SAMUEL QUEIROZ SOUZA ROCHA

Advisor: Joubert de Castro Lima

ANALYSIS AND CLASSIFICATION OF TEXTS FOR INDUSTRIAL MACHINES MAINTENANCE

Ouro Preto

December 2018

Federal University of Ouro Preto
Institute of Exact Sciences and Biology
Undergraduate Program in Computer Science

ANALYSIS AND CLASSIFICATION OF TEXTS FOR INDUSTRIAL MACHINES MAINTENANCE

Monograph presented to the Undergraduate Program in Computer Science of the Federal University of Ouro Preto in partial fulfillment of the requirements for the degree of Bachelor in Computer Science.

SAMUEL QUEIROZ SOUZA ROCHA

Ouro Preto
December 2018

R582a Rocha, Samuel.
Analysis and classification of texts for industrial machines maintenance [manuscrito] / Samuel Rocha. - 2018.

39f.: il.: color; grafs; tabs.

Orientador: Prof. Dr. Joubert Lima.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Departamento de Computação.

1. Aprendizado do computador. 2. Mineração de dados (Computação). I. Lima, Joubert. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 004.85

Catalogação: ficha.sisbin@ufop.edu.br

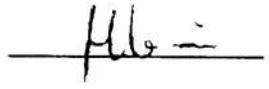
FEDERAL UNIVERSITY OF OURO PRETO

CERTIFICATE OF APPROVAL

Analysis and Classification of Texts for Industrial Machines Maintenance

SAMUEL QUEIROZ SOUZA ROCHA

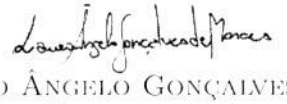
Monograph defended and approved by the examination board composed by:



Dr. JOUBERT DE CASTRO LIMA Advisor
Federal University of Ouro Preto

M.Sc. REINALDO SILVA FORTES
Federal University of Ouro Preto

Dr. RODRIGO ROCHA SILVA
Faculdade de Tecnologia do Estado de São Paulo



M.Sc. LAURO ÂNGELO GONÇALVES DE MORAES
Federal University of Ouro Preto

Ouro Preto, Dezembro 2018

Resumo

Análise e classificação de textos são importantes tarefas em mineração de dados. Através de tais tarefas conseguimos, a partir de dados brutos, extrair ou gerar informações relevantes que podem ser usadas para resolver problemas ou melhorar soluções já existentes. Este trabalho aborda uma solução para o problema em que dado uma descrição textual para um erro de uma máquina industrial, o classificador deve atribuir um código de erro para a tal descrição. Uma solução computacional que resolva tal problema pode contribuir para agilização no processo de manutenção, diminuindo o tempo de inatividade das máquinas e reduzindo o custo dos erros de alocação de técnicos. Neste trabalho, nós testamos três métodos de aprendizado de máquina com uma base de dados de históricos textuais em inglês que contém as descrições de erro, e as técnicas são: Support Vector Machine (SVM), Random Forest e FastText. Uma importante inovação deste trabalho é um caso de estudo para a literatura na área de aprendizado de máquina, no qual o número de classes a serem identificadas é alta, passando de 1200 diferentes classes. Em contrapartida, o estado da arte em classificação de textos descreve casos de estudo com apenas dezenas de classes. Os resultados usando uma estratégia de 5-fold cross validation e IRace para configurar os parâmetros de entrada dos três classificadores alcançaram altos f1-score para quase todos os métodos avaliados, com os resultados variando precisamente entre 67,5 e 94,92% em um conjunto de dados composto por mais de 800 mil descrições textuais curtas. O FastText provou ser mais rápido que os outros, precisamente 4 vezes mais rápido que o SVM e 1,3 vezes mais rápido que o Random Forest.

Palavras-chave: Aprendizado de Máquina, Classificação de Texto, Mineração de Dados, Manutenção de Máquinas Industriais.

Abstract

Analysis and classification of texts are important tasks in data mining. Through such tasks, we are able to extract or generate relevant information from raw data that can be used to solve problems or improve existing solutions. This work addresses a solution for the problem of given a textual description for an industrial machine error, the classifier must assign a fault code for such a description. A computational solution that attenuates such a problem can contribute to streamlining the maintenance process, reducing the machine utilization downtime and reducing the cost of technical allocation errors. In this work, we tested three machine learning methods with a historical English text dataset with error descriptions and they are: Support Vector Machine (SVM), Random Forest and FastText. One important innovation of this work is a case study for the machine learning literature where the number of classes to be identified is high, reaching more than 1200 different classes. In contrast, the state-of-art in text classification described case studies with only tens of classes. The results using a 5-fold cross validation strategy and IRace for configuring the three classifiers input parameters achieved high f1-score for almost all evaluated methods, precisely the results varied from 67,5 to 94,92% in a dataset composed of more than 800k short text descriptions. The FastText proved to be faster than the others, precisely 4 times faster than SVM and 1,3 times faster than Random Forest.

Keywords: Machine Learning, Text Classification, Data Mining, Industrial Machines Maintenance.

I dedicate this work to my family, for believing and investing in me, supporting me at all moments.

Acknowledgments

I thank God first for enlightening my path, my family for all support and incentive, and my advisor, Dr. Joubert de Castro Lima, who made this work possible with all the help, collaboration and opportunity provided to me. I also thank DECOM and UFOP, with all the teachers and staff who contributed to my education.

Contents

1 Introduction **1**
1.1 Goal . . . 2
1.1.1 Specific Goals . . . 2
1.2 Hypothesis . . . 3
1.3 Work Organization . . . 3

2 Related Work **4**

3 Development **11**
3.1 Data Characterization and Problem Formulation . . . 11
3.2 Training Models . . . 13
3.3 Text Classification Pipeline . . . 13
3.3.1 Pre-processing . . . 14
3.3.2 Classification . . . 15
3.3.3 Output Analysis . . . 17
3.4 Scope Reductions . . . 17

4 Experiments **19**
4.1 Setup . . . 20
4.2 Individual Results . . . 20

5 Conclusion **23**

Bibliography **24**

List of Figures

3.1 Frequency Distribution of Codes 12
3.2 Frequency Distribution of Chapters 12
3.3 Steps of a classification pipeline 14
3.4 Language Distribution 18

4.1 Results of the Experiments 20

List of Tables

2.1 Comparison of the related work using the previously defined requirements. 10

4.1 FastText Tuning Process. 21
4.2 SVM Tuning Process. 21
4.3 Random Forest Tuning Process. 22
4.4 Results of the Experiments. 22

Chapter 1

Introduction

In recent years, the data generation is increasing quickly and there are many factors that explain this, including the huge growth of social media usage in the last decade Stieglitz et al. (2018) and the generation of enterprise data, for instance for the latter, we have companies like Amazon and Walmart that process millions of operations every day Chen et al. (2014). The point at which capacities and infrastructures are overcomed by information is being reached in enterprises Chen et al. (2014). Meanwhile, technologies with great potential of data generation, such as IoT, are being explored and developed Habibzadeh et al. (2018). Among many data types (e.g., graph, stream, video, audio, spatial, image, text and alphanumeric), the text data type plays an important contribution to what we call Big Data. Consequently, there is a huge interest in investigating new methods and tools for analyzing textual data Joulin et al. (2016).

One technique for textual data analysis is the classification, a supervised data mining technique that involves assigning a label to a set of unlabeled input objects. Based on the number of classes present, there are two types of classification: i) Binary classification – classify input objects into one of the two classes; and ii) Multi-class classification – classify input objects into one of the multiple classes. Precisely in text classification, the unlabeled input objects are texts. Typically, text classification problems involve Multi-class classification Sriram et al. (2010).

For a classifier to learn how to classify the texts, it needs some kind of truth. For this purpose, the input objects are partitioned into training and testing data. Training data are those where the texts are already labeled. Testing data are those where the texts are unlabeled. The goal is to learn the knowledge from the already labeled training data and apply this on the testing data and predict the class label for the test dataset accurately. Sometimes, the business domain requirements include as fast as possible, online or real time demands, thus the solution can become even more complex. There are many examples of different business domains, representing very useful case studies using text classification: (i) hate speech detection Badjatiya et al. (2017); (ii) classify a resume in different areas Sayfullina et al. (2017); (iii) fake news detection Bajaj (2017); (iv) user comment moderation Pavlopoulos

et al. (2017); (v) classify movie reviews as positive or negative Pang et al. (2002); (vi) classify a customer review as positive or negative Kim (2014) and many more. Due to so many demands for text classification solutions, numerous techniques and computational methods were presented to realize the classification automatically. Among them, there is a group called machine learning that has achieved good results, in which we have linear classifiers as FastText Joulin et al. (2016), Support Vector Machines (SVM) Pang et al. (2002), Random Forests Breiman (2001), use of neural networks with few layers Mikolov et al. (2013a) and even the use of deep learning Kim (2014).

In this work, we followed a pipeline (described in detail in Chapter 3) containing three major steps: (i) pre-processing, (ii) classification and (iii) output analysis. Some scope reductions were necessary in the pre-processing and classification steps due to our lack of time and infrastructure to perform some tests. The scope reductions were: (i) the execution of the first solution using three methods to solve the presented problem, precisely FastText, SVM and Random Forest; (ii) utilization of English texts of the dataset, since it has more than 25 languages identified using an open source project called Language Detection from Google Code (https://code.google.com/archive/p/language-detection/); (iii) Automatically tuning of the classifiers hyper-parameters using a partition of the dataset, precisely 20% of it.

The FastText is a library that provides solutions for text representation and classification, and in this work we only use the classifier. The FastText and SVM classifiers achieved good results being 90,27% and 94,92% of f1-score respectively, while the Random Forest classifier did not obtain a high f1-score (67,5%), however some limitations, explained in detail in Section 4.2, had to be made to make its execution feasible.

1.1 Goal

Given the advances obtained with the text classification, the goal of this work is to present the best text classification method, among the tested methods, for the industrial machines maintenance problem. Precisely, from a descriptive text collection of the machine owner, as well as technical opinions of mechanics, electricians and other specialists about the error type, we must present the best text classification methods in terms of f1-score in classifying industrial machines error codes. The classification runtime is also fundamental, what means the test phase of a classifier must be done as fast as possible, but the training phase can require vast amount of time to be initially done.

1.1.1 Specific Goals

1. Make a collection of case studies about the use of machine learning methods for text classification, their limits and strengths, which can be useful for future technical and research investigations;

2. Perform a pre-processing of data;
3. Perform a hyper-parameters tuning to improve the results;
4. Perform a 5-fold cross-validation to ensure greater reliability of results;
5. Use three machine learning methods to conduct all experimental evaluations;
6. Perform a set of analysis of the results obtained in the experiments. Repeat steps 3, 4 and 5 for each new method tested.

1.2 Hypothesis

Our hypothesis is that there is a certain machine learning method that ensure the best classification scores in terms of runtime and accuracy while analyzing a vast collection of texts about industrial machines maintenance. The equipment maintenance industry can consider this work as an alternative solution that can learn from errors, as well as the capacity of classifying thousands of inputs per minute, something that can reduce the machines downtime, costs with erroneous technician allocations, and the undue charges to customers that often foul the company image. The improvements in textual description may also occur once everyone realizes efficiency gains.

1.3 Work Organization

The rest of the work is organized as follows: In Chapter 2, we present the related works that are machine learning case studies, precisely classification methods that adopt only texts, thus very similar to our case study about industrial machines maintenance. In Chapter 3, we describe in more detail the problem and present the text classification pipeline used to conduct this work. In Chapter 4, we present the solutions to solve the classification problem stated previously, as well as the results of using a 5-fold cross-validation strategy. Finally, in the last chapter there is a conclusion of the work.

Chapter 2

Related Work

In this chapter, there is a description of related works that also investigated machine learning methods for text classification. The related work was evaluated or classified according to the following requirements and these requirements are from the related work Kim (2014), Pavlopoulos et al. (2017):

1. Number of Techniques (**NT**): indicates how many techniques were explored in the work. Results from other papers were not counted;
2. Number of Classes (**NC**): indicates how many classes the classifier must learn. It interferes in the difficulty of the problem by increasing the training space;
3. Multi Language (**ML**): indicates whether the work explored datasets in different languages. It interferes in the difficulty of the problem by increasing the feature space;
4. Data Size (**DS**): indicates the size of dataset used in the work in terms of registers. A large dataset allows us simulate a real world processing time;
5. Pre-trained Vectors (**PV**): indicates whether the work used pre-trained word vectors to improve the results accuracy;
6. Parameter Tuning (**PT**): indicates whether the work performed a parameter tuning for the classifiers;
7. Data Partition (**DP**): indicates the method used to partition the data into training set and test set.

In the work of Pang et al. (2002) it was evaluated three machine learning methods for movie reviews classification according to their sentiments, which may be a positive or negative criticism. The baselines were results obtained from simple decision procedures with two word sets, one with positive words and the second with the negative ones. Those bag of words

were created by a graduate student who listed the words that they thought likely to appear in positive and negative reviews. The baselines results were 58% with a tie rate of 75%, and 64% with a tie rate of 39%. The tie rate represents the documents that the algorithm indicated an equal probability of belonging to both classes (positive and negative). The dataset used on tests was taken from the Internet Movie Database (IMDb). The data was partitioned into three equal folds for cross-validation. The methods used were Naive Bayes Lewis (1998) (best result: 81,5% of accuracy), Maximum Entropy Berger et al. (1996) (best result: 81% of accuracy) and Support Vector Machines (SVMs) Joachims (1998a) (best result: 82,9% of accuracy). All the machine learning techniques presented very good results, above the baselines. In terms of performance, the Naive Bayes method was the worst and the SVM the best, although the differences were not very large.

In the work of Pang and Lee (2004) it was performed a two-step sentiment analysis on movie reviews. First, the subjective sentences inside review are identify and then they are used as standard classifiers inputs for a polarity classification. To identify the subjective sentences was proposed a graph-based formulation relying on finding minimum cuts inspired by Blum and Chawla (2001), in which each sentence represents a vertex, the subjective class represents a vertex and the objective class represents another vertex. The edges between sentence vertices receive as weight an estimate of how important is that the sentences belong to the same class, while the edges between a sentence vertex and a class vertex receive as weight an estimate of preference of the sentence vertex belonging to class vertex. The cut is a partition on the graph into two sets where the first has to contain at least the subjective vertex and the second has to contain at least the objective vertex. As standard classifiers were used SVM and Naive Bayes. The SVM performed better on the full text (87,15% accuracy), in other words, without the subjective sentences identification (86,4% with subjective detector); however, the Naive Bayes method performed better with the subjective sentences detector (86,4% against 82,8%). The results were obtained through a 10-fold cross-validation strategy. The dataset used on tests contains 1K positive and 1K negative reviews and it is available on-line.

In the work Go et al. (2009) it was compared the performance among Naive Bayes, SVM and Maximum Entropy methods against a baseline method for sentiment classification of Twitter messages (tweets). There are many topics being discussed on Twitter, because of that it is very tough to label the data manually. To attenuate this problem the authors collected to train the tweets that contain emoticons. The ":)" emoticon represents the positive tweets, while ":(" represents the negative one. The emoticons were used to label the data, however they were stripped out from training data because they caused an accuracy decrease on Maximum Entropy and SVM classifiers. Tweets with both emoticons were not used. It is very common the users use their usernames in tweets, so all the usernames were replaced by a token USERNAME. The URLs also are very common and they were replaced by token URL. In a word that a letter repeats more than two times the repeated letter is replaced by two

occurrences. These three simple pre-processing activities reduced the feature set in 45,85% of its original size. The training data contains 800K positive and 800K negatives tweets, while the test data contains 182 positive and 177 negative tweets. The test data was collected and labeled manually. The authors tested the classifiers using unigram, bigram, unigram + bigram and unigram + part of speech (POS) tags. The best result using unigram was achieved by SVM (82,2% accuracy), using bigram the Naive Bayes performed better (81,6%), using unigram + bigram Maximum Entropy was the best (83%) and using unigram + POS SVM was the best again (81,9).

In Glorot et al. (2011b) it was performed a two-step procedure on domain adaptation for sentiment analysis. The first step consists on extracting features and then applying a linear classifier. The authors used an unsupervised method to extract high-level features from the text reviews of all the available domains, the Stacked Denoising Auto-encoder (SDA) Vincent et al. (2008) with rectifier units for the code layer Glorot et al. (2011a). As linear classifier, the SVM was selected due to its good results in sentiment analysis Pang et al. (2002). To compare the results the authors selected the adapted Structural Correspondence Learning (SCL) from Blitzer et al. (2007), the Multi-label Consensus Training (MCT) from Li and Zong (2008), the Spectral Feature Alignment (SFA) from Pan et al. (2010) and a baseline. The baseline is a SVM classifier trained on the raw data whereas the SVM used on the proposed method was trained on the high-level features extracted from SDA. The dataset used contains more than 340K reviews regarding 22 different product types from Amazon labeled as either positive or negative. The authors created a smaller and more controlled version of dataset selecting only 4 domains being 1K positive and 1K negatives reviews for each domain. The dataset was divided into 80% for training and 20% for test. For the 4 domains we have 12 possibilities of domain adaptation, and the proposed method outperformed the other methods in almost all experimental scenarios. In the adaptation of Kitchen Appliances to Electronics, the SCL achieved the best result (around -2%). The results was calculated by error of domain adaptation. The best result for the proposed method was in the adaptation of Electronics to Kitchen Appliances (around -2%), the MCT achieved the best result in the adaptation of Electronics to Kitchen Appliances (around 1,5%), the SVA performed better in the adaptation of Kitchen Appliances to Electronics (around -0,5%) while the baseline achieved the best result in the adaptation of Electronics to Kitchen Appliances (around 1%). The negative results mean that the classifier trained on a different domain can outperform the classifier trained on the target domain.

In the work Kim (2014), it was compared the performance of four convolutional neural network (CNN) models Collobert et al. (2011) against 7 real datasets. Among the four CNN models, three of them used pre-trained word vectors obtained from an unsupervised neural language model Mikolov et al. (2013b). The first CNN model used random word vectors that were updated during the training, the second model used pre-trained word vectors that were

kept static during the training, the third model used pre-trained word vectors but updating the vectors during the training and the latter used two sets of pre-trained word vectors, one maintaining the vectors static and the other updating the vectors during the training. Words not present in the set of pre-trained words were initialized randomly. The authors tested the models with 7 datasets: MR from Pang and Lee (2005), SST-1 from Socher et al. (2013), SST-2, Subj from Pang and Lee (2004), TREC from Li and Roth (2002), CR from Hu and Liu (2004), and MPQA from Wiebe et al. (2005). MR is a movie review dataset labeled as positive or negative and the third CNN model achieved the best result with 81,5% accuracy. SST-1 is also a movie review dataset labeled as very positive, positive, neutral, negative or very negative. The CNN models did not achieve better results than the state-of-art method result in this dataset - the best was the Paragraph-Vec from Le and Mikolov (2014), achieving 48,7% of accuracy. SST-2 is same as SST-1, but with neutral reviews removed and binary labels and the fourth CNN model outperforms the literature, achieving 88,1% of accuracy. Subj is a subjectivity dataset containing subjective or objective sentences. The CNN models did not achieve better results than the best methods in this dataset, being the best both the MNB from Wang and Manning (2012) and F-Dropout from Wang and Manning (2013) with 93,6% of accuracy. TREC is a question dataset labeled in 6 types and the best result was obtained with SVM from Silva et al. (2011), achieving 95% of accuracy. CR is a customer reviews dataset labeled as positive or negative and the fourth CNN model achieved the best result with 85% of accuracy. MPQA is a opinion polarity detection dataset and the second model was the best in this case with 89,6% of accuracy. This work gives a better understanding of how diverse the classification method results can be, reinforcing the importance of many case studies in the literature.

In Badjatiya et al. (2017), it was evaluated three machine learning methods to detect hate speech in tweets. It was used as baseline three embedding methods: Char n-grams, TF-IDF and BoWV (GloVe) Pennington et al. (2014), and three classifiers: Logistic Regression, Balanced SVM and Gradient Boosted Decision Trees (GBDT). The proposed methods were: CNN Kim (2014), Long-Term Short Term Memory (LSTM) Hochreiter and Schmidhuber (1997) and FastText Joulin et al. (2016), they were used along with Random Embedding and GloVe. The experiments were performed with a 16K tweets dataset labeled as sexist, racist or neither. In the dataset, 3383 registers were labeled as sexist, 1972 as racist and the remainder labeled as none. In the results, it was shown that the proposed methods obtained better results compared to the baseline methods. Among the proposed methods, CNN achieved the best result with 83,9% of F1, followed by FastText with 82,9% and LSTM with 80,8%. The results were calculated by a 10-fold cross-validation.

The paper Sayfullina et al. (2017) presented a CNN model to solve domain adaptation for summary classification. The goal was to classify the summaries into 27 areas of expertise. According to the work, because summaries are sensitive data, obtaining a large dataset is a

tough and costly task, however job descriptions are abundant and easy to obtain. Since both job description and expertise area summary are directly related, it is possible use such job descriptions as training set. Around 80K job descriptions labeled in the 27 areas of expertise were used as training set and 5K job descriptions, 523 anonymous summaries and 98 child-written summaries for dream jobs were used for validation. The authors point out that child-written summaries are more focused on the emotional and less on the skills, describing their interests, and precisely because of the lack of some information they become an interesting set for testing the abstraction of the machine learning method used. The FastText classifier Joulin et al. (2016) was used as baseline. The results showed that the CNN method achieved better results in the three validation sets (job description: 74,88% against 71,99%; summary: 40,15% against 33,4%; children's dream job: 51,02% against 28,5%), obtaining the biggest difference in the child-written summary analysis.

In the work Bajaj (2017), it was compared the performance between 8 text classification methods for fake news detection. According to the work, the spread of fake news is a growing threat to society, often being used for commercial purposes, such as attracting people to generate advertising revenue, however there is a significant increase in fake news to influence events and politics. In the work, 63K articles were collected, of which 13K were fake news and 50K authentic articles, both came from the public domain. The dataset was shuffled and separated into 60% for training, 20% for validation and 20% for test. The validation data were used to make the hyper-parameters selection, while the test data were used to evaluate the performance of the models. The models used were: Logistic Regression, Two-layer Feedforward Neural Network, Recurrent Neural Networks (RNN) Jozefowicz et al. (2015), LSTM, Gated Recurrent Units Chung et al. (2014), bidirectional RNN with LSTM, CNN with Max Pooling and Attention-Augmented CNN. The results showed that the Attention-Augmented CNN achieved the best precision (97%), while a Gated Recurrent Units achieved the best recall (79%) and F1 (84%).

In Pavlopoulos et al. (2017), it was compared the performance between RNN, CNN, DETOX Wulczyn et al. (2017) (previous state-of-art) and a baseline called LIST in comment moderation. In order to obtain feedback and loyalty, news portals and blogs allow their readers to interact with comments, but sometimes these comments can be abusive, causing problems. Moderators are employed in order to filter the comments but they often are overwhelmed by the volume of comments. An alternative is to use a semi or fully automatic moderation. They experimented with a dataset of 1.6M manually moderated user comments from a Greek sports portal and with a datasets with English Wikipedia comments labeled for personal attacks, aggression and toxicity Wulczyn et al. (2017). They proposed the use 5 attention mechanisms in conjunction with RNN to improve further the performance. The RNN (98,42 of AUC) was always better than CNN (97,86), and the LIST baseline was always the worst (93,95). Between CNN and DETOX (97,13) there is no clear winner. The a-RNN

was the best attention mechanism, performing always better than RNN on sports portal comments, but not always on Wikipedia comments. All methods performed better with Wikipedia datasets than sports portal. AUC means area under ROC curve.

The work Park and Fung (2017) compared a one-step approach with a two-step approach to classify abusive language. The one-step approach uses multi-class classifiers to detect sexist, racist or none labels. The two-step approach uses two binary classifiers, the first classifies if the text is abusive or not, and the second, among abusive texts, classifies if the text is sexist or racist. They used two English Twitter datasets from Waseem and Hovy (2016) and Waseem (2016) that were merged. The dataset contains in total 18350 tweets being 12427 of none, 2059 of racism and 3864 of sexism. They propose the use of three CNN models: CharCNN Zhang et al. (2015), WordCNN Kim (2014) and HybridCNN. The difference among the methods is that the first receives characters as input features, the second, words and the last can receive both. As baselines were used n-gram logistic regression (LR) Waseem and Hovy (2016), SVM and FastText. Their new HybridCNN model achieved the best result on one-step approach with 82.7% of F1, followed by WordCNN with 81.6%, LR with 81.4%, CharCNN with 81.1%, FastText with 80.4% and SVM with 79.3%. On two-step approaches the combination of two LR classifiers was the best result with 82.4% of F1, followed by HybribCNN combined with LR achieving 81.8%, two HybridCNNs with 80.7% and two SVMs with 80.3%. All the results were obtained through a 10-fold cross-validation.

In the table 2.1 the related works described above and this work are compared, indicating if and how the requirements were implemented. NT was filled with ✓ if the work used up to 3 techniques, ✓✓ if the work used up to 5 techniques and ✓✓✓ if the work used more than 5 techniques. NC was filled with ✓ if the work used up to 10 classes, ✓✓ if the work used up to 100 techniques and ✓✓✓ if the work used more than 100 classes. ML was filled with ✓ if the requirement was implemented, 'x' if the requirement was not implemented and '-' if the information was not found in the work. DS was filled with ✓ if the dataset size was up to 50K entries, ✓✓ if the dataset size was up to 500K and ✓✓✓ if the dataset size was more than 500K. PV was filled with ✓ if the requirement was implemented, 'x' if the requirement was not implemented and '-' if the information was not found in the work. PT was filled with ✓ if the requirement was implemented, ✓ * if the requirement was implemented but the work did not inform detail, 'x' if the requirement was not implemented and '-' if the information was not found in the work. DP was filled with 'nC' if cross-validation was implemented, in which 'n' is the number of folds used, 'H' if holdout method was implemented and '-' if the information was not found or the work partition the dataset not using a method. There was one work that said they use cross-validation, but they did not indicate the number of folds used, so for this we indicated in the table with 'C'.

Table 2.1: Comparison of the related work using the previously defined requirements.

Related Work	NT	NC	ML	DS	PV	PT	DP
Our Solution	✓	✓✓✓	x	✓✓✓	x	✓	5C
Pang et al. (2002)	✓	✓	-	✓	x	x	3C
Pang and Lee (2004)	✓	✓	-	✓	x	x	10C
Go et al. (2009)	✓	✓	x	✓✓✓	-	✓*	-
Glorot et al. (2011b)	✓	✓	-	✓✓	x	✓*	C
Kim (2014)	✓✓	✓	-	✓	✓	✓	10C
Badjatiya et al. (2017)	✓✓✓	✓	-	✓	✓	-	10C
Sayfullina et al. (2017)	✓	✓✓	-	✓✓	✓	✓*	-
Bajaj (2017)	✓✓✓	✓	-	✓✓	✓	✓*	H
Pavlopoulos et al. (2017)	✓✓	✓	✓	✓✓✓	✓	✓	-
Park and Fung (2017)	✓✓✓	✓	x	✓	✓	✓*	10C

According to Table 2.1, the number of classes of datasets explored in the related works is very low, being very rare finding a work using more than 10 classes, so we did not find a work that comes close to our work. In the number of techniques used we can observe that half use up to 3 techniques like us and few use more than 5. Multi-language is other requirement rarely explored in the literature and we only found one work that considered this requirement. Data size is a requirement that varies widely and we found other works using datasets with hundred of thousands, like our 870k industrial machine maintenance registers. The use of pre-trained word vectors is a very common requirement found in the literature because of the improvements achieved using them. Almost all works use some form of parameter tuning as a way to optimize their results. The 10-fold cross-validation was the form of data partition most common found in the literature and other variants of the cross-validation or other methods rarely appears.

Chapter 3

Development

In this chapter we formulate the problem a bit more in Section 3.1, we explain the proposed solutions in Section 3.2, we present the scope reductions in Section 3.4 and after we start the explanation of a traditional machine learning pipeline in Section 3.3.

3.1 Data Characterization and Problem Formulation

The problem was to indicate for a given textual description, which error code that text belonged to, so each error code contains various descriptions. An error code indicates a possible failure in the industrial machine. The error codes were also separated into groups, called chapters. This way, each chapter contains several error codes, and the same code can belong to different chapters. Our dataset contains more than 1.400.000 registers distributed in more than 25 different languages, manually labeled into more than 1200 distinct error codes and 240 chapters, collected between the years 2007-2017. The languages of the registers were discovered using an open source project called Language Detection from Google Code (https://code.google.com/archive/p/language-detection/).

The data distribution according to the 1200 error codes and the 240 chapters are illustrated in Figures 3.1 and 3.2. As we can see, few error codes have many entries in the 1.400.000 registers, consequently few classes are associate with most of registers. Precisely, 1% of error codes are associate with 19% of registers. In terms of 240 chapters, approximately 7,1% have 36,2% of total registers.

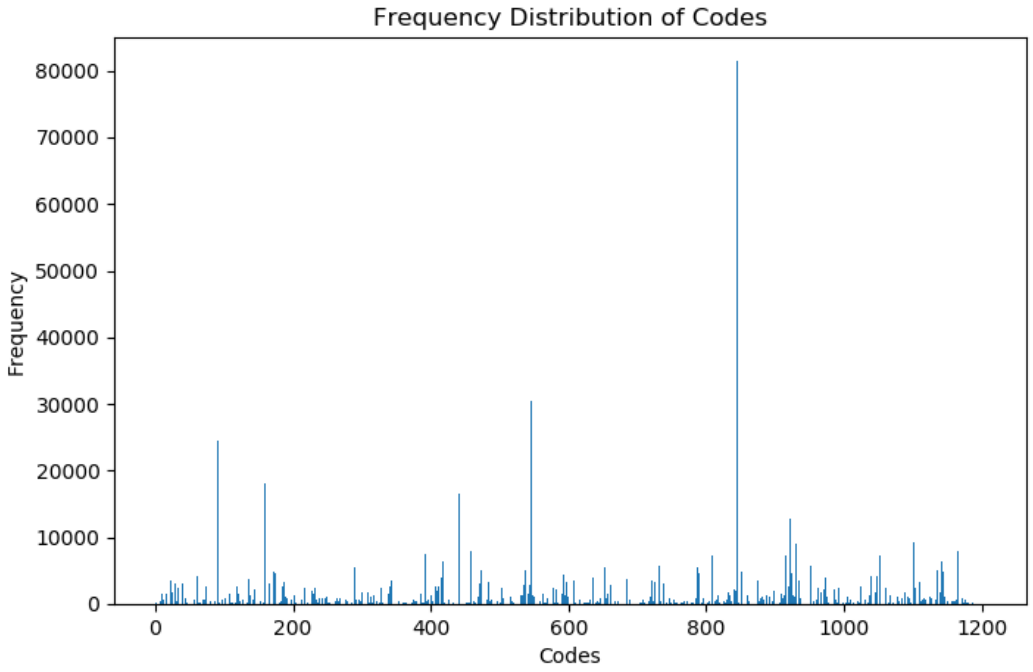


Figure 3.1: Frequency Distribution of Codes

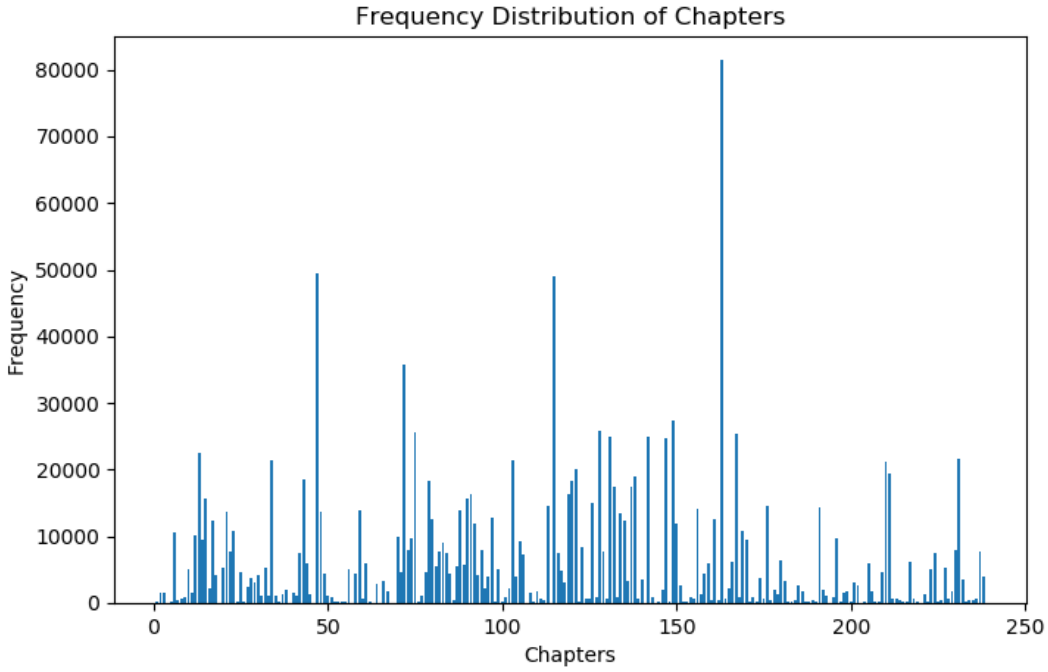


Figure 3.2: Frequency Distribution of Chapters

Given a percentage of the total registers to train the classifier (normally 80% of the input data) and the remaining registers to test the classifier with the created model(s) from the training phase, the problem is to classify correctly code errors automatically, with high accuracy and as fast as possible to evaluate each register.

3.2 Training Models

Due to the high number of chapters and error codes to be classified, we present three training models for the classifier and use one of them in this work. The first selects all the error codes together with their corresponding chapters and produce a single training model in which the classifier knows all the error codes. This solution is the fastest one to test the registers, but it is the one that achieved the worse results of f1-score due to the generality of the training model. It is the fastest because the classifier does not change the model during the test phase. This training model was used in this work with all classifiers studied.

The previous model limitation in terms of generality was solved by multiple training models, one for each chapter of error codes. This way, each classifier model is a specialization of previous solution, so probably with better f1-score, but due to high number of models this alternative performance can be the worst, depending on the hardware used. In our first implementation using FastText Joulin et al. (2016) this alternative produced around 198GB of data to produce 240 training models, what is impractical to be stored in RAM-only in most of ordinary machines. The consequence is that this alternative can be not feasible for a realistic test scenario where error codes and, consequently, chapters and sub-chapters are non-preemptive, thus many swaps may occur, turning each register test a time-consuming task, being not practical for online industrial case studies. There are alternatives to solve the multiple models problem and they are part of future studies. We can, for instance, use a 240 cluster solution, where each cluster device stores a trained model.

To solve the problems arising from the previous models, the third solution creates groups of some chapters, so 20 training models can be created aleatory from the 240 chapters. This solution balances a good result of f1-score with a good runtime. It is part of future plans to group the chapters in different ways, e.g., aleatory or using entropy measures Huang (2008) to establish the classes similarity and consequently their groups. Even this alternative is not suitable for a single machine, since more than 18GB of RAM was necessary to store around 20 training models.

3.3 Text Classification Pipeline

Figure 3.3 illustrates the steps of a text classification pipeline activity, very common for data mining analysis.

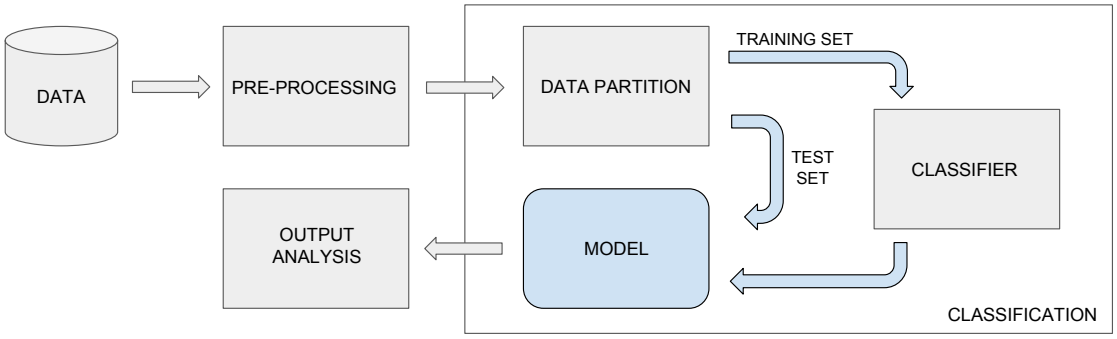


Figure 3.3: Steps of a classification pipeline

3.3.1 Pre-processing

The real world datasets contain noisy, missing and inconsistent data due to their business domains, e.g., tweets have several particularities in text, representing a concept as several text abbreviations alternatives. Another challenge for data pre-processing are the multiple heterogeneous data sources, where noisy, missing values and inconsistencies occurred very often. Among the several pre-processing techniques we have data cleaning, in which we remove noisy and correct the inconsistencies, data integration, in which we merge consistently different sets from different sources, data reduction, in which we reduce the data size eliminating redundancies, and data transformation, in which we normalize the data. The application of these techniques can considerably improve the results of the data mining Han et al. (2011).

In our dataset it was applied data cleaning, in which it was removed all registers with no label and cleaned the texts by removing their special characters and stop-words. We did not label the registers because this activity requires technical knowledge, this way all registers were previously labelled. We also used a WordNet Miller (1995) to improve each text in terms of meaning and semantic adding some synonyms in the corpus. In this step each word in the register was searched in the WordNet and four synonyms of this word were incorporated into the text. This step was responsible for tripling the size of our input file and for this reason only four synonyms were selected for each word. The WordNet is a large lexical dataset of English, where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.

We have used a stemming process in order to reduce the vocabulary space, since different words can represent similar meanings, like democracy and democratic. In this example, both words can be used and the stemming process is possible to represent those words using the same stem. The stemming is a heuristic process that tries to remove the derivational affixes in order to represent a word only by its common base form Schütze et al. (2008).

3.3.2 Classification

Data Classification is a two-step process, in which we have the learning step and the classification step. In the first step we construct the model that describes a set of classes or concepts, making a map between training registers and their associated classes. This mapping can be done by classification rules, decision trees or mathematical formulae. For instance, a register can be represented by an n-dimensional attribute vector (in this work each attribute is an n-gram) and each register is assumed to belong to a predefined class. The trained registers are samples from the training dataset, so this step is called supervised learning because we know the class label of each training register, providing it to the classifier. We also have the unsupervised learning in which we do not know the class labels such as clustering methods. In this work we only explored supervised techniques, specifically the machine learning ones.

In the second step we adopt the trained model to classify the unknown data. However, first it is necessary to measure the performance of the model and for this we can use metrics to tell us how accurate the model is in classifying the data correctly. Metrics, such as precision, recall, F1 score and so forth are used in the literature Joachims (1998b), Badjatiya et al. (2017), Pang et al. (2002), Go et al. (2009). In this work we used the F1-score, which is a harmonic mean of precision and recall, thus widely used in classification Han et al. (2011). Precision is a measure of exactness, indicating the percentage of registers labeled as positive that are actually such. Recall is a measure of completeness, indicating the percentage of positive registers are labeled as such. In order to calculate these metrics we used a confusion matrix, i.e, a table of size mxm, where m is the number of classes, a row indicates the real class of the register and the column indicates the class provided by the model. Ideally, most of the registers are represented along the diagonal of the matrix.

In the second step the test dataset are used to generate the F1-score explained previously. If we use the training dataset to measure the performance of that model, we can obtain optimistic results, because the classifiers tend to overfit the data, learning particular anomalies present in the training dataset that maybe particular for training and not for the entire dataset. The test registers should be independent of the training registers, meaning that they were not used to construct the model.

In order to train the model and measure its performance, we need first to partition the dataset from pre-processing phase into training and test. There are many ways to do the data partition, such as holdout method Han et al. (2011), in which the dataset is partitioned randomly (typically two-thirds to the training dataset and the remaining to the test dataset), random sub-sampling Han et al. (2011), a variant of holdout in which the method is repeated k times, or cross-validation Han et al. (2011), in which the dataset is divided into k folds being k-1 folds used for training and one fold for test, repeating the process k times. In this work, we implemented the 5-fold cross-validation method to ensure greater reliability of results, in which we selected the first 20% of data for validation and the remaining 80% for training,

the second 20% for validation and the remaining 80% for training, repeating the process five times to cover all partitions. The results of precision, recall and F1-score were the mean of the 5 executions. This validation alternative is also used by the following case studies Pang and Lee (2004), Kim (2014), Badjatiya et al. (2017) and Park and Fung (2017), thus present in text classification literature.

Before performing the 5-fold cross-validation, we divided the dataset into two parts (80% and 20%, respectively), being 20% used to tune the hyper-parameters while the 80% partition was assigned to the cross-validation as explained in next section. The hyper-parameters values were found using the Irace package López-Ibáñez et al. (2016), since it automatically finds the most appropriate settings of an optimization problem.

We selected three classifiers in this work and they are detailed in the next subsections. These classifiers were also selected by the following works Pang et al. (2002), Pang and Lee (2004), Go et al. (2009), Kim (2014), Badjatiya et al. (2017), Sayfullina et al. (2017), Bajaj (2017), Pavlopoulos et al. (2017) and Park and Fung (2017), thus present in most of the related work used to conduct this work.

In this work we used the TF-IDF (Term Frequency - Inverse Document Frequency) technique to represent a sentence for SVM and Random Forest. The TF factor is the count of occurrences of a term in the dataset and the IDF factor is calculated typically by log N / n, where N is the number of documents and n is the number of documents containing the term. The word is represented by the product of the TF and IDF factors Salton and Buckley (1988) and the sentence by the set of values of each word that composes it. The FastText classifier has an internal representation of sentences based on Bag of Word techniques.

3.3.2.1 FastText Classifier

The FastText classifier was introduced by Joulin et al. (2016) as a competitive solution to deep learning alternatives. This method is a linear classifier that uses a hierarchical softmax function, i.e. a normalized exponential function, to compute the probability distribution over the predefined classes and a low dimensional vector to represent a text. In general, FastText can reduce the complexity of the training step, achieving good results that are close to deep learning ones, but in a shorter runtime, precisely hours and days against seconds.

From an register, each of its word is transformed into an n-dimensional vector, by default n equal to 100, with its values initiated by an uniform real distribution. The mean of these vectors is done to generate a vector representing the register. This vector is then multiplied by an mxn matrix, where m is the number of classes of the problem in question, and n is the size of the initial vectors, resulting in a new m-dimensional vector that is then passed by the softmax function that generates the probabilities of that register belong to each of the classes. The values of this matrix are found during the training by making an error calculation at each iteration.

In FastText a hierarchical softmax is used that represents the nodes of each class with their respective probabilities using a binary tree constructed using the algorithm of Huffman Huffman (1952). This way it is possible to determine which class an register belongs to in O(log m) instead of O(m), thus accelerating both training and validation.

3.3.2.2 Support Vector Machine Classifier

Support Vector Machine (SVM) Hearst et al. (1998); Suykens and Vandewalle (1999) is a classifier that use a non-linear mapping to transform the data into a higher dimension and then find the linear optimal, separating hyperplanes using the support vectors (training registers) and margins defined by support vectors. The training time can be reduced significantly and the model can have high accuracy due to its ability to deal with complex limits of nonlinear decisions. The SVM is less sensitive to overfitting than other methods Han et al. (2011).

3.3.2.3 Random Forest Classifier

Ensemble classifier is a classifier composed of k other classifiers with the aim of creating an improved composite classification model. Random Forest Breiman (2001) is an ensemble classifier in which the classifiers are decision trees and for a register x the most voted class is selected. The construction of the trees in the Random Forest can occur in two ways: i) with random input selection; or ii) with random linear combinations. Those options change the way that the attributes are selected in each node of the tree Han et al. (2011).

3.3.3 Output Analysis

The output analysis is the last step in the classification process in which we evaluated the results in order to identify if the results obtained corresponded to the initial expectations and hypotheses. Furthermore, with the output analysis we can discuss how we could change the process to achieve the desired results. In Section 4.2 we discuss the results obtained by each classifier, as well as presenting suggestions that may improve the results. Normally, we need to balance a good accuracy result with a good runtime to test the registers.

3.4 Scope Reductions

The first scope reduction refers to the use of only the first proposed solution detailed in Section 3.2 and represents the unique training model used, a generalist version where all error codes together with their chapters are learned.

The second scope reduction refers to the use of only English registers of the dataset. Our dataset contains registers in more than 25 different languages and the English language represents 90,1% of the total as we can see in Figure 3.4. One of the main step of a text

classification pipeline is the pre-processing, where for each language stop-words are removed, synonyms are added and a stemming process is applied. Due to the limit of time in BCC 390 and 391 disciplines to produce all pre-processing steps for all languages, we start with the most frequent - English.

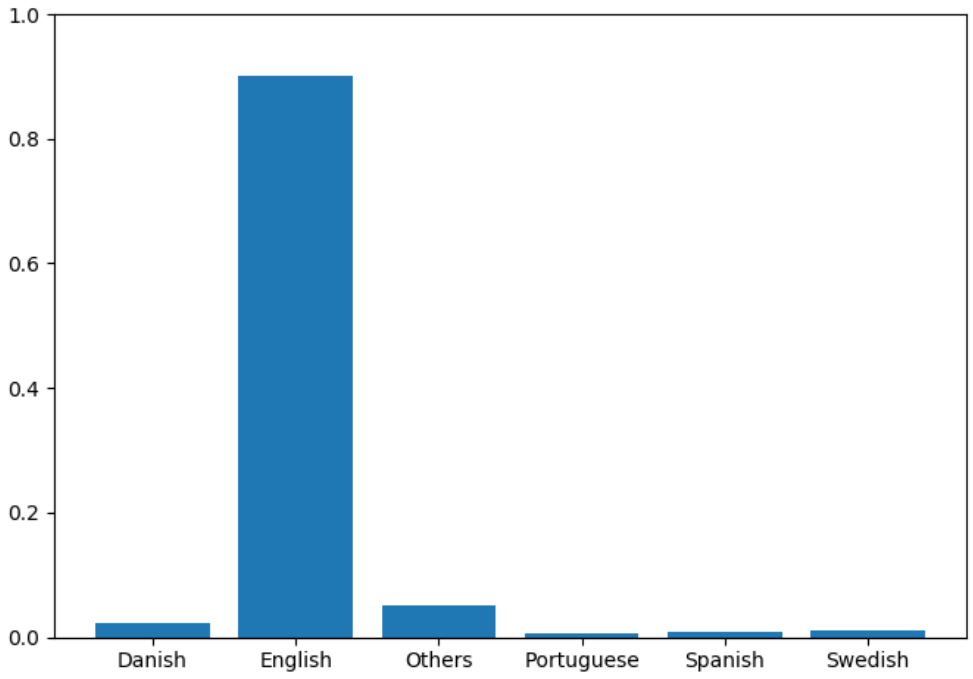


Figure 3.4: Language Distribution

Also due to the short time to produce the final report of an undergraduate Computer Science course, the last scope reduction refers to tuning the hyper-parameters of the classifiers, since it took several days to calibrate a single classifier hyper-parameters, precisely the Random Forest classifier. In this work, we decided to randomly divide the dataset into two parts (80% and 20%) and use the 20% part to tune the hyper-parameters while the 80% part was assigned to the classification phase, i.e., the test and training steps of the text classification pipeline, illustrated in Figure 3.3.

Chapter 4

Experiments

In this chapter we present the machines used in the work in Section 4.1 and the experimental evaluations of a single training model solution using FastText, SVM and Random Forest classifiers discussed in Section 4.2. The FastText result won if we considered its f1-score taking into account the time spent as we can see in Figure 4.1, while the SVM also achieved good f1-score results, but spending quadruple the FastText time. The Random Forest result was not so good, but we had to adopt some extreme configurations to make it possible of using all RAM memory available and those configurations produced both better performance and worse accuracy, so new experiments will be necessary. The Random Forest limitations are explained in details in Section 4.2.

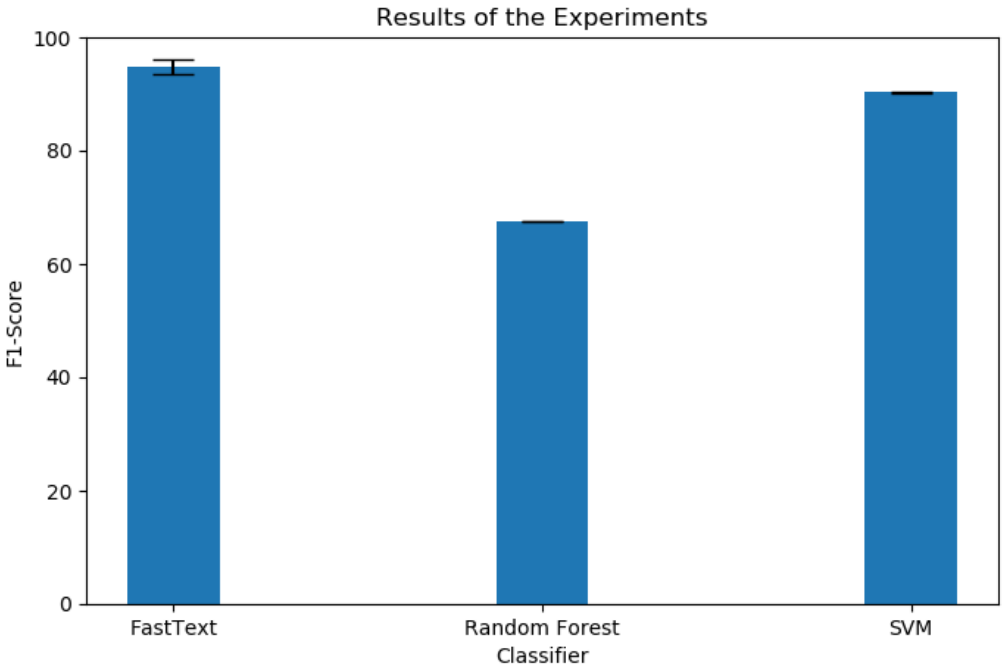


Figure 4.1: Results of the Experiments

4.1 Setup

The machines used have the following configurations:

1. Windows 7, Intel(R) Core(TM) i7-4790 @ 3.60GHz, 8 cores, 16GB of RAM;
2. Ubuntu 12.04, Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz, 24 cores, 128GB of RAM.

Our dataset consists of more than 1.400.000 registers of which about 1.300.000 are English entries. We only used English language as explained in Section 3.4. After the language selection, we divided our dataset into two parts (80% and 20%) being the 80% part assigned to the experiments using 5-fold cross-validation and the 20% part assigned to tune the hyper-parameters. The dataset was pre-processed using WordNet Miller (1995), a stemming process and a stopwords removal process as described in detail in Section 3.3.1.

4.2 Individual Results

All the results presented in this section were found over the single model solution described in detail in Section 3.2. As discussed in Section 3.4 the other proposed training models will remain as future work. The processing time refers to the total execution time of the 5 folds of the cross validation.

The result using the FastText classifier achieved 94,92% of f1-score with a processing time of 5 hours and 13 minutes. The confidence interval for FastText was 93,6% to 96,2% with 95% of confidence level. The FastText hyper-parameters tuned by Irace package López-Ibáñez et al. (2016) were learning rate, epoch and n-gram and their values are: 0,9854, 33 and 2, respectively as shown in Table 4.1. As we can see in the Table 4.4, FastText obtained a small value of standard deviation, which indicates that the results are conclusive and each fold provided a representative partition of the dataset.

In terms of runtime, FastText outperformed the others by far, since it was at least twice as fast as the others, but this result can be even better. Unfortunately, the biggest machine could not execute the FastText due to GCC deployment problems in such a hardware, so we estimated how fast it was by comparing both machines. The machine 2 has three times more processing power and eight times more memory than machine 1, but the latter would not be necessary since FastText did not spend more than 7GB of memory.

Table 4.1: FastText Tuning Process.

Parameter	**Range**	**Best Value**
Learning Rate	0,1 - 1,0	0,9854
Epoch	20 - 40	33
N-Gram	1 - 4	2

The SVM result was also good, achieving 90,27% of f1-score with a processing time of 8 hours and 6 minutes. The confidence interval for SVM was 90,2% to 90,3% with 95% of confidence level. This processing time was achieved using the machine 2, but without requiring huge amount of RAM memory, precisely no more than 8GB of memory. The SVM hyper-parameters tuned by Irace package were: C parameter (penalty parameter of the error term), maximum iterations and n-gram and the values obtained: 0,9671, 982 and 1, respectively as we can see in Table 4.2. SVM also obtained a small value of standard deviation, indicating a conclusive result with the 5-folds, as we can see in the Table 4.4. The result achieved by SVM confirmed what the literature has already reported Joachims (1998b).

Table 4.2: SVM Tuning Process.

Parameter	**Range**	**Best Value**
C	0,1 - 1,0	0,9671
Iteration	500 - 1000	982
N-Gram	1 - 2	1

The Random Forest, as well as the SVM, was executed using the machine 2, but even in a top quality machine such a method consumed all the existing memory when we set the hyper-parameters tuned values. The Random Forest hyper-parameters tuned by Irace package were

number of estimators (number of trees in the forest), criterion (function to measure the quality of a split) and n-gram. The values obtained are: 49, gini and 4, respectively as shown in Table 4.3. As the other methods, Random Forest obtained a low value of standard deviation.

Since it was not possible to run the method in our biggest machine with its tuned parameters, it was necessary to tune the split parameters of Random Forest until the required memory was less than 250GB, so swap was necessary, since the machine has 128GB of RAM. The limitations in the split parameters probably made f1-score worse and the method fast. It was the one that had the lower runtime, concluding the task in 2 hours and 10 minutes, as we can see in the Table 4.4. Another factor that helps to explain the memory consumption problem is that Irace selected 4-gram for Random Forest and the number of features using 4-gram exceeds 9 million compared to the 1-gram that generates just over 105 thousand of features.

Table 4.3: Random Forest Tuning Process.

Parameter	**Range**	**Best Value**
Estimators	10 - 50	49
Criterion	gini, entropy	gini
N-Gram	1 - 4	4

Table 4.4: Results of the Experiments.

Model	**F1-Score**	**Confidence Interval**	**Standard Deviation**	**Processing Time**
FastText	94,92%	+- 1,31%	0,015	5h13min
SVM	90,27%	+- 0,0438%	0,0005	8h6min
Random Forest	67,50%	+- 0,0877%	0,001	2h10min

Chapter 5

Conclusion

In this work, we present the problem of classifying correctly and fast industrial machines error. This case study is important due to the number of different industrial machines that are in maintenance every day in so many industries niches, so with this work it is possible to understand that it is feasible to automatically classify huge amount of errors with both good accuracy and good runtime. Other steps of the industry maintenance pipeline, like the automatic classification of how to fix the previous detected errors have real potentials in producing also promising results.

An important innovation of this work to the classification literature is the high number of classes, where the error codes bypassed 1200 different values and in the literature there are case studies with only tens of classes. Three classifiers were used and two of them produced promising results in terms of accuracy, but we can conclude that the FastText method outperformed the others in terms of both accuracy and runtime, since the Random Forest requires conclusive tests with all the hyper-parameters tuned according to Irace. Thus, future experiments must occur with such a method.

As presented in Section 3.2, we proposed three training models to solve the stated problem, but just one was implemented, the generalist one, so the others have a great potential of achieving high f1-scores with a reasonable runtime if deployed in clusters and not in single machines. The FastText must be re-executed in identical machines with the other methods to have realistic comparisons. The other languages must be incorporated. The use of CNN classifier Kim (2014) is part of our future work as a representative of deep learning category. We also intend to explore, in the future work, the word embedding techniques like Word2Vec Mikolov et al. (2013b) and GloVe Pennington et al. (2014) to represent the sentences. In the literature it has been shown that good improvements have been achieved using those techniques Kim (2014).

Bibliography

Badjatiya, P., Gupta, S., Gupta, M., and Varma, V. (2017). Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Bajaj, S. (2017). "the pope has a new baby!" fake news detection using deep learning.

Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447.

Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Chen, M., Mao, S., Zhang, Y., and Leung, V. C. (2014). Big data generation and acquisition. In *Big Data*, pages 19–32. Springer.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Glorot, X., Bordes, A., and Bengio, Y. (2011a). Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323.

Glorot, X., Bordes, A., and Bengio, Y. (2011b). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Habibzadeh, H., Boggio-Dandry, A., Qin, Z., Soyata, T., Kantarci, B., and Mouftah, H. T. (2018). Soft sensing in smart cities: Handling 3vs using recommender systems, machine intelligence, and data analytics. *IEEE Communications Magazine*, 56(2):78–86.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Huang, A. (2008). Similarity measures for text document clustering. pages 49–56.

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

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

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

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer.

Li, S. and Zong, C. (2008). Multi-domain adaptation for sentiment classification: Using multiple classifier combining methods. In *Natural Language Processing and Knowledge Engineering, 2008. NLP-KE'08. International Conference on*, pages 1–8. IEEE.

Li, X. and Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., and Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

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

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Pan, S. J., Ni, X., Sun, J.-T., Yang, Q., and Chen, Z. (2010). Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760. ACM.

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Park, J. H. and Fung, P. (2017). One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*.

Pavlopoulos, J., Malakasiotis, P., and Androutsopoulos, I. (2017). Deep learning for user comment moderation. *arXiv preprint arXiv:1705.09993*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

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

Sayfullina, L., Malmi, E., Liao, Y., and Jung, A. (2017). Domain adaptation for resume classification using convolutional neural networks. *arXiv preprint arXiv:1707.05576*.

Schütze, H., Manning, C. D., and Raghavan, P. (2008). *Introduction to information retrieval*, volume 39. Cambridge University Press.

Silva, J., Coheur, L., Mendes, A. C., and Wichert, A. (2011). From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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.

Stieglitz, S., Mirbabaie, M., Ross, B., and Neuberger, C. (2018). Social media analytics–challenges in topic discovery, data collection, and data preparation. *International Journal of Information Management*, 39:156–168.

Suykens, J. A. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Wang, S. and Manning, C. (2013). Fast dropout training. In *international conference on machine learning*, pages 118–126.

Wang, S. and Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Waseem, Z. (2016). Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on NLP and computational social science*, pages 138–142.

Waseem, Z. and Hovy, D. (2016). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Wiebe, J., Wilson, T., and Cardie, C. (2005). Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Wulczyn, E., Thain, N., and Dixon, L. (2017). Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399. International World Wide Web Conferences Steering Committee.

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

Certifico que o aluno **Samuel Queiroz Souza Rocha**, autor do trabalho de conclusão de curso intitulado "**Analysis and Classification of Texts for Industrial Machines Maintenance**", efetuou as correções sugeridas pela banca examinadora e que estou de acordo com a versão final do trabalho.

Joubert de Castro Lima
Orientador

Ouro Preto, 17 de dezembro de 2018.