0	rf
40	w146	tf-idf	none	none	0,99	>0	rf
41	w147	tf-idf	none	none	0,98	none	rf
42	w175	freq	none	porter	0,98	none	rf
43	w86	tf-idf	none	porter	0,99	>0	rf
44	w188	freq	smart	none	0,98	>0	rf
45	w189	freq	smart	porter	0,99	none	rf
46	w174	freq	none	porter	0,99	>0	rf
47	w141	freq	smart	porter	0,99	none	svm
48	w176	freq	none	porter	0,98	>0	rf
49	w186	freq	smart	none	0,99	>0	rf
50	w178	freq	default	none	0,99	>0	rf
51	w137	freq	smart	none	0,99	none	svm
52	w138	freq	smart	none	0,99	>0	svm
53	w181	freq	default	porter	0,99	none	rf
54	w182	freq	default	porter	0,99	>0	rf
55	w126	freq	none	porter	0,99	>0	svm

Continued on next page

Table A.2 – continued from previous page

rank	workflow	repr	stop	stem	spar	info	algo
56	w168	tf-idf	smart	porter	0,98	>0	rf
57	w191	freq	smart	porter	0,98	none	rf
58	w179	freq	default	none	0,98	none	rf
59	w192	freq	smart	porter	0,98	>0	rf
60	w151	tf-idf	none	porter	0,98	none	rf
61	w164	tf-idf	smart	none	0,98	>0	rf
62	w165	tf-idf	smart	porter	0,99	none	rf
63	w154	tf-idf	default	none	0,99	>0	rf
64	w169	freq	none	none	0,99	none	rf
65	w150	tf-idf	none	porter	0,99	>0	rf
66	w157	tf-idf	default	porter	0,99	none	rf
67	w158	tf-idf	default	porter	0,99	>0	rf
68	w162	tf-idf	smart	none	0,99	>0	rf
69	w136	freq	default	porter	0,98	>0	svm
70	w155	tf-idf	default	none	0,98	none	rf
71	w72	tf-idf	smart	porter	0,98	>0	qda
72	w107	tf-idf	default	none	0,98	none	svm
73	w24	tf-idf	smart	porter	0,98	>0	lda
74	w105	tf-idf	default	none	0,99	none	svm
75	w21	tf-idf	smart	porter	0,99	none	lda
76	w127	freq	none	porter	0,98	none	svm
77	w108	tf-idf	default	none	0,98	>0	svm
78	w23	tf-idf	smart	porter	0,98	none	lda
79	w112	tf-idf	default	porter	0,98	>0	svm
80	w111	tf-idf	default	porter	0,98	none	svm
81	w71	tf-idf	smart	porter	0,98	none	qda
82	w119	tf-idf	smart	porter	0,98	none	svm
83	w63	tf-idf	default	porter	0,98	none	qda
84	w106	tf-idf	default	none	0,99	>0	svm
85	w60	tf-idf	default	none	0,98	>0	qda
86	w97	tf-idf	none	none	0,99	none	svm
87	w67	tf-idf	smart	none	0,98	none	qda
88	w19	tf-idf	smart	none	0,98	none	lda
89	w109	tf-idf	default	porter	0,99	none	svm
90	w110	tf-idf	default	porter	0,99	>0	svm
91	w129	freq	default	none	0,99	none	svm
92	w22	tf-idf	smart	porter	0,99	>0	lda
93	w115	tf-idf	smart	none	0,98	none	svm
94	w70	tf-idf	smart	porter	0,99	>0	qda

Continued on next page

Table A.2 – continued from previous page

rank	workflow	repr	stop	stem	spar	info	algo
95	w114	tf-idf	smart	none	0,99	>0	svm
96	w68	tf-idf	smart	none	0,98	>0	qda
97	w59	tf-idf	default	none	0,98	none	qda
98	w121	freq	none	none	0,99	none	svm
99	w64	tf-idf	default	porter	0,98	>0	qda
100	w65	tf-idf	smart	none	0,99	none	qda
101	w116	tf-idf	smart	none	0,98	>0	svm
102	w128	freq	none	porter	0,98	>0	svm
103	w102	tf-idf	none	porter	0,99	>0	svm
104	w130	freq	default	none	0,99	>0	svm
105	w101	tf-idf	none	porter	0,99	none	svm
106	w18	tf-idf	smart	none	0,99	>0	lda
107	w17	tf-idf	smart	none	0,99	none	lda
108	w62	tf-idf	default	porter	0,99	>0	qda
109	w122	freq	none	none	0,99	>0	svm
110	w144	freq	smart	porter	0,98	>0	svm
111	w125	freq	none	porter	0,99	none	svm
112	w20	tf-idf	smart	none	0,98	>0	lda
113	w61	tf-idf	default	porter	0,99	none	qda
114	w113	tf-idf	smart	none	0,99	none	svm
115	w66	tf-idf	smart	none	0,99	>0	qda
116	w69	tf-idf	smart	porter	0,99	none	qda
117	w118	tf-idf	smart	porter	0,99	>0	svm
118	w133	freq	default	porter	0,99	none	svm
119	w132	freq	default	none	0,98	>0	svm
120	w143	freq	smart	porter	0,98	none	svm
121	w123	freq	none	none	0,98	none	svm
122	w42	freq	smart	none	0,99	>0	lda
123	w131	freq	default	none	0,98	none	svm
124	w103	tf-idf	none	porter	0,98	none	svm
125	w104	tf-idf	none	porter	0,98	>0	svm
126	w53	tf-idf	none	porter	0,99	none	qda
127	w139	freq	smart	none	0,98	none	svm
128	w134	freq	default	porter	0,99	>0	svm
129	w135	freq	default	porter	0,98	none	svm
130	w124	freq	none	none	0,98	>0	svm
131	w47	freq	smart	porter	0,98	none	lda
132	w142	freq	smart	porter	0,99	>0	svm
133	w140	freq	smart	none	0,98	>0	svm

Continued on next page

Table A.2 – continued from previous page

rank	workflow	repr	stop	stem	spar	info	algo
134	w99	tf-idf	none	none	0,98	none	svm
135	w98	tf-idf	none	none	0,99	>0	svm
136	w33	freq	default	none	0,99	none	lda
137	w15	tf-idf	default	porter	0,98	none	lda
138	w90	freq	smart	none	0,99	>0	qda
139	w57	tf-idf	default	none	0,99	none	qda
140	w32	freq	none	porter	0,98	>0	lda
141	w34	freq	default	none	0,99	>0	lda
142	w40	freq	default	porter	0,98	>0	lda
143	w41	freq	smart	none	0,99	none	lda
144	w31	freq	none	porter	0,98	none	lda
145	w56	tf-idf	none	porter	0,98	>0	qda
146	w79	freq	none	porter	0,98	none	qda
147	w14	tf-idf	default	porter	0,99	>0	lda
148	w8	tf-idf	none	porter	0,98	>0	lda
149	w58	tf-idf	default	none	0,99	>0	qda
150	w25	freq	none	none	0,99	none	lda
151	w26	freq	none	none	0,99	>0	lda
152	w27	freq	none	none	0,98	none	lda
153	w89	freq	smart	none	0,99	none	qda
154	w55	tf-idf	none	porter	0,98	none	qda
155	w16	tf-idf	default	porter	0,98	>0	lda
156	w7	tf-idf	none	porter	0,98	none	lda
157	w39	freq	default	porter	0,98	none	lda
158	w91	freq	smart	none	0,98	none	qda
159	w43	freq	smart	none	0,98	none	lda
160	w36	freq	default	none	0,98	>0	lda
161	w44	freq	smart	none	0,98	>0	lda
162	w5	tf-idf	none	porter	0,99	none	lda
163	w30	freq	none	porter	0,99	>0	lda
164	w4	tf-idf	none	none	0,98	>0	lda
165	w77	freq	none	porter	0,99	none	qda
166	w45	freq	smart	porter	0,99	none	lda
167	w48	freq	smart	porter	0,98	>0	lda
168	w49	tf-idf	none	none	0,99	none	qda
169	w83	freq	default	none	0,98	none	qda
170	w50	tf-idf	none	none	0,99	>0	qda
171	w37	freq	default	porter	0,99	none	lda
172	w38	freq	default	porter	0,99	>0	lda

Continued on next page

Table A.2 – continued from previous page

rank	workflow	repr	stop	stem	spar	info	algo
173	w78	freq	none	porter	0,99	>0	qda
174	w11	tf-idf	default	none	0,98	none	lda
175	w9	tf-idf	default	none	0,99	none	lda
176	w46	freq	smart	porter	0,99	>0	lda
177	w51	tf-idf	none	none	0,98	none	qda
178	w73	freq	none	none	0,99	none	qda
179	w13	tf-idf	default	porter	0,99	none	lda
180	w52	tf-idf	none	none	0,98	>0	qda
181	w10	tf-idf	default	none	0,99	>0	lda
182	w82	freq	default	none	0,99	>0	qda
183	w12	tf-idf	default	none	0,98	>0	lda
184	w84	freq	default	none	0,98	>0	qda
185	w88	freq	default	porter	0,98	>0	qda
186	w75	freq	none	none	0,98	none	qda
187	w54	tf-idf	none	porter	0,99	>0	qda
188	w3	tf-idf	none	none	0,98	none	lda
189	w1	tf-idf	none	none	0,99	none	lda
190	w35	freq	default	none	0,98	none	lda
191	w85	freq	default	porter	099	none	qda
192	w76	freq	none	none	0,98	>0	qda
193	w29	freq	none	porter	0,99	none	lda
194	w92	freq	smart	none	0,98	>0	qda
195	w74	freq	none	none	0,99	>0	qda
196	w95	freq	smart	porter	0,98	none	qda
197	w94	freq	smart	porter	0,99	>0	qda
198	w2	tf-idf	none	none	0,99	>0	lda
199	w6	tf-idf	none	porter	0,99	>0	lda
200	w80	freq	none	porter	0,98	>0	qda
201	w28	freq	none	none	0,98	>0	lda
202	w96	freq	smart	porter	0,98	>0	qda
203	w93	freq	smart	porter	0,99	none	qda
204	w87	freq	default	porter	0,98	none	qda