University of Stuttgart
Germany

EXPLORING CLASSIFICATION ALGORITHMS AND DATA FEATURE SELECTION FOR DOMAIN SPECIFIC INDUSTRIAL TEXT DATA

Master Thesis IMSE

Student:
Alejandro Gabriel Villanueva Zacarias

Academic Supervisors:
Prof. Dr.-Ing. Habil. Bernhard Mitschang – University of Stuttgart
Prof. Dr. Kostas Magoutis – University of Crete
Prof. Dr. Willem-Jan van den Heuvel– Tilburg University
M.A. Laura Bernadette Kassner – University of Stuttgart

Abstract

Unstructured text data represents a valuable source of information that nonetheless remains sub utilised due to the lack of efficient methods to manipulate it and extract insights from it. One example of such deficiencies is the lack of suitable classification solutions that address the particular nature of domain-specific industrial text data. In this thesis we explore the factors that impact the performance of classification algorithms, as well as the properties of domain-specific industrial text data, to propose a framework that guides the design of text classification solutions that can achieve an optimal trade-off between accuracy and processing time. Our research model investigates the effect that the availability of data features has on the observed performance of a classification algorithm. To explain this relationship, we build a series of prototypical Naïve Bayes algorithm configurations out of existing components and test them on two role datasets from a quality process of an automotive company. A key finding is that properly designed feature selection techniques can play a major role in achieving optimal performance both in terms of accuracy and processing time by providing the right amount of meaningful features. We test our results for statistical significance, proceed to suggest an optimal solution for our application scenario and conclude by describing the nature of the variable relationships contained in our research model.

Keywords: text mining, unstructured text data, classification algorithms.

TABLE OF CONTENTS

TABLE OF FIGURES

1 Introduction

As data grows, so does the need to design new ways to manage and process its ever increasing volume. For years public and private organizations have collected data in an electronic format concerning multiple issues, from technology-enabled processes to automated monitoring via sensors. In this context, unstructured data, in particular text data, represent a challenge not just to manage it in an efficient manner, but also to integrate it in an effective way. This means combining structured and unstructured data sources so the latter can provide depth and insight to queries for which the former only gives shallow (but wide) answers.

From a scientific standpoint, this challenge derives into an equally ambitious problem. Unstructured text data by nature requires new processing and analytic models which are not immediately compatible with the traditional structured ones. Moreover, these structured processing and analytic models are in comparison more efficient in terms of time and computational resources; something to consider given the current size and projected growth of unstructured data.

In this work we explore one possible solution to these organizational challenge and scientific problem in the context of automated text classification. We explore the various ways in which unstructured text data properties impact the performance of this kind of task. For this we focus on text classification algorithms and the selection of features used by them.

We start by defining the nature of the unstructured data we are concerned with. As a result of this we characterise the concept of "messy data" defined by (Kassner & Mitschang 2016) to adequately meet the context of a real business scenario in an automotive company from which this research topic originates (see (Kassner & Mitschang 2016)).

With this characterisation in mind we devise a method to select the appropriate text data features to test different *classification algorithm configurations* (variants in the inner workings of the same selected basic classification algorithm) and as a result, identify the one that yields the optimal performance in terms of accuracy and processing time. This method relies on a conceptual architecture developed to identify the elements from various disciplines that need to be taken into consideration when choosing a classification algorithm. In addition to this, we define a list of requirements that the ideal classification algorithm should have.

As a result of applying our method, we proceed to test the Naïve Bayes algorithm with different configurations and feature sets applied to the above mentioned business scenario/process from an automotive company. We benchmark their performance and discuss how these algorithm configurations perform compared to configurations from an adapted k Nearest Neighbours algorithm. This allows us not only to discuss the relevance of the solutions proposed regarding the defined problem, it also enables the definition of future research lines to generate a more optimal solution.

1.1 Research Motivation

The pervasive use of information technology in all aspects of society has led in the last decades to an ever-accelerating growth in the creation and manipulation of data. By 2012, (Turek 2012) estimated that from the beginning of recorded time until 2003, mankind had created 5 Exabytes of information, whereas in 2011 that same amount was created every two days (Turek 2012). Based on a study by (Turner et al. 2014) from IDC, in 2013 that same amount of data took a little less than 10 hours to be created or copied, and by 2020 it is predicted to take a little less than one. This data explosion is of particular importance for companies, since they already had liability or responsibility for 85% of the data in 2013 (Turner et al. 2014).

This phenomenon has become widely recognized since 2011 as Big Data (Gandomi & Haider 2014). Even though the concept has been open to discussion, there are certain characteristics that can define it in terms of data management challenges. In addition to the standard three V's of Big Data (Volume, Variety and Velocity), (Gandomi & Haider 2014) also mention additional challenges such as Veracity, Variability, and Value.

In many cases, these challenges are derived from the fact that 95% of all data is unstructured and cannot be processed with traditional tools and techniques (i.e. relational databases) (Gandomi & Haider 2014). This has led to the creation of new technologies that can deal with vast amounts of data (Volume), coming from different sources (Variety) at different flow rates (Variability). Additionally they have to be prepared to process it at the required speed (Velocity), despite of the presence of unreliable sources (Veracity) and the low density of value relative to the volume (Gandomi & Haider 2014).

One area were the challenge is particularly significant for companies is data analytics. Although this area has been traditionally dominated by structured data techniques (Lang et al. 2009), the growth of unstructured data has brought the opportunity to enrich the structured analysis by providing insights hidden in it. This is relevant because organizations have been collecting both kinds of data (Gandomi & Haider 2014), but have struggled to properly integrate them.

There have been different approaches to address this integration problem. (Lang et al. 2009) consider the integration of unstructured text data into data warehouse applications via *Unstructured Business Intelligence*. This term comprises three steps that aim to enrich the existing ETL flow, warehouse schema, and BI infrastructure. (Gandomi & Haider 2014) points out a process by Labrindis and Jagadish with five stages organised in two main sub-processes: data management and analytics. "Data management involves processes and supporting technologies to acquire and store data and to prepare and retrieve it for analysis. Analytics, on the other hand, refers to techniques used to analyse and acquire intelligence (…)" (Gandomi & Haider 2014). Aside from specific examples, (Kassner & Mitschang 2016) mention that normally "approaches to automatically analyzing (sic) these unstructured data with traditional analytics for structured data are either very specific and case-based or too generic."

As we see, there is a valid research interest in further exploring ways to do unstructured data analytics. In the particular context of this work, this interest can be described both from a scientific and an industry perspective.

1.1.1 Scientific Perspective

From a research standpoint, there are several challenges that structured analytics do not have to face. (Lang et al. 2009) mentions on one side the need to deal with misspellings, domain-specific, company-specific or even employee-specific acronyms and on the other side advanced data cleansing, to pre-process data beyond the capabilities of traditional cleansing. On top of that, text analysis technology typically requires an adaption to the domain where it is applied to work correctly (Lang et al. 2009), which makes this technology hard to apply and maintain in different scenarios. (Lang et al. 2009) also point out that unstructured text data involves term disambiguation based on the context, something that is rarely leveraged by traditional analytics.

In a similar way, (Gandomi & Haider 2014) highlight the problems that result from applying traditional statistical methods to big volumes of unstructured data.

There are two problems in particular that originate from the inherent properties of unstructured text data. Firstly, since this data can be obtained from multiple sources, it can actually represent different sub-populations instead of a single one. If this is not recognised, small populations may be discarded under the assumption that they are outliers. Secondly, because of the sheer amount of data, independent random variables or features, may show false correlations.

In addition to this, some statistical methods may not be good enough in terms of computational efficiency to be feasible in the scale we are dealing with (Gandomi & Haider 2014).

1.1.2 Industry Perspective

For companies, achieving an effective integration of unstructured data means achieving time reductions to discover failures or complains, and as a result being able to react before it is too late. This is key in avoiding losing customers in favour of the competition and to maintain a good reputation with the existing and potential customers (Lang et al. 2009). Not only that, whenever failure or error happen, an effective use of unstructured data can also improve the quality of the detection by providing direct insight to the causes (Lang et al. 2009).

This is all the more important "in the face of ever larger amounts of data, faster innovation cycles and higher product customization (…)" (Kassner & Mitschang 2016).

1.2 Research Problem

This thesis is a continuation of the work of (Kassner & Mitschang 2016) in collaboration with a large automotive original equipment manufacturer (OEM). As such, the problem it addresses as well as the environment where it occurs are the same. In the following subsections there are reference descriptions of these two elements.

1.2.1 Process Description

The research problem occurs in a business environment, specifically in a quality inspection process in an automotive OEM. In this process, parts removed from cars already owned by customers are analysed to inspect the quality issues involved in their potential failure. This analysis involves three different roles, each one inspecting the part and writing a report until an error code is assigned. It is this error code the one that categorises each text report into 1 of 1271 possibilities (for the purposes of our scenario; total classification categories are actually more). The flow of data as the process progresses is shown in Figure 1.

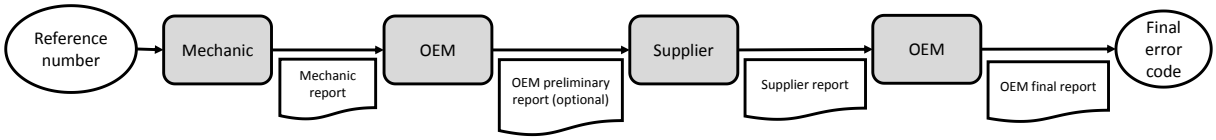


Figure 1 Simplified quality inspection process (adapted from (Kassner & Mitschang 2016))

To assign the final error code, all reports written up to that moment (second participation of the OEM worker) are considered and a decision is made by a human quality expert at the OEM. In addition to the text reports being created, other structured data is also recorded for every car part that goes through this process.

To support the work of the quality expert in this classification task, a system should receive as input several texts in unstructured format related to each faulty or damaged car part along with their related structured data (see Figure 9), set that we will call a *data bundle* (Kassner & Mitschang 2016). Every *data bundle* has then to be processed to suggest a list of possible error codes to the quality expert.

From an academic perspective, this can be thought then as a text analytics task, in particular a specific application of automated text classification. As such, this task has to pick some features out of the unstructured dataset that allow executing a suitable classification algorithm. The "technical" process (the execution of the text classification task) begins with the reports to be classified located into a single source ready to be processed and finishes when the reports are given a list of error code classifications.

1.2.2 Problem Description

There are particular characteristics of the process and the dataset that give the research problem a different nature from purely academic ones.

Concerning the process, the fact that it is a quality management process and not a manufacturing one makes its behaviour less predictable. This because there is no a priori estimation that can be made about the process load or performance beyond the fact that the quality process is to some extent related to the manufacturing volume at a previous moment in time.

Regarding the dataset, there several characteristics that make it different from traditional approaches. First, the wide amount of categories to classify a text, exemplified by the existence of more than 1200 error codes in a dataset of 7500 instances; second, what (Kassner & Mitschang 2016) describe as *messy data*: "Text which consists of non-standard, domain-specific language, riddled with spelling errors,

idiosyncratic and non-idiomatic expressions and OEM-internal abbreviations.". On top of this already identified challenges, other important issues such as multilingualism, incompleteness of records, and a very big amount of possible classification categories (see section 4.3) complete a data profile that contradicts almost every good practice in traditional information systems. In addition to this, the fact that not every car part has the same amount of reports also adds to the dataset heterogeneity.

Previously, (Kassner & Mitschang 2016) have implemented several prototypical classifiers based on an adaptation of the k-Nearest-Neighbours (k-NN) algorithm (see sub section 3.3.6). In it, the result consists of a list of up to 25 suggested error codes instead of just one code assignation per *data bundle*. This enables the authors to evaluate the algorithm accuracy at different cut off levels (1, 5, 10, 15, 20 and 25 elements) and aligns with the goal of supporting the work of the human expert instead of replacing him.

They run this adapted k-NN algorithm in 12 different configurations (or variants) considering the following factors:

- *Data abstraction model*: using all words in the text as classification features (*bag of words*) or the identified mentions of error and parts (*bag of concepts*).
- *Similarity measure*: Used to delimit the scope of the majority vote to assign error codes. Either *Jacquard similarity coefficient* (the absolute value of the intersection of feature sets A and B over the absolute value of the union of feature sets A and B) or *Overlap similarity* (the absolute value of the intersection of feature sets A and B over the absolute value of the smallest feature set, either A or B)
- *Roles*: Using the reports of the *Mechanic role only*, using the reports of the *Supplier role only*, and using *all data available* to the human expert to assign the error code (mechanic report, optional initial report, supplier report, part description).

In all cases the pre-processing consists of tokenisation (based on punctuation and whitespaces) and language detection.

When analysing the accuracy results, they are compared (among others) to the accuracy of a so *called code frequency baseline*, which consists of retrieving all error codes available for the part type considered, ranking them by frequency and returning the desired cut off level. This baseline accuracy has values of (approximately) 35% at 1, 76% at 5, 88% at 10, 90% at 15, 94% at 20, and 100% at 25, which is assumed to be "an artifact (sic) of our randomly selected data set." (Kassner & Mitschang 2016)

In their results, all configurations tend to converge to a similar (high) value when the cut off is made at 25. This makes the lower cut off levels the ones of interest. The best configuration is the "*bag of words with Jacquard similarity on all available text*" with accuracies of 81% at 1 and 94% at 5, closely followed by the configuration "*bag of words with Jacquard similarity on the Supplier report only*" with accuracies of 78% at 1 and 93% at 5.

On the opposite side the worst configurations are those using only the Mechanic report, with accuracies of 16% to 29% at 1, below the *code frequency baseline*. This makes it clear that in terms of roles, text data coming from the Supplier can be assumed more useful than that of the Mechanic. This, the authors observe, can be attributed to the quality of each data source: "Mechanic reports tend to be poor in detail, focused on superficial problem description and often error-riddled, such that even human experts cannot draw conclusions about the detailed nature of the problem, whereas supplier reports tend to contain more detail and include descriptions of potential causes." (Kassner & Mitschang 2016)

Also, when it comes to the *data abstraction model*, the *bag of words* configurations show most of the time better accuracy levels than those of the *bag of concepts*, especially in lower cut off levels. This however, comes with a price. Given the fact that every word is a feature in this approach, it is easy to encounter memory and processing time issues even at this reduced experimental level. In the *bag of words* approach classification takes about 11 minutes for ca. 1250 *data bundles*, resulting in approx. 0.5 seconds of processing time per data bundle. In contrast, the *bag of concepts* approach classifies the same amount of data bundles in three minutes, or 0.14 seconds per unit. This turns the *bag of words* approach inviable for a real implementation.

As we can see, there is a trade-off between accuracy and processing time. While it is clear that features and the way they are represented plays a significant role in the final outcome, there is no certainty on how all these variables interact, and whether this is the case with other classification algorithms.

To investigate this matters, we can think of the *bag of words* and *bag of concepts* approaches as opposite ends in a continuous spectrum of features usage for text classification. On one side, the *bag of words* approach proposes to use all words in a document collection as features for classification, ensuring in this way that even the least significant of the terms is taken into account, albeit at the expense of bigger training sets, and prolonged execution times. On the other side, the *bag of concepts* approach aims to use only a carefully selected set of words that represent relevant concepts in the domain at hand (parts, failures, errors, symptoms in our case), based on the premise that those are the truly meaningful words upon which the classification should be run. However, as (Kassner & Mitschang 2016) suggest, having a reduced knowledge coverage of the applicable concepts in our domain can translate into a less than optimal accuracy.

Framing the accuracy vs. processing time trade-off in this spectrum hints a possible way to discover the optimal middle point. In this thesis we direct the research efforts towards the *bag of words* approach by exploring the kind of relationship that features (and subsets thereof) have with the performance of classification algorithms both in terms of accuracy and execution time. By doing this, we can help understand how can someone identify a good subset of classification features as well as its optimal use with a suitable classification algorithm.

1.2.3 Goal Description

To understand the effects of text features on an algorithm`s accuracy regarding the classification of car part *data bundles* in one of several error code categories. This should be done via the implementation of a different classification algorithm, whose performance can then be compared to the performance of the (previously) adapted k-NN algorithm. As part of this goal, there has to be a selection of the most useful features to perform the classification.

Additionally, improvements in the processing time of the classification algorithm in comparison to the baseline performance achieved by the derived k-NN algorithm are also desirable.

1.2.4 Goal Metrics Definition

Derived from the description above, the following metrics were defined:

- Accuracy: Number of instances (*data bundles* of text reports) assigned the correct error code over the total number of instances being classified. This assignation is measured at different positions in a list of error code suggestions (1, 5, 15 and 25).
- Processing time: Total amount of time elapsed for the classification to complete. Time spent per instance (text report).

1.3 Research Questions

Throughout this thesis, the answers to the following questions are explored. They are meant to decompose the research problem and goal into more manageable tasks. Relevant sections that answer these questions are referred as well.

1. How to conceptualize the process of Automated Classification based on features of unstructured text data? (see Figure 8 in Chapter 3)
2. What are the best features to classify unstructured text data? (see section 5.9)
3. What characteristics make classification algorithms more suitable for this problem? (see section 5.1)
4. What factors and/or features affect the performance and accuracy of a classification task? (see section 5.9)

5. How do unstructured text data features relate to a classification algorithm's performance? (see Figure 4 Research model and section 5.9)

1.4 Research Method

So far, the nature of the process and problem have made clear that this study is focused on what (Hevner et al. 2004) consider addressing a problem in an organizational context, so that "the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished". This sets the direction of this thesis to that of Information Systems (IS) research.

Moreover, the objective of prototyping a new technical solution to the defined problem in section 1.2 can be seen as designing and implementing "innovations that define the ideas, practices, technical capabilities, and products" to "extend the boundaries of human problem solving and organizational capabilities" (Hevner et al. 2004). Because of this, Design Science Research is a valid and meaningful methodological framework to design the research method of this work.

According to (Hevner et al. 2004), Design Science is a paradigm in IS research that is based in the pragmatic principle of utility, according to which research contributions "should be evaluated in light of its practical implications".

When performing research based on this methodology, the authors suggest to avoid its direct application, advising instead to use "creative skills and judgment to determine when, where, and how to apply each of the guidelines in a specific research project." (Hevner et al. 2004). Supported on this, we present in the remaining sections the concepts that are of particular importance for this work.

1.4.1 Design Science Process

As a research process, (Hevner et al. 2004) describe Design Science as an iterative set of activities concerned with the creation and application of an artefact. The iteration derives from the fact that the artefact is artificial in nature, an abstraction of reality. As such, there is a need to evaluate it against the problem it is intended to solve, both to improve the effectiveness of the artefact design and to gain understanding of the problem. As a result, there are two main activities to distinguish in this loop, namely: build and evaluate.

For a solution to be considered valid, it has to observe the means, ends and laws imposed by the environment. "Means are the set of actions and resources available to construct a solution. Ends represent goals and constraints on the solution. Laws are uncontrollable forces in the environment." (Hevner et al. 2004). The set of solutions that meet these three conditions can be represented mathematically, even though that hardly occurs in IS research (Hevner et al. 2004).

Once a solution is found, it is more relevant to delimit the exact conditions and cases for which the solution works, instead of finding out why it works (Hevner et al. 2004). Additionally, to determine how good a solution is, it can be compared to a pre-defined optimal solution or against existing solutions.

1.4.2 Guidelines

While carrying out the process described in the previous subsection, (Hevner et al. 2004) describe some guidelines to assess the adherence to the Design Science paradigm. These are shown in Figure 2 Seven guidelines of Design Science ResearchFigure 2 along with a brief description.

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Figure 2 Seven guidelines of Design Science Research (Hevner et al. 2004).

Guidelines 1, 3 and 6 are covered in more detail in other points of this work and as a result, they are not commented here but instead discussed in subsections 1.4.3, 1.4.4 and 1.4.1 respectively.

In the context of guideline 2 and in general for this methodology, a problem is defined as a difference between the current and intended states of a system. The actions to go from the former to the latter are driven by goals given by the problem's context, which is composed of the business needs. This may be expressed as profit maximization, cost reduction, resource consumption, performance optimizations, etc. Because IS target those same goals, the research problem is relevant as long as its solution contributes to their fulfilment by IS (Hevner et al. 2004). The details on how this work is relevant according to this guideline is presented in sections 1.1 to 1.3.

In guideline 4 (Hevner et al. 2004) distinguish among three main kinds of contribution: the design artefact, the foundation knowledge or the methodologies. The first is the most common since it is the product of the design science methodology (see subsection 1.4.3); the second refers to extensions or improvements of the knowledge base; the third focuses on evaluation methods and metrics being invented or creatively used. In all cases, the contribution is deemed valid or not in terms of what the authors call *representational fidelity* (to the business environment and technology environment) and *implementability* (sic) (to actually solve the business need). The contributions of this thesis are detailed in chapters 4 and 5.

Concerning research rigor, guideline 5 advocates for the effective application of the knowledge base both in the creation and evaluation of artefacts. This ensures rigor in research, particularly during the artefact creation. However, rigor should also be balanced against relevance, because formalism may decrease the degree in which an artefact can be applied or generalized (Hevner et al. 2004). Taken into the evaluation part of the process, this means ensuring that subject groups used for evaluation should aim "to determine how well an artifact (sic) works, not to theorize about or prove anything about why the artifact (sic) works." (Hevner et al. 2004)

Finally, guideline 7 advises to provide technology and management-oriented audiences with relevant information about the research based on their profiles. The first "need sufficient detail to enable the described artefact (sic) to be constructed (implemented) and used within an appropriate organizational context." (Hevner et al. 2004), while the second need information "to determine if the organizational

resources should be committed to constructing (or purchasing) and using the artifact (sic) within their specific organizational context." (Hevner et al. 2004)

1.4.3 Design Artefact

Considered the main result of design science research, "IT artifacts (sic) are broadly defined as constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems)." (Hevner et al. 2004) In all four cases, the authors emphasize their definition of an IT artefact does not "include people or elements of organizations (…) nor (…) the process by which such artifacts (sic) evolve (…)" (Hevner et al. 2004).

Constructs are the vocabulary to define problems and solutions. As a result, they enable the creation of models. Models then employ those constructs to represent a real world situation where the design problem, the solution space and the way these two connect can be identified. As such, models are useful to explore "the effects of design decisions and changes in the real world." (Hevner et al. 2004).

Methods are guidelines for the solution of problems, in other words they advise on how to improve the construction process of a design artefact. Depending on the scenario (as defined by the model and constructs, and within them, the problem and environment) they can be "formal, mathematical algorithms that explicitly define the search process" or "informal, textual descriptions of 'best practice' approaches, or some combination." (Hevner et al. 2004)

Artefacts of the type instantiation are the demonstration of a design and of the process that led to it. Thanks to this is that they are considered *significant IS research.* According to (Hevner et al. 2004), it is fundamental that this instantiation occurs after an initial assumption of uncertainty, in other words, that the artefacts proves possible something that has not been done before.

It is important to mention that instantiations can be thought as precursors of the other artefacts, since they are proof of feasibility for otherwise purely theoretical concepts. Therefore, once an instantiation is made, constructs and models can be elaborated to properly define the problem being solved and the possible solutions that can be further developed. This then triggers future research involving defining or using methods to explore the solution space.

Artefacts produced in this thesis are presented, even if they are covered elsewhere, in chapter 4.

1.4.4 Evaluation Methods

(Hevner et al. 2004) summarize the available evaluation methods as shown in Figure 3. They acknowledge that these methods are meant to be applied on the basis of suitability to the artefact. In other words, considering both the requirements of the problem the artefact is intended to solve and the knowledge base employed in the design of the artefact, one must select the most appropriate method.

Methodologies	Methods
1. Observational	Case Study: Study artifact in depth in business environment
2. Analytical	Field Study: Monitor use of artifact in multiple projects
	Static Analysis: Examine structure of artifact for static qualities (e.g., complexity)
	Architecture Analysis: Study fit of artifact into technical IS architecture
	Optimization: Demonstrate inherent optimal properties of artifact or provide optimality bounds on artifact behavior
	Dynamic Analysis: Study artifact in use for dynamic qualities (e.g., performance)
3. Experimental	Controlled Experiment: Study artifact in controlled environment for qualities (e.g., usability)
	Simulation – Execute artifact with artificial data
4. Testing	Functional (Black Box) Testing: Execute artifact interfaces to discover failures and identify defects
	Structural (White Box) Testing: Perform coverage testing of some metric (e.g., execution paths) in the artifact implementation
5. Descriptive	Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifact's utility
	Scenarios: Construct detailed scenarios around the artifact to demonstrate its utility

Figure 3 Design Evaluation Methods (Hevner et al. 2004).

Taken this observation into consideration, the corresponding evaluation methods for the artefacts presented in this thesis depend on the knowledge base discussed on Chapter 3.

1.5 Thesis Structure

The rest of this thesis is organized in the following manner. Chapter 2 elaborates on the adaption of the Design Science methodology to the specific environment of this work, starting with the presentation of the research model to be explored. Chapter 3 describes the relevant concepts that conform the theoretical foundation that can serve as knowledge base for artefact creation and evaluation. These theoretical concepts set the context to define the contributions of this thesis in relationship with relevant previous efforts. Chapter 4 presents the resulting artefacts and contributions. Chapter 5 then proceeds to present the results of applying the designed artefacts into a real scenario as described in subsections 2.6 and 2.7. And chapter 6 finally discusses some directions to continue the research along with the final concluding thoughts.

2 Methodology

In this chapter we describe the stages considered to carry out the research for this topic. Each section briefly describes the main activities, focus and goals of the corresponding stage as well as the adaptation of the Design Science concepts presented in section 1.4 to the particular context of this work. Finally in section 2.8 we mention the limitations of this study as a way to delimit its scope.

Based on the problem environment described in the previous chapter and based on the concepts of (Cooper & Schindler 2011), we establish the research model in Figure 4 that serves as a starting point for the adaption of the Design science research methodology.

In this model we define *Availability of Data Features* as the presumed cause, operationalized by the variables *Quantity of data features* and *Quality of data features*.

As presumed effect we establish *Classification Algorithm Performance* as defined by *Classification algorithm accuracy* and *Classification processing time*. Both of these operational variables are defined in sub section 1.2.4.

As moderating variables (MV) in this model we identify the following constructs: first and foremost the *Classification Algorithm*, along with *Feature Selection Strategy* and *Feature Extraction Mechanism*.

As confounding variable we designate the *Amount of categories*, whereas the *Availability of complementary structured data* is considered a control variable.

With this model we aim to explore the effect that the presence (or absence) of relevant data features has on the performance of a classification algorithm as to determine a method to optimize the selection of the best algorithm configuration and features; but without focusing on the effects that the amount of categories or the availability of structured data may have.

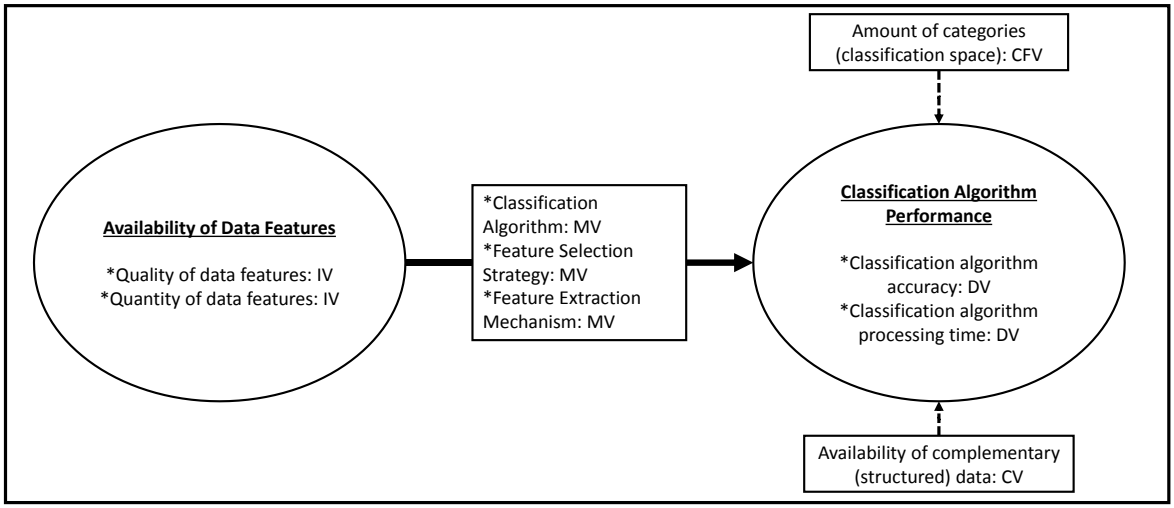


Figure 4 Research model based on the concepts of (Cooper & Schindler 2011)

2.1 Knowledge Base Creation

The first step to build a series of artefacts that can provide information about the effect of interest in our research model is to build a knowledge base. Based on the previous work by (Kassner & Mitschang 2016) and the execution of a literature survey, we compile a collection of theoretical concepts coming from four main disciplines: Natural Language Processing (NLP), Text mining, Machine Learning, and Statistics. We also cover the working principles of the required technologies to implement a solution based on those concepts. The goal is to generate a solid theoretical foundation that can be referred to at different points later in this work.

2.2 Conceptual Architecture for Text Document Classification

The focus on this step is to arrange the concepts gathered in the previous step in such a way that it allows us to distinguish the function and contribution of each element to the intended solution. As such, we aim to build a conceptual architecture that abstracts from the knowledge base the relevant concepts needed to address our problem. In doing so, we define the solution space with the intention to better understand both the problem and the potential solutions. Because of this, the resulting conceptual architecture is a model artefact (Hevner et al. 2004).

2.3 Data Exploration

At this stage of the work, the focus is to discover the properties, patterns, and assumptions that best describe domain-specific unstructured text data, particularly in the context of our application scenario (see section 4.3). To do so, different visual and statistical techniques are applied to a sample dataset of up to 7500 data points provided by an automotive company (OEM).

This dataset is analysed from the perspectives of the roles involved in the process that created it (Functional organization). They are the Supplier, Mechanic and the OEM itself. Inside of every role, we examine the data as a collection of data points and as a set of text documents.

The goal of this step is to delimit the characteristics that characterise our particular object of study. This would provide the necessary evidence to identify the specific problems to be considered in this work and constitutes the first step to define a list of "empirical" requirements (as opposed to the theoretical aspects covered by the conceptual architecture) to develop a solution.

2.4 Study Object Characterisation

Based on characteristics collected during the data exploration, this stage aims to characterise the study object within the context of our application scenario. Starting from the original definition of messy data by (Kassner & Mitschang 2016), this step focuses on bringing evidence for the characteristics present in the original definition as well as for the new ones introduce as part of the characterisation to justify their inclusion, identifying the underlying theoretical concepts that underpin the characteristics identified during the data exploration results, and framing this new definition within our research model.

2.5 Method to Select Optimal Classification Algorithm Configuration and Features

In this step we design an appropriate method artefact that guides the creation of an instantiation artefact to search for an optimal solution. Such a method considers all possible factors that can affect accuracy and processing time according to our research model and conceptual architecture. The goal for this method is to obtain evidence on why is a configuration and feature set combination better than others.

We focus on creating a list of requirements for the classification algorithm that could in theory best fulfil the goal and metrics described in sections 1.2.3 and 1.2.4. This is then used as reference to evaluate some candidate algorithms in order to select one to be tested with an instantiation artefact and a series of experiments.

2.6 Experiments

To evaluate the utility of the conceptual architecture and method to help design solutions to our research problem, in this step we build and instantiation artefact with the selected classification algorithm. We do so by applying our method artefact starting from the selection of the algorithm itself, and continuing with the design of the algorithm configurations, the execution of each configuration as an experiment, and conclude by presenting the results. We use our application scenario data to test its performance, in terms of accuracy and elapse time, so as to render it comparable to the previous k-NN implementation.

2.7 Evaluation

In the final step, we proceed to evaluate the results of running the experiments on our instantiation artefact as well as the other two artefacts. As a consequence, our evaluation comprises two levels. Firstly, we focus on the evaluation of the instantiation artefact in terms of the goals, questions and metrics proposed in sections 1.2 and 1.3. Secondly, we proceed to evaluate the other artefacts (conceptual architecture and optimal method) in the terms of the Design Science Evaluation Methods (see sub section 1.4.4).

2.8 Limitations

Along the process of developing this work, there are certain areas that are considered out of scope due to their distant relation to the research model or because of the magnitude of work they would entail. They are mentioned in this section as a way to delimit the scope.

The main interest in this thesis is to explore effects and relationships of certain variables on end results, not to implement software components meant for others do that explorations. As such, we aim to reuse existing (open source) tools and components whenever possible. This not only speeds up the prototyping process, it also makes the exploration rely on proven software.

Despite of this, it would be innocent to assume that existing software for text mining, natural language processing, machine learning or statistics is free of errors. Since the early stages, this was evident with language detection components. A sort of classification problem on its own, improving the performance of this kind of component is beyond the efforts of this work. Instead, we simply mention the cases when the performance of language detection affects the execution of our own analysis (e.g. by losing reports).

A similar situation occurs with spelling mistakes. While it is clear that this represents a source of error in text classification, the implementation of a bilingual spell checker exceeds the reach of our goal.

Regarding the business environment in which our research problem exists, there are a set of constraints derived from it that affect the reach of our efforts. Although they are transparent for the most part of this work, they may become notorious when analysing the dataset on which we work. Therefore it is important to point out that the nature of this dataset, the assumptions under which it was created, and the conditions for its use are all given characteristics that cannot be altered.

Finally, concerning the Conceptual Architecture for Text Classification introduced in section 4.1, it is important to emphasise that this architecture is supposed to work as definition of the solution space for this problem and exhausting the possibilities it offers to design alternative solutions, even in the best of scientific interests is beyond the scope of this work. Still, we do present an instantiation that makes use of representative components of each layer as a way to prove its adequacy.

3 Theoretical Framework

In order to give theoretical foundations to the solutions proposed in chapter 4, a literature survey was conducted. An academic enterprise on its own, the exploration of existing work is organised according to an extension of the preceding work of (Kassner et al. 2014) (from which this thesis derives) with the overview on text classification by (Khan et al. 2010). This allows to have a more detailed conceptual framework that enables better-focused research of related work.

By looking at the study object characterisation in section 4.3, it is clear that there is more than one way to address this text-document classification problem and as a matter of fact, each candidate solution may require different disciplines to come to fruition. (Kassner et al. 2014) already recognise this fact and in their conceptual architecture they propose a modularised separation of activities to analyse structured and unstructured data. Specifically, they propose in their middle layer *Analyse* a two-level structured composed of *Core Analytics* and *Value-Added Analytics* as shown in Figure 5.

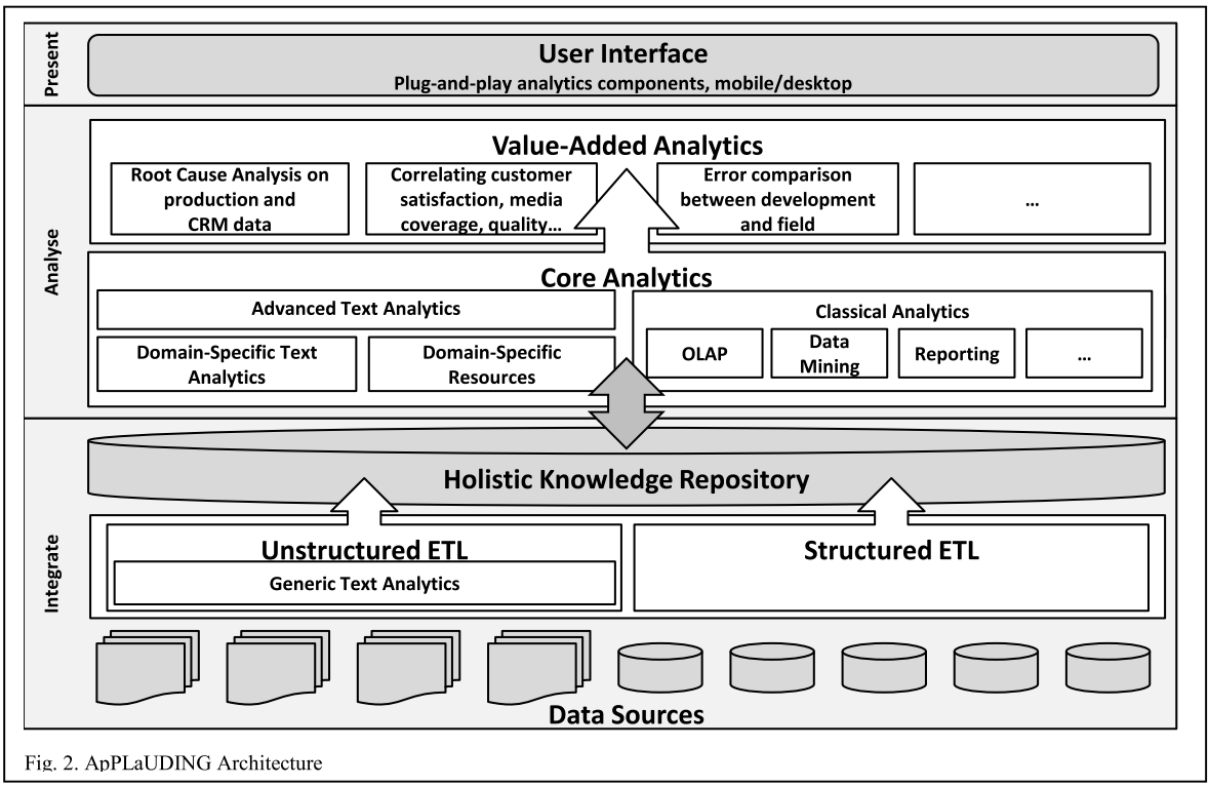


Figure 5 ApPLaUDING Conceptual Architecture (Kassner et al. 2014)

The upper level is meant to generate complex value-adding analytic capabilities based on the results obtained from a composition of modular core analytics components. Meanwhile, the lower level is concerned with tools for both unstructured and structured data, respectively depicted on the left and right sides of the layer.

This thesis focuses on the components for unstructured data of the Analyse layer, namely *Domain-specific text analytics*, *Domain-Specific Resources* and *Advanced Text Analytics*.

The first two elements are considered part of the same analytics toolbox, which uses "domain-specific NLP resources such as taxonomies, wordlists / dictionaries and schemas (…)" and domain-specific text analytic tools for "the recognition of entities or expressions from a particular domain (…)" (Kassner et al. 2014). The third element and toolbox, *Advanced Text Analytics*, "contains advanced analytics drawing on both the domain-specific resources and on analytics techniques from the domain-specific

toolbox." (Kassner et al. 2014). Examples for this are topic detection, and clustering and classification algorithms.

However as the enriched concept of messy data in section 4.3 highlights, there are equally important components involved in the generation of meaningful data and that are nonetheless not represented on the ApPLaUDING architecture. Prior to any analytics activity, there are certain pre-processing steps that are needed to handle unstructured data. Examples of these are: tokenisation, fundamental to transform unstructured text into a feature vector and language detection, to be able to apply other pre-processing steps like stop word removal or stemming correctly.

At the same time, (Khan et al. 2010) suggest that in every text document classification endeavour, three disciplines are required: Text Mining, Natural Language Processing, and Machine Learning. Even more, regardless of the technique employed, they tend follow a particular order.

Text mining begins with the application of two kinds of methods: Information Extraction and Information Retrieval. The first one is meant "(…) to extract specific information from text documents." while the second employs statistical methods "for automatic processing of text data (…)" (Khan et al. 2010).

Natural Language Processing aims to analyse the data on a syntactical level so as to improve the classification process and to enable the usage of taxonomies or similar complementary resources, such as ontologies. Properly speaking, syntactical analysis aims to parse "sentences and paragraphs into key concepts, verbs and proper nouns." (Khan et al. 2010).

Finally, Machine Learning involves all supervised approaches to document classification. This is particularly useful for our study subject since "supervised learning techniques are used for automatic text classification, where pre-defined category labels are assigned to documents based on the likelihood suggested by a training set of labelled documents." (Khan et al. 2010)

If we then expand the original architecture by (Kassner et al. 2014) in order to distinguish the participation of the three above mentioned disciplines, we find a again a three-layered structured instead of the original pair of Core and Advanced Analytics. This is represented in Figure 6.

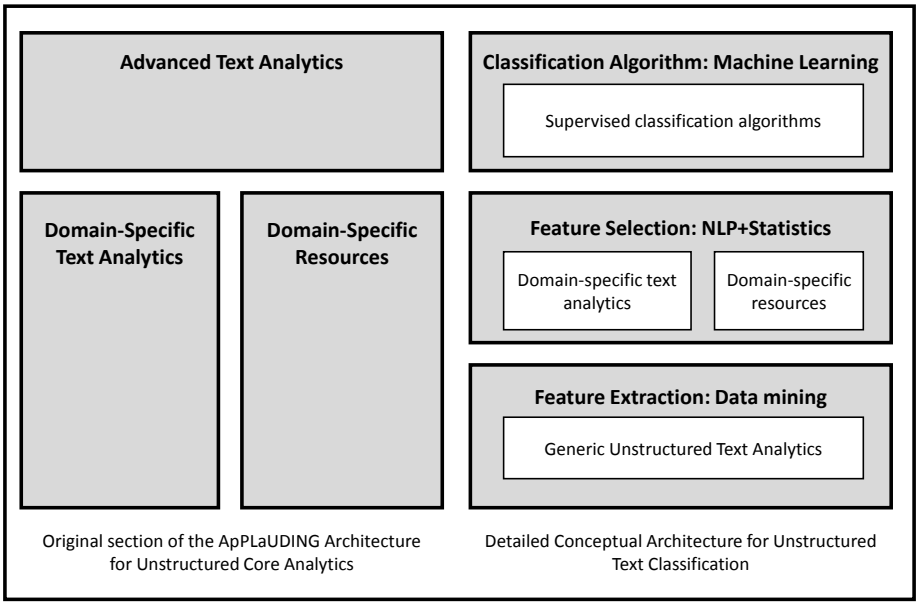


Figure 6 Comparison between the original and detailed conceptual architectures. Adapted from (Kassner et al. 2014)

This detailed conceptual architecture divides the classification of text documents into three layers. The first one consists of feature extraction techniques. This is therefore related to knowledge from Data

Mining, in other words, the pre-processing techniques that can be applied to "make clear the border of each language structure and to eliminate as much as possible the language dependent factors, tokenization, stop words removal, and stemming." (Khan et al. 2010).

Then, the feature selection layer is used "to select subset of features from the original documents (…) by keeping the words with highest score according to predetermined measure of the importance of the word." (Khan et al. 2010). This is intended to make the existing set of available features manageable and scalable so that the selected classification algorithm can be applied on the dataset. To achieve this, it is composed of Natural Language Processing and Statistics components.

Finally, the classification algorithm layer contains Machine Learning algorithms which can be selected according to their performance when handling certain kinds of data.

The following sections in this chapter present the theoretical concepts behind the (potential) components in each of these three layers.

3.1 Feature Extraction Mechanisms

In this section we refer to the different steps that transform unstructured text data into more structured formats that can be later used by algorithms to analyse it. In this respect, we consider these steps as techniques from text mining. In this context we understand text mining from the perspectives of Data Mining and the Knowledge Discovery Process as described by (Hotho et al. 2005). As such, we see text mining simply as a process with sub components that perform certain pre-processing tasks to extract useful patterns for document analysis.

3.1.1 Term

Also known as feature, it represents the smallest meaningful unit of text with which a document can be represented. Depending on the approach taken, a term can refer to a word or a phrase (Sebastiani 2002). According to (Khan et al. 2010), other alternatives include N-Gram and RDR, this last one representing document as logical predicates (Khan et al. 2010). On the other hand, an N-Gram is "(…) a string-based representation with no linguistic processing" (Khan et al. 2010).

3.1.2 Feature Vector

This is the cornerstone of the Vector Space Model (VSM). It represents a text document as a vector composed of term weights, is a feature vector (Khan et al. 2010). When all distinct terms *m* from all documents to be analysed are merged into a single collection, this is called the dictionary of the document collection (Hotho et al. 2005). This dictionary represents the *m*-dimensional space in which feature vectors are expressed (Sebastiani 2002). Unfortunately, it is often the case that the amount of features considerably outnumber the number of documents available, something known as high-dimensionality (Khan et al. 2010).

This high dimensionality brings many challenges when it comes to text classification, such as prolonged execution times both in training and testing, the need for bigger samples to train the algorithm and difficulties in visualising the dataset (Rafeeque & Sendhilkumar 2011).

3.1.3 Bag of Words

A model to represent text documents for text classification in which every word is considered a term (Sebastiani 2002). Each position of a feature vector contains the occurrence of a word, with the total amount of words usually overcoming the total amount of training documents by more than an order of magnitude (Forman 2003). It does not preserve the semantic context (Rafeeque & Sendhilkumar 2011).

3.1.4 Bag of Concepts

A text document representation model based on the identification in text documents of mentions to domain-specific concepts (Kassner & Mitschang 2016). It requires the mapping of words in text to concepts stored in a semantic resource via named entity recognition (Kassner & Mitschang 2016). For

this to work, the semantic resource should be aware of synonymy relationships among its concepts (Kassner & Mitschang 2016).

3.1.5 Term-Weighting Techniques

They comprise the different ways to calculate the contribution of a term (or feature) to the semantics of a document, or to the semantics of the whole document collection (Sebastiani 2002), in other words, the representation of a term's value (Forman 2003). In their most basic form, weights reflect the presence of a term within a document, but different approaches may use take into consideration other aspects, depending on the needs of a classification algorithm (Sebastiani 2002). According to (Forman 2003), this basic form is a binary representation, which should be enough for short texts since terms hardly repeat. It also enables the use of other feature selection metrics, such as Odds Ratio.

For the cases when weights are not binary, (Bank 2013) discusses three common components that can be used, namely the local, global and normalisation components. The first one refers to the importance of a term in the document where it is contained. The second is based on the importance of the document in the whole document collection. The third component is employed to negate the influence of very different document lengths in the weight of a term.

A technique with focus on the local component is term frequency, understood as the number of times *N* a term *i* is mentioned in document *j* (Ruotsalo 2012), expressed in the formula:

$$tf_{i,j} = N_{i,j}$$

For the global component, inverse document frequency is a common way to calculate it. It is based on the number of documents *n* where term *i* appears in the document collection *N* (Ruotsalo 2012), expressed in the formula:

$$idf_i = \log(\frac{N}{n_i + 1})$$

These two components combined derive in the creation of a major calculation method known as TF-IDF (Sebastiani 2002) with the formula:

$$tfidf_{i,j} = N_{i,j} \cdot \log(\frac{N}{n_i + 1})$$

TF-IDF captures two notions. The first one is that the more often a term appears in a document, the more it represents the content of the document. The second one is that the more often a term is present across multiple documents, the less useful it is to discriminate among them (Sebastiani 2002). The TF-IDF approach then weights terms according to how unique they are among terms, documents and particular categories (Khan et al. 2010). It is important to note however, that his formula does not take into account the order in which terms may appear inside of documents (Sebastiani 2002).

A variant of the TF-IDF formula also considers the thirds normalisation component identified by (Bank 2013) in the form of a cosine normalisation, which is the square root of the sum of square TF-IDF weights for all terms *i* from 1 to *m* as shown in the formula:

$$\sqrt{\sum_{i=1}^{m} (tf_{i,j} \cdot idf_i)^2}$$

This then turns the TF-IDF formula into the expression

$$tfidf_{i,j} = \frac{N_{i,j} \cdot \log(\frac{N}{n_i + 1})}{\sqrt{\sum_{i=1}^{m} (tf_{i,j} \cdot idf_i)^2}}$$

3.1.6 Document Term Matrix

A Document Terms Matrix (DTM) is the aggregation of all document vectors describing the term frequencies of all terms considered in the document collection, also known as the collection dictionary. It contains document IDs as rows and terms as columns, and every resulting intersection contains

weighted-term frequency of term *i* in document *j*. It is one of the most common ways to represents texts in text mining (Feinerer et al. 2008).

As (Damljanovic et al. 2012) mention, there are issues concerning the scalability of DTMs with large corpora. This is due to the fact that the matrix grows every time additional documents or terms are added.

3.1.7 Tokenization

The process of breaking a text document into a sequence of words or terms, each of them separated by nothing else than a whitespace (Hotho et al. 2005). It is fundamental preliminary step for text mining that renders the document in a compact format in order for subsequent tasks to be performed on it (Khan et al. 2010).

If certain components require it, this process can also partition documents into sentences instead of doing it by words (Khan et al. 2010). However, this document representation has not yielded significantly better performance (Sebastiani 2002).

3.1.8 Stop Word Filtering

A processing task to remove stop words, which are words with little significance for text classification, because they are both too frequent across documents and they have no discriminating effect across documents (Hotho et al. 2005). Examples of these stop words are articles, conjunctions and prepositions. The goal of doing this filtering is to reduce the size of the dictionary and as a result mitigate, albeit partly, the typical high dimensionality problem (see sub section 3.1.2).

Stop words can be both language-specific and domain-specific (Forman 2003).

3.1.9 Stemming

It refers to the process of reducing words to their morphological root (Sebastiani 2002). This root, also called stem, is the one that groups words with equal or very similar meaning (Hotho et al. 2005). Since all words from a group are now represented by their stem, the overall number of terms present in a document decreases, which in turn helps address the problem of high dimensionality (see sub section 3.1.2). As it can be inferred, this process requires knowledge of the language used in the document.

3.1.10 Part Of Speech Tagging

One type of linguistic pre-processing methods (Hotho et al. 2005) that tags every term in a document with the role they perform as part of the speech, the so called Part of Speech tag.

3.1.11 Named Entity Recognition (NER)

An information extraction task that deals with the identification of entities in natural language text and their classification according to their entity type (Freire et al. 2012). The identification of entities usually requires identifying one or two adjacent terms that textually refer to them (Schierle 2011). The possible categories typically considered are people, organizations, locations, expressions of time, quantities, etc. (Freire et al. 2012). This implies that the identification of domain-specific entities is normally outside of the scope of classical NER approaches and needs to be implemented on its own (Hänig 2012).

This fact already hints that for a NER system to be successful, there is a strong need for manually created rules, manually created dictionaries or manually labelled training data, which makes the implementation in new domains or languages complicated (Schierle 2011). Current solutions already reach near human performance when applied to grammatically well-formed text (Freire et al. 2012).

3.1.12 Concept Recognition

Beyond the identification of entities in textual data, in domain-specific scenarios it is also necessary to identify the relevant concepts. This is due to the fact that concepts may be contained in more than nouns. Actions or properties can be expressed in adjectives, verbs and adverbs which are not identified by NER systems (Schierle 2011; Schierle & Trabold 2008).

Therefore concept recognition is a task focused on identifying concepts (entities, actions, properties or symptoms) in domain-specific textual data. This can involve the identification across different languages (despite of a potential lack of one to one term mappings), handling of synonyms and word sense disambiguation (Schierle & Trabold 2008).

Because of this transformation of the different terms referring to the same concept into a single identifier, Concept recognition can be seen as another kind of document representation that lies between phrase and single word representations.

3.1.13 Content and Function Words

Two major lexical classes to group words (Pulvermüller 1999). Content words (or open class words), which usually refer to more concrete meanings, include nouns, verbs and adjectives. Function words (or closed class words) can include articles, pronouns, auxiliary verbs, conjunctions and in general any word that contributes to the meaning of sentences by fulfilling a certain grammatical purpose, (Pulvermüller 1999).

3.2 Feature Selection Strategies

Defined by (Dasgupta et al. 2007) as the process of "selecting a subset of the features available for describing the data before applying a learning algorithm", the main focus is to reduce the number of features extracted from a document collection to a number that can be managed by a classification algorithm. While traditionally there are two main types of feature selection methods, wrappers and filters, in this work we focus only in the latter. This is not only due to the fact that wrappers are in general not suitable for text classification (Khan et al. 2010), but also, as (Blum & Langley 1997) mention, because filters are independent of the algorithm that will use their output, which turns them into ideal methods for our proposed conceptual framework.

3.2.1 Dimensionality Reduction

A preliminary step before applying a classification algorithm, it reduces the amount of terms that comprise the feature space so that algorithms do not face high-dimensionality issues (see sub section 3.1.2). The new set of features is called a reduced term set (Sebastiani 2002).

Besides of enabling algorithms to handle bigger feature sets, performing dimensionality reduction also helps to avoid overfitting the classification model due to having a small amount of training data (Sebastiani 2002).

There are two main ways to perform dimensionality reduction: either by selecting a subset of the original term space (term selection) or transforming the original terms to obtain fewer (and new) ones (term extraction) (Sebastiani 2002). Within the term selection techniques we can find wrappers and filters, whereas in the term extraction approach Term Clustering and Latent Semantic Indexing are good examples (Sebastiani 2002).

3.2.2 Wrapper

It consists of creating successive new term sets, either by adding or removing terms to the original term set, to apply the classifier algorithm until the most effective set is found. It implies using the same algorithm for both the classification and term selection (Sebastiani 2002). As a consequence, this approach is time consuming when the number of original terms is very high (Khan et al. 2010).

The logic behind using the same algorithm for both tasks is that in this way, the effectiveness of the resulting reduced term set is guaranteed, since it was calculated with the actual algorithm that will perform the classification, instead of using any other measure geared towards other purposes (Blum & Langley 1997).

Common algorithms used in the wrapper approach are Naïve Bayes and k-Nearest Neighbours (Blum & Langley 1997).

3.2.3 Filter

It refers to a feature selection approach where a mechanism different (and independent) from the intended classification algorithm is used to subset the total number of features (Blum & Langley 1997). The filter decides which terms to keep based on a feature scoring metric that assesses the usefulness of the term for the classification (Khan et al. 2010).

A simple metric can be term frequency, which can for example keep only the most frequent terms. This seemingly simplistic action can nonetheless achieve a reduced set 10 times smaller than the original one with no loss in effectiveness, given that stop words are first filtered (Sebastiani 2002). Other ways to select terms based on term frequency is to remove terms with minimal occurrences over the whole training set, or those appearing in a minimal amount of documents (regardless of how many times they appear inside the document) (Sebastiani 2002).

3.2.4 Feature Selection Metrics

They measure the ability of a feature to help differentiate the target classification categories (Khan et al. 2010). While originally conceived as the core of filter approaches to term selection, they can also be used as heuristics to improve the performance of wrapper methods (Forman 2003).

Common metrics according to (Forman 2003) are Chi-Square, a measure of divergence from a statistic distribution (thus prone to failure with small frequencies); Information Gain, measuring the decrement in entropy when a feature is considered; Odds Ratio, measuring the chances a term appears in one category over the chances of appearing in other categories; and Document Frequency, or how many documents contain a word.

3.2.5 Language Statistics

These are statistical measures that help characterise and understand language datasets (or corpora) (Bank et al. 2012). With them, datasets used in research projects can be compared in a fair manner, and by assessing their differences, it is possible to evaluate the transferability of the natural language processing methods or algorithms applied on them.

Among the statistics proposed by (Bank et al. 2012), we focus on four: Shannon's entropy, relative vocabulary size, vocabulary concentration, and vocabulary dispersion.

Shannon's entropy H for language engineering represents the mean amount of information of a term t_i. High entropy means there are many words with low frequencies. It is given by the formula:

$$H = -\sum_{t_i \in V} p(t_i) \log_{|V|} p(t_i)$$

Where V is the vocabulary size (all terms comprised in the dataset) and $p(t_i)$ is the probability of the term in the corpus. Moreover, (Hofmann & Chisholm 2016) estimate the probability of terms following a power-law distribution according to the formula:

$$p_r \approx [rln(1.78N)]^{-1}$$

Where p_r is the probability of the word in rank r (the typical frequency-based rank used for example in Zipf-plots, see sub section 3.2.6) and N is the total number of terms.

The relative vocabulary size R_{Voc} is a ratio of the vocabulary size V over the total number of occurrences of meaningful words N_m, where N_m refers to words that are not function words (see sub section 3.1.13):

$$R_{Voc} = \frac{|V|}{N_m}$$

Vocabulary concentration C_{Voc} is understood as the ratio of the number of occurrences of the most frequent terms in the vocabulary N_{top}, over the total number of occurrences of all terms in the dataset N:

$$C_{Voc} = \frac{N_{top}}{N}$$

The vocabulary dispersion D_{Voc} expresses the ratio of terms with low frequency V_{low} (terms whose number of occurrences is less or equal to 10) over the vocabulary size V (total number of terms in the dataset):

$$D_{Voc} = \frac{|V_{low}|}{|V|}$$

3.2.6 Power-Law Distribution

It is a type of cumulative distribution that is commonly found both in natural and man-made systems, including the frequency of words used in human language (Newman 2005). In it, the distribution of the quantities being measured, in this case the frequency of words, is proportional to the rank of the word (Newman 2005). We can express this in the formula:

$$P(x) = Cx^{-a}$$

Where $P(x)$ is the fraction of words with frequency greater or equal than x, x is the frequency with which a word occurs (or a quantity in general), and both C and a (exponent or scaling parameter) are constant parameters that are estimated in a case by case basis. Also $C = e^c$. It is important to notice that the estimation of the exponent a requires choosing a minimum x value above which the power law is valid. This points to the fact that in real life scenarios, distributions often present power-law behaviours in certain ranges, and not across the whole dataset (Newman 2005).

Power-law distributions can be visually represented in different ways. The most common include the rank/frequency plots or CDF plots, which include Zipf-law plots (with P(x) on the y axis) and Pareto-distribution plots (P(x) on the x axis); histograms on logarithmic scales, or a simple histogram (Newman 2005). Examples of these are found in Figure 10 for the simple histogram and Figure 40 for the Zipf plot.

It is also called a scale-free distribution due to the fact that regardless of the units in which x is measured, the power-law distribution remains present, albeit with a change of value at the constant C(Newman 2005).

Power-law distributions can easily induce high-dimensionality problems over time as data grows and frequencies of very rare terms increase, even if it is in marginal levels. This is of particular relevance in scenarios like the one that (Liu et al. 2013) point out, where multi-category classification is performed on an open text collection (new items are added over time). They also highlight that in real-life applications, this represents a challenge in terms of storage-time-cost sensitivity that needs to be controlled. After all, in many scenarios many features will be useless for classification because of their comparatively low frequency, thus supporting the need of reducing the total number of features. They propose a method that takes advantage of the power-law distribution to achieve this with low storage, computing time and cost.

For this (Liu et al. 2013) define the uselessness ratio R_u "as the ratio of the number of token features with less frequency to the total number of token features", where less frequency is considered to be less or equal to two. The complement to R_u is then the random sampling ratio R_{rs} that can be used to reduce the size of the power law distribution without altering its overall distribution.

Additionally, very frequent terms can also present an obstacle for proper classification. (Cavnar et al. 1994) point out that a power-law distribution implies the dominance of a small set of words in a given language both in general and in particular subjects. Empirically, they discovered that around the top 300 terms in a language, there is a high correlation among those terms regardless of the subjects covered in the composing texts. Beyond this point, terms are more specific to the subjects of each document. While they discover this around the 300th rank for a collection of short texts, they mention that this tipping point was discovered manually, and could change for other collections.

Finally, (Newman 2005) and (Clauset et al. 2009) raise a warning to avoid identifying power-law behaviour in any distribution that graphically presents some exponential trend. This is particularly relevant since a very common way to check for power-law behaviour is visually inspecting a plot of term frequencies with a logarithmic scale in both axes (Clauset et al. 2009). When the distribution seems

to resemble a straight line, the dataset is considered to follow a power-law. However this offers erroneous results and inaccurate parameter estimations.

Instead, (Clauset et al. 2009) propose to estimate the scaling parameter with a maximum-likelihood method (MLE for Maximum-Likelihood Estimator) and the minimum x value with a Kolmogorov-Smirnov statistic (KS statistic or test), the most common statistic used for non-normal data. The MLE method is based on the Hill Estimator, while the KS test measures the distance between two distributions: the one fitted by the parameters and the actual data distribution. The adequacy of this estimations is then verified with a p-value test on the hypothesis of data being drawn from a different distribution. Lower values for the KS Statistic evidence better fit while values higher than 0.05 for the p-value reject the hypothesis of data being drawn from other distributions.

3.3 Classification Algorithms

3.3.1 Levels of Supervision

Machine learning algorithms can be classified according to the amount of preliminary human effort needed to use them. According to (Hänig 2012), this effort refers to the creation of a train input set and to the specification of the expected output (regarding for example the amount of categories). The supervision is then this initial labelled dataset (Gupta 2011). Depending on the amount of supervision needed, there are three distinct categories.

Supervised methods, which can achieve very accurate results when their conditions are properly met, require a fully annotated training set; this is an initial set of data with labels describing the particular attribute by which all data has to be classified (Hänig 2012). They also require to specify the number of categories in which data has to be classified. Their main drawback is precisely the amount of effort required to label the training set, which in some cases may not even be possible to do, because the data refers to subjects or phenomena no longer available. Traditional classification algorithms such as Naïve Bayes, Support Vector Machines, K-Nearest Neighbours and Artificial Neural Networks are supervised algorithms (Gupta 2011).

Unsupervised methods on the other side, are very easy to apply on new data sets because they neither require a previously annotated train set nor the specification of the output (Hänig 2012). On the positive side, this means that they are an efficient alternative to obtain some structure out of a new data set even if that structure is not that accurate (Hänig et al. 2008). Instead, they cluster data based on some measure of differences or distance between observations. This can lead to classifications that do not match the problem at hand, because they do not consider the categories in place, resulting in unpredictable results with both useful and useless patterns (Gupta 2011). Some examples of unsupervised algorithms are K-Means, Latent Dirichlet Allocation Topic and Expectation-Maximization for mixture of Gaussians.

The final category of algorithm incorporates characteristics of the previous two. Semi-supervised algorithms do begin with a set of labelled data. This is used to classify data unless a certain abort condition is met (Hänig 2012). The classified results can be then reused as additional train data if a human expert confirms the correctness of the classification (Hänig 2012).

3.3.2 Multiclass Text Classification

It refers to the process of labelling each document (written in natural language) in a document collection with a single category (or class) out of a set of predefined thematic categories (Giorgetti & Sebastiani 2003).

The process begins with a document corpus already labelled with categories from its predefined category set. This is split into training and testing sets to be used at different stages of the process. Afterwards, an algorithm, also called learner, builds a classification model for the target categories using the documents in the training set. The model's effectiveness is then measured by running classifying the documents on the testing set using the same model (Giorgetti & Sebastiani 2003). Effectiveness is calculated in terms of accuracy, understood as the proportion of correct classifications over the total number of classifications (Giorgetti & Sebastiani 2003).

3.3.3 Naïve Bayes

It is a classification algorithm based a probabilistic model that aims to select the category with the highest probability for a document based on the words the document has (Giorgetti & Sebastiani 2003).

Given a document training set or corpus *D*, where each document can be represented with a feature vector w_j, the probability of a document represented by vector w_j to belong to a category c_i is calculated by applying the Bayes theorem in the following manner:

$$P(c_i|w_j) = \frac{P(c_i)P(w_j|c_i)}{P(w_j)}$$

The assumption that gives this algorithm the quality of Naïve, and also simplifies the calculation probability, is applied to the calculation of $P(w_j|c_i)$, because normally the number of vectors (or documents) to consider is too high (Sebastiani 2002). It is simply assumed that each element of the feature vector (typically words) are independent from each other and from the order in which they appear. Instead, the document is considered a sort of "*bag of words*" without any contextual or semantic information about the feature vectors (Gupta 2011). This is represented in the following equation:

$$P(w_j|c_i) = \prod_{k=1}^{K} P(t_k|c_i)$$

Where t_k represents the k^{th} term *t* and c_i the i^{th} category.

Although this assumption can be considered unrealistic in particular for the text classification domain, the accuracy it yields, along with its ease to be implemented and its efficient computation use (Khan et al. 2010), has led Naïve Bayes to be a foundation algorithm upon which many improvements are proposed (Hotho et al. 2005). Moreover, this algorithm can be trained with a small amount of training data without affecting the performance of the classifier, proving its robustness despite of miscalculations in the probability model (Khan et al. 2010).

There are two common variants, the more performant multinomial, where all documents are considered a single document to do the calculations (Khan et al. 2010), and multi-variate Bernoulli method (Giorgetti & Sebastiani 2003).

Still it is worth mentioning that Naïve Bayes is not considered one of the best performers, especially when compared against the SVM algorithm. It also sees its performance reduced when features are highly correlated (Khan et al. 2010).

3.3.4 Support Vector Machines

A supervised machine learning algorithm that is commonly among the top performers in classification tasks. At its basic form, it is a binary classification model that aims to optimise the separation between two opposite category datasets, which can then be considered as positive and negative categories, each one requiring its own training data (Khan et al. 2010).

In the space in which document vectors are represented (see Figure 7), a hyperplane can be defined as the linear separation between the two classes for which the distance between itself (the hyperplane), and the closest elements from either class (distance called the margin), is maximised. In such space, documents are represented as vectors of real numbers (i.e. with term frequency weights) (Chih-Wei Hsu, Chih-Chung Chang 2008).

Documents located at the limits of the margin constitute the support vector. Once these are created, all training data not being part of the support vector can be removed without altering performance (Khan et al. 2010).

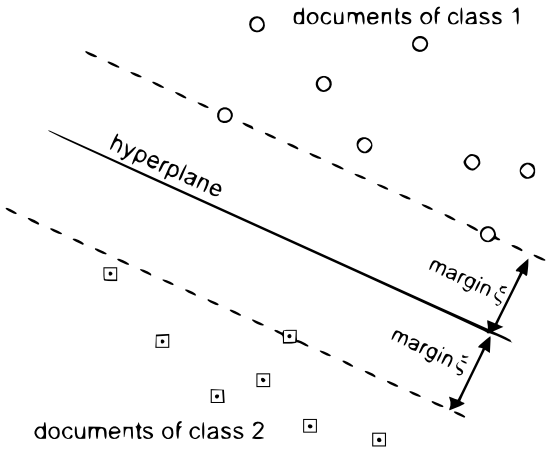


Figure 7 Example of Support Vector Machines Classification

Because of the linear representation it needs, finding the optimal hyperplane can be calculated in terms of a maximization problem for the Euclidean distance between the support vectors and the hyperplane (Hänig 2012). This is a "constrained quadratic optimization problem which can be solved efficiently for a large number of input vectors" (Hotho et al. 2005).

SVM was originally conceived as a binary classification algorithm, which means that it requires modifications to handle more classes (Zhang et al. 2011). These multi-class classification problems are approached by applying the algorithm as many times as there are classes (Hänig 2012). If the data is also represented in a highly-dimensional space, then using kernel functions allows the application of SVM by mapping the multidimensional space into a higher dimensional feature space where linear separation is possible (Hänig 2012). Options for kernel functions are linear, polynomial, radial basis function (RBF) and sigmoid (Chih-Wei Hsu, Chih-Chung Chang 2008). Another possibility to address this issue is to use a slack variable (Hänig 2012).

Even though SVM can indeed handle high-dimensional input, complex training and categorising algorithms are a problem (Khan et al. 2010). Another consequence of this complexity is the processing time of $O(N^2)$ for a training data set of size N makes it unsuitable for large datasets (Kyriakopoulou 2008). Also, in contrast to other algorithms, SVM reduces its effectiveness when feature selection techniques are applied to the dataset (Giorgetti & Sebastiani 2003).

3.3.5 Decision Trees

This classification algorithm builds its logic into a tree data structure where leaves are the classification categories and branches are sequences of selection tests that decide to which category a document should belong. Each document goes through a series of queries based on selected terms starting from the root node (Khan et al. 2010). To achieve this, algorithm is built using a "divide and conquer" principle (Hotho et al. 2005). Moreover, since most of the variants of this algorithm are based on binary document representations, and each node on the tree usually queries a single term (or feature), the resulting trees are binary (Sebastiani 2002). Therefore, they can also be seen as an organised set of if-then rules (Schierle 2011).

The creation of a decision tree starts with a set of labelled documents. From them, the term (or feature) that can better predict the documents' labels (categories) is selected to split the set into two groups: those with the selected term and those without it. This logic is recursively applied until all documents in a group (or subset) belong to the same category (Hotho et al. 2005). How to choose the first and successive features to continue building branches, the key step in this algorithm, is based on different measures, information gain being a common one (Schierle 2011).

Popular types of decision trees algorithms are Classification and Regression Tree (CART), ID3, and C4.5 (Murty et al. 2012).

Decision trees are commonly used in Data Mining because of their speed and scalability when it comes to the number of variables (features in our domain) and the size of the training set. However, these

advantages can become drawbacks for their performance in text mining, since they tend to employ only a small amount of the available features (Hotho et al. 2005). This results in poor performance to classify documents. However this can be overcome with the use of a few structured attributes (Khan et al. 2010).

Other challenges are its tendency to over fit the model to the training data, and the creation of overly complicated trees when the dataset is very large (Khan et al. 2010).

The main advantage of decision trees is its ease to be interpreted by humans (Sebastiani 2002), something that does not occur with probabilistic methods.

3.3.6 K Nearest Neighbours (k-NN)

An example-based classifier, it is also considered a lazy learner because of its lack of computation during the train phase, performing it all during the actual classification (Sebastiani 2002). In fact, training simply comprises storing documents as feature vectors along with their categories. The classification phase then computes similarities between all train vectors and the new vector (document) in order to choose the k most similar or "nearest" vectors. The most common category among those k vectors is allocated to the new document (Khan et al. 2010).

Based on this description, we notice two are the main steps in the algorithm, namely estimating the optimal k value and calculating the similarity between document vectors.

The optimal value for k can be obtained with additional training data using cross-validation (Hotho et al. 2005). It is important to note that this estimation should also take into account the number of classes and the size of the training set (Gupta 2011). Some authors argue the optimal values lie somewhere between 30 and 45, even though increasing the value does not significantly degrade performance (Sebastiani 2002). On large datasets, a classifier with k=1 has an error rate never larger than twice the optimal error rate (Hotho et al. 2005).

Similarity or semantic relatedness of documents can be calculated in multiple ways. Normalised count of common terms is one option (Hotho et al. 2005), others include cosine similarity, Euclidean distance, and Kullback-Liebler distance measure (Gupta 2011). According to (Sebastiani 2002) this measure can be probabilistic, or vector-based. In all cases, it is computed between a new document and all documents in the train set (Hotho et al. 2005).

This algorithm is known for its good performance in terms of accuracy and fast training phase (Murty et al. 2012). This is the case even for multi-categorised documents (Khan et al. 2010).

However, this does not come free of challenges. It takes a long time to be executed and is computationally intensive given the fact that it uses all features in distance comparison (Khan et al. 2010). Its performance is degraded with noisy or irrelevant features in the training data (Murty et al. 2012). Finally, the estimation of an appropriate K value is complicated if data is not evenly distributed or if there is noisy data (Murty et al. 2012).

3.4 Related Technologies

3.4.1 R

It is a functional programming language and environment for statistical computing and graphics distributed under a GNU-style copy left (R Core Team 2001). Besides of its core functionality on statistical procedures, R also has a package specification that allows to create purpose-specific modules. Thanks to this, it is possible to extend the scope of the R methods to unstructured text data. One such package that is relevant for text mining is the "tm" package.

The "tm" package provides a framework that integrates R statistical methods with advanced text mining or natural language processing methods from other toolkits, such as Weka and OpenNLP (Feinerer et al. 2008). This is done thanks to a modular design that can interface with the RWeka and Snowball packages to offer stemming, tokenisation, sentence detection and part of speech tagging.

"tm" is designed around a typical three-step text mining process including: 1)importing text and structuring it to be accessed in a uniform manner, 2) pre-processing text to obtain a convenient

representation (which may involve reformatting, whitespace removal or stemming), and 3) transforming texts in a format useful for computation like clustering or classification (Feinerer et al. 2008). The process starts with the creation of a corpus (or text document collection) as a data structure to manage documents in a generic way, and ends with the generation of a document term matrix on which computations can be performed.
Another package offering a stark different approach to supervised learning on text data is "RTextTools". This packages streamlines the process of pre-processing data, training several classification algorithms, performing the classification, comparing teach algorithm's performance, and exporting the results (Jurka et al. 2013). This package goes through a nine-step process starting with a document term matrix and finishing with a document summary to review accuracy of each of the nine classification algorithms available, namely Support Vector Machines, glmnet, maximum entropy, scaled linear discriminant analysis, bagging, boosting, random forest, neural networks and classification tree.

3.4.2 Weka

Standing for the Waikato Environment for Knowledge Analysis, it comprises a collection of machine learning algorithms and data pre-processing tools implemented in Java and released as open source software (Hall et al. 2009). It is built with a modular, extensible architecture that also provides an API and a graphical interface.

Because of its Java implementation, Weka requires a Java Virtual Machine with enough heap space, thus demanding to specify in advance the amount of needed memory. In addition to this, the amount specified also has to be less than the total amount of physical memory available to avoid swapping. These two conditions represent obstacles to its widespread use in practise (Hall et al. 2009).

It includes algorithms for regression, classification, clustering, association rule mining and attribute selection. Data exploration capabilities include data visualisation and pre-processing tools (Hall et al. 2009).

3.4.3 Unstructured Information Management Architecture (UIMA)

A middleware architecture initially developed by IBM, it is designed to support the creation of applications which process vast amounts of unstructured information with the use of structured data. The final goal of such applications is to extract relevant knowledge from data sources like natural language text, voice recordings, audio or video (Ferrucci & Lally 2004).

As discussed by (Ferrucci & Lally 2004), UIMA is meant to accelerate the creation of Unstructured Information Management (UIM) solutions by facilitating the integration of different technologies within a common framework. Moreover, it enables the reutilisation of existing software components, thus increasing the solution's flexibility as well.

At its very core, UIMA-based applications can be conceived as a sequence of *Analysis Engines* and or *Consumers* that perform different kinds of analyses on *documents* (units of unstructured information processing) and or *Collections* generating as a result a series of *Annotations*. A more structured overview of UIMA groups the previously mentioned components along with others into 4 different kinds of services: Acquisition, Unstructured Information Analysis, Structured Information Access and Component Discovery (Ferrucci & Lally 2004).

It is important to mention that originally, UIMA requires file descriptors for the creation of Analysis Engines where the input requirements, output specifications and external resources dependencies are specified (Ferrucci & Lally 2004). According to (Ogren & Bethard 2009) this is to be expected because of UIMA being a programming framework. However, in the long term this can become a burden due to the additional effort needed to maintain the descriptor files consistent with the code. To circumvent this problem, (Ogren & Bethard 2009) introduced what is known today as *uimaFIT*, a set of classes to instantiate, run and test UIMA components easily and without descriptor files. It is particularly useful to run *Pipelines* (sequences of analysis engines that process documents typically supplied by a Collection Reader) in a simplified manner.

4 Framework to Optimise Data Features and Classification Algorithms

In this chapter we design a framework to create solutions to the problem described in 1.2.2. To achieve this, we first refine the conceptual architecture for text classification presented at the beginning of chapter 3, specifically in Figure 6, to detail how specific concepts are used in the design of text classification solutions. We then explore the properties of our application scenario's dataset to characterise the concept of messy data to our particular context. With this enriched definition we then proceed to devise a method that supports the creation of text classification solutions both by making use of our conceptual architecture and being aware of the specific challenges present in this research problem.

In terms of the Design Science methodology, this chapter introduces two artefacts: a model (the conceptual architecture) and a method which are used to build an instantiation artefact in chapter 5.

4.1 Conceptual Architecture for Text Classification

There are multiple reasons to refine the detailed conceptual architecture introduced in chapter 3 and turn it into a reference model that can help us develop our own text classification solution. First, and as initially mentioned, with this detailed architecture we give equal importance to text mining pre-processing techniques in the development of text analytics solutions. This is important to stress the fact that the process of building text analytics solutions involves more than just parsing text and applying algorithms into it. It is also necessary to decide the way text is processed, represented, and even transformed to discover relevant features upon which algorithms can be applied. Underestimating this choices, especially for the unexperienced practitioner, can result into considerable performance degradations.

Second, by enforcing a clear distinction of three layers we clarify the way concepts from different disciplines are combined to develop text classification solutions. In this way, the architecture can help identify alternatives not only during development, but also when revisiting existing solutions by pointing at other components on the same level that are worth considering.

In this sense, the conceptual architecture describes more than component aggregations. It delimits the scope of possible solutions that can be developed to address our problem, something (Hevner et al. 2004) refer to as the *solution space*. Solutions derived from the application of this architecture can then be assumed comparable with each other, facilitating the evaluation of their respective performances in the search of the optimal solution. Because this architecture is inspired by the same analysis and research line that conceived the solution implemented by (Kassner & Mitschang 2016), comparisons can also be made with it.

Figure 8 shows the refined version of the conceptual architecture for text classification. It classifies components in three levels with the same amount of layers, without taking dependencies into account. This is to remain flexible and reusable while still addressing this and other domain specific analytics problems, one of the design goals conceived by (Kassner & Mitschang 2016). Components are thought as modules that can be combined horizontally and vertically to build a text classification solutions. The only constraints are consistency and the availability of features and structured data, thus encouraging the possibility of developing multiple solutions.

In a basic scenario, different components on the first two layers can be combined following the needs of a selected classification algorithm and the availability of the features in the dataset. As long as a combination of elements from every layer produce meaningful results, the solution is considered valid. It could also be possible to combine two or more algorithms and their necessary components from the layers above to come up with more complex solutions, but this is an alternative we do not focus on.

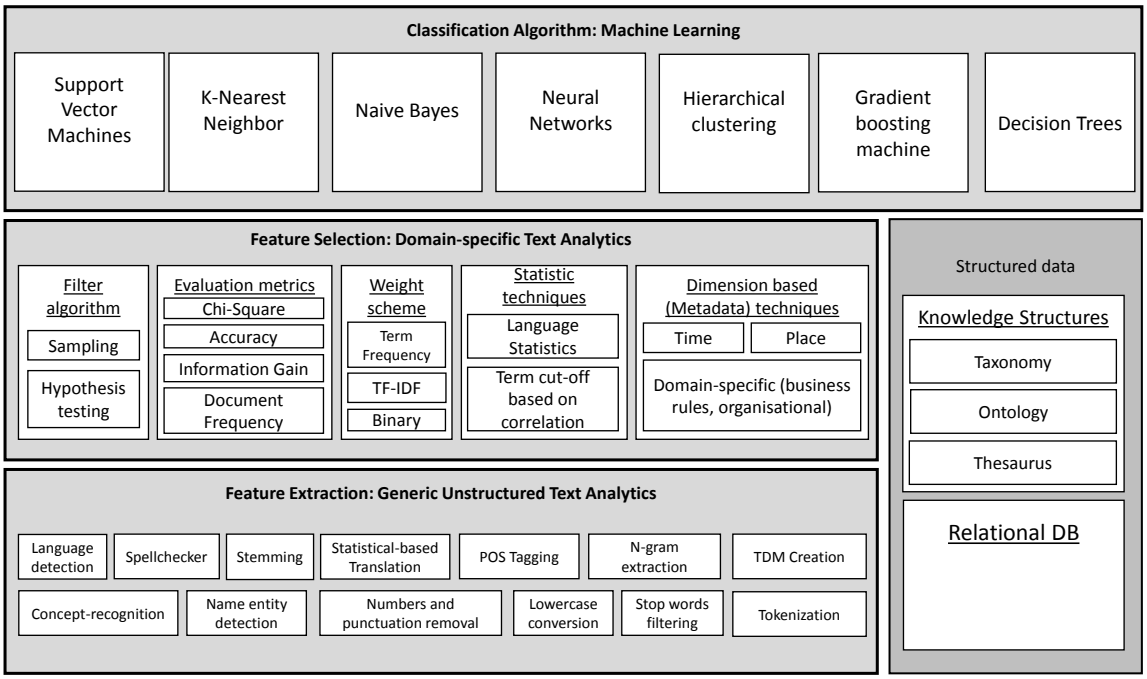


Figure 8 Full detail view of the Conceptual Architecture for Text-Document Classification

In this architecture three kinds elements are contemplated. On the bottom, feature extraction components refer to the generic unstructured text analytics that process the dataset to identify or help identify potential features in it to be later employed by a classification algorithm. This layer contains then components from Text Data Mining that can process unstructured textual data to identify terms, correct spelling mistakes, identify concepts, detect the language, etc. The objective is to obtain a set of potentially useful features contained in convenient data structures to be used by the following layer. An important result here is a Document Term Matrix. A structured data layer is at the same level because some of the components in the feature extraction layer may need access to structured resources to perform their task. For example, a concept annotator needs to access a taxonomy to identify the concepts in text.

The feature selection layer then focuses on choosing a subset with the most useful features out of all the ones previously obtained. This enables the selected classification algorithm to work more efficiently while still being able to describe the dataset in a reasonably accurate way. To fulfil this purpose, this layer addresses the need to specify an appropriate filter (see sub section 3.2.3), evaluation metric (see sub section 3.2.4), and weighting scheme (see sub section 3.1.5). The choice of including only filters and not wrapper methods (see sub section 3.2.2) is based on the design goal of maintaining every component in the layer modular and flexible to be used with as many different choices as it makes sense. Since wrappers are directly related to the selected classification algorithm, they are not part of our feature selection alternatives.

Based on the power-law nature commonly found in text, this layer also offers the use of statistic methods based on the properties of this kind of distribution (see sub section 3.2.6). Taking into consideration the fact that the data we study is framed in a domain-specific context (see section 4.3), the layer also contemplates selection alternatives based on other kinds of dimensions that can be relevant in the domain, like space, time or others based on business rules. As (Zhang et al. 2011) show and argue, this is possible because text contains many times information related to these dimensions albeit in an unstructured format. Examples of the information contained are names of places, spatial terms and certain POS elements like orientation words, prepositions and verbs (Zhang et al. 2011). This kind of techniques would infer the relevance of features based on, for example, how recently they were generated, how close to one another were the authors who created their source text reports or whether a pair of reports refer to similar car parts or not. Because of this, and in a similar way to the Feature Extraction Layer, the execution of certain elements on this layer depends on the use of structured data, this time to obtain relationships that can serve as input. Therefore, the structured data layer is also part of this level.

The top layer simply points out the existence of several classification algorithms that can be applied whenever an algorithm suits the target dataset's conditions, for example being able to handle the required number of classification categories. It should be noted that since there are multiple combinations of feature selection techniques and feature extraction components that can be applied to every classification algorithm, the elements on this top layer actually represent a family of *algorithm configurations* that share the same logic but apply it on different feature sets.

4.2 Data Exploration

The dataset we study comes from the quality inspection process described in section 1.2.1. It is a randomly sampled subset of the original dataset dealing with three major car part classes. It contains 7500 distinct *data bundles* or collections of text reports and structured data referring to an equal amount of car parts that went through the process and have an error code assigned. Aside from other irrelevant fields, the structured data that is created is shown on Figure 9. All data is stored in a relational database. Text reports, error descriptions and part description are written either in English or German. All data has been anonymised, so that no individual, organisation or vehicle can be identified.

Name	*Description*	*Type*	*Example*
Reference number	A 9–digits code to identify every part that is processed. Generated at the beginning of the process	Text	768192821
Error code	An 11-characters long code to identify the particular kind of failure the part suffered. Assigned by the OEM at the end of the process. This are the target categories for classification of reports. In the studied dataset, there are 1271 different codes.	Text	33107B61AV7
Part code	A 7-characters-long code to identify the type of part being analysed. It is part of the error code. There are 31 different codes in our subset.	Text	33107B6
Mileage	Distance in kilometres that the car had by the time it went to repair.	Numeric	13809
Production date	Date when the car was finished.	Date	2002-08-04
Admission-to-drive date	Date when the car is authorized to drive in the streets.	Date	2002-11-11
Repair date	Date when the car is taken to repair.	Date	2002-12-25

Figure 9 Relevant structured data for every car part

We focus our interest in the values of the error code, since they represent the categories in which the text reports have to be classified. Since the whole dataset is annotated with an error code, we can use part of it as a training test, and the rest as a testing set for our classification algorithm.

It is important to mention that not all *data bundles* contain the same information. When querying the database to obtain a bundles containing all reports and structured data fields, the amount of records (or rows) decreases up to 5538. This is due to the use of an INNER JOIN in order to obtain results with all fields. Instead of using a LEFT OUTER JOIN to retrieve incomplete results on certain fields (either missing text reports or structured data fields), we split the total dataset based on the roles involved in the process, namely the Mechanic, Supplier and OEM. By doing this we obtain datasets with 5624, 7182

and 583 observations respectively. In the case of the OEM we consider only those with the optional preliminary report, which is the only one available at the moment of classification.

Even though this decision reduces already the size of the initial dataset, it gives clarity about its composition with regards to the real-life context where it belongs. Moreover, it allows to attribute performance improvements or deficiencies to the quality of the data produced at each role, shedding light at how each role's contribution to the overall dataset should be treated.

As an initial step we do a visual exploration of each role's dataset with the objectives of: characterising the behaviour of each potential feature, finding patters in the interaction of different features, and determining the suitability of each feature for the classification task. Every dataset is extracted from the relational database and loaded into an R environment to explore. For each role we first look at its structured data and then at the contents of its reports.

4.2.1 Supplier Role Dataset

From the original 7182 observations retrieved, we begin by verifying the usefulness of the data for the classification task by looking at different criteria. Any observation that does not meet the requirements is then removed from the dataset.

We first remove 180 elements due to inconsistent or erroneous production or admission-to-drive dates (from years 1900 or 1997). Not only are these dates useless to locate the moment in time when these observations occurred, they also represent outliers that bring noise to the analysis of data trends over time. These elements with faulty dates constitute the totality of observations for 38 error codes. The remaining 7002 observations span across a 10 year period from 2004 to 2014.

In these observations, there are 709 error codes that only have a single occurrence. This makes them unsuitable for the classification, since it either adds a category to classify for which there is no way to test the accuracy, or it brings observations that can only create misclassifications. We then remove them from the dataset.

Still, the fact of having two occurrences of the same error code does not guarantee full utility for a classification task. The lack of enough elements of the same error code restricts the possibilities to perform statistical inference based on their structured data, a method that could otherwise help identify relevant features to improve classification performance. For example, 480 of the remaining 519 error codes have less than 30 occurrences. If we were to test significant differences of some structured data feature between samples of two error codes, such a small number of elements per error code would leave the t-distribution, especially designed for small samples (de Winter 2013), as the only option. Even then, with 426 out of those 480 error codes having very small numbers of occurrences (below 13, what (Johnson 1978) estimates enough elements for data with extremely asymmetrical distributions), the t-distribution tests would have problems not to produce false positives or false negatives. The reason is that for very small samples to produce accurate results, the effect of the variable (feature) involved has to be very large (de Winter 2013), and this is not guaranteed.

All in all, considering the significance of the loss it would represent to give up some many error codes, and the possibility to explore their utility for the classification task with other (less precise) methods, we keep these observations.

After these considerations, we start our exploration with 6293 observations belonging to 519 error codes. Figure 10 shows a little less than a third of all error codes ordered by the number of observations. Their distribution seem to follow a power-law.

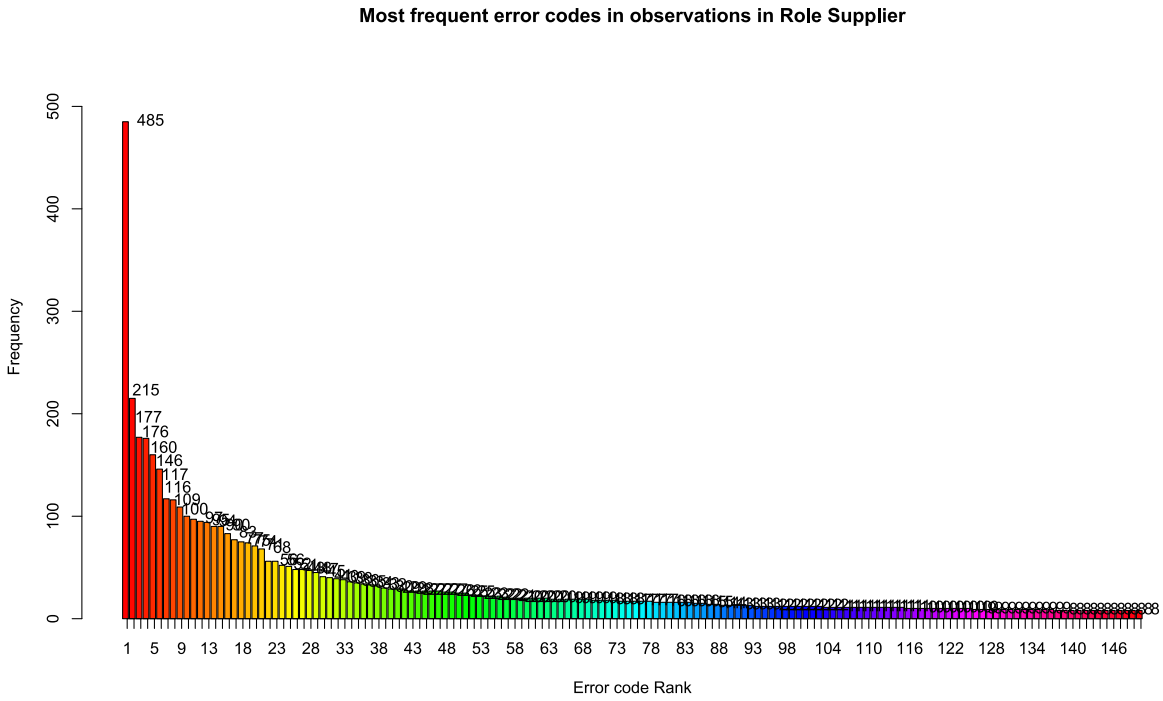


Figure 10 Plot of the 150 most frequent error code for the Supplier Role (filtered dataset)

We then test their fit using the KS tests (see sub section 3.2.6) and obtain a p-value of 0.4519 and a statistic of 0.0377. The first value rejects the hypothesis of the data being drawn from a distribution other than the power-law distribution, while the second one shows there is little distance between the fitted and the actual distribution. We can then conclude that for a scaling parameter of 1.8334 and a minimum frequency *x* of 2, the tests suggest the data indeed follows a power-law.

If we compare at Figure 11 and Figure 12, we see that after removing more than half of the error codes (which were in 12.37 % of the observations), the distribution of error codes per type of part (which we will error code families for clarity) does not change drastically. This suggests that the missing error codes were more or less evenly distributed across all part codes. It also contributes to validate the assertions made on the filtered dataset as a whole, since it resembles the original one. It is important to note, however, that three car parts are no longer represented, going from 31 original car parts, to 28.

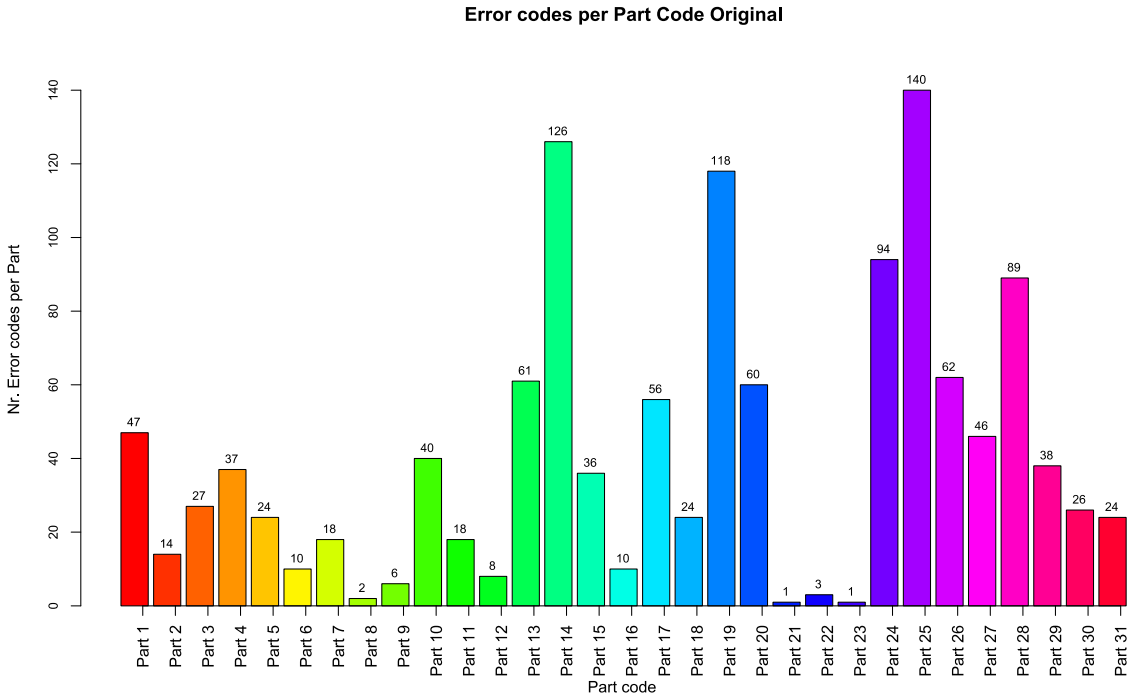


Figure 11 Distribution of error codes per part code in the original Supplier dataset

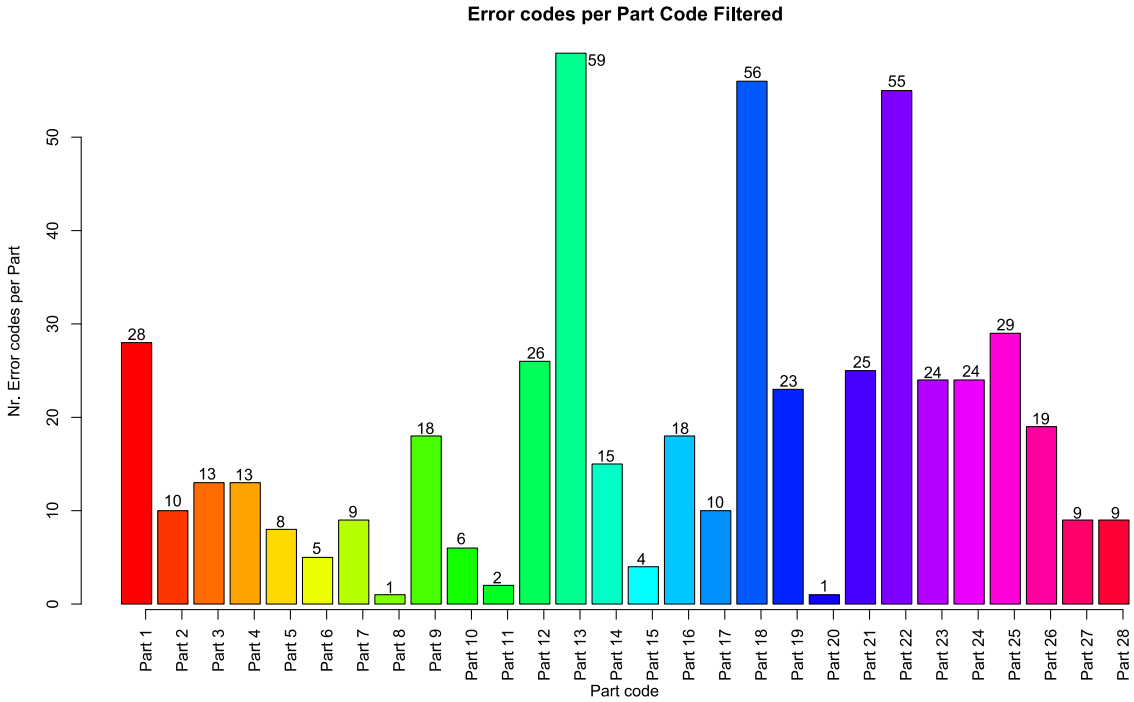


Figure 12 Distribution of error codes per part code in the filtered Supplier dataset

Looking at the size of error code families in the filtered data set, we see strong variations between them. From the perspective of the original process, this shows that some parts have many distinct ways to present failures, whereas others do not. This can represent a problem at the moment of evaluating the performance of the classification algorithm because not all car parts have the same amount of possible error codes to be assigned, thus making some accuracy values lack sense. After all, if a given part can

only have 6 different error codes (even in the original unfiltered scenario), how can we interpret accuracy in the 10 most probable error codes?

Focusing only on the filtered dataset, if we plot how many observations does every part have (shown in Figure 13), we see that their distribution is more or less similar to that of the error codes (Figure 12), except for a few notable cases. The leftmost part type has a lot of observations for its total error codes, averaging 36.5 observations per part, while "Part 22" has only an average of 7.49 observations per part. This suggests that some error codes will have very well trained models and, as a consequence, very good results while others will have more misclassifications because of the opposite scenario: very little observations and a lot of error codes. Similarly, this could represent a challenge if we tried to classify the observations of each part separately. In some cases the amount of data will be enough to build robust train and test sets, while in some others it will not. Moreover, it is likely that we would need more than one algorithm to perform the classification, just so we can address all the different behaviours present in every error code family. Averages for each part type are shown in Figure 14.

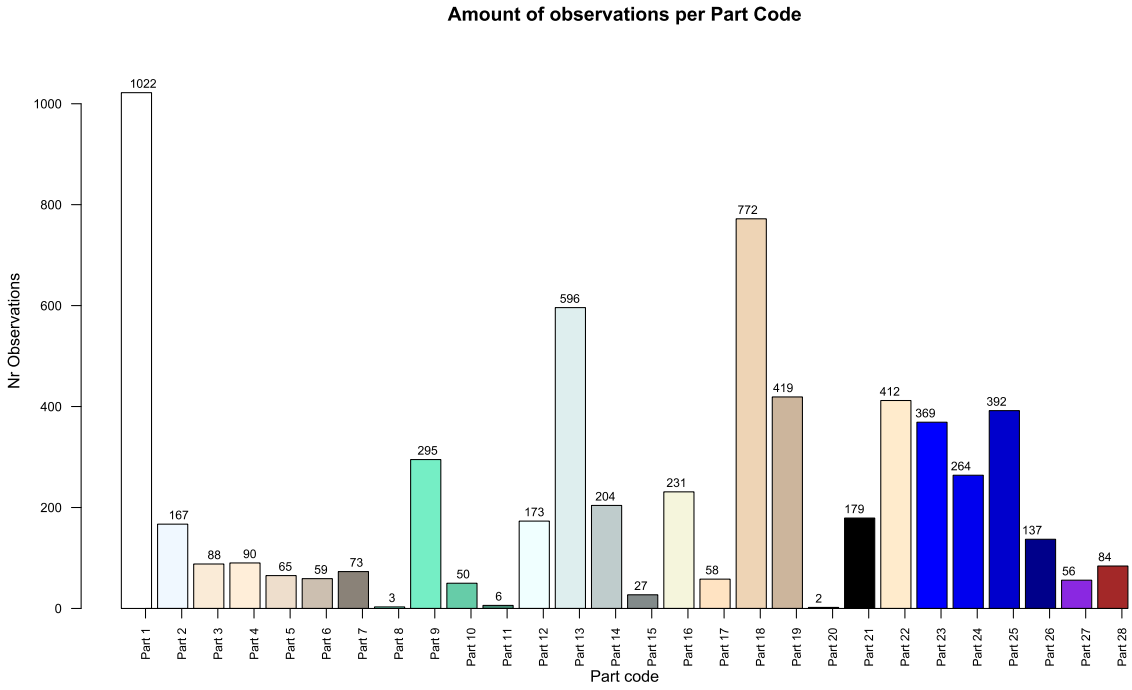


Figure 13 Observations per type of part for the filtered Supplier dataset

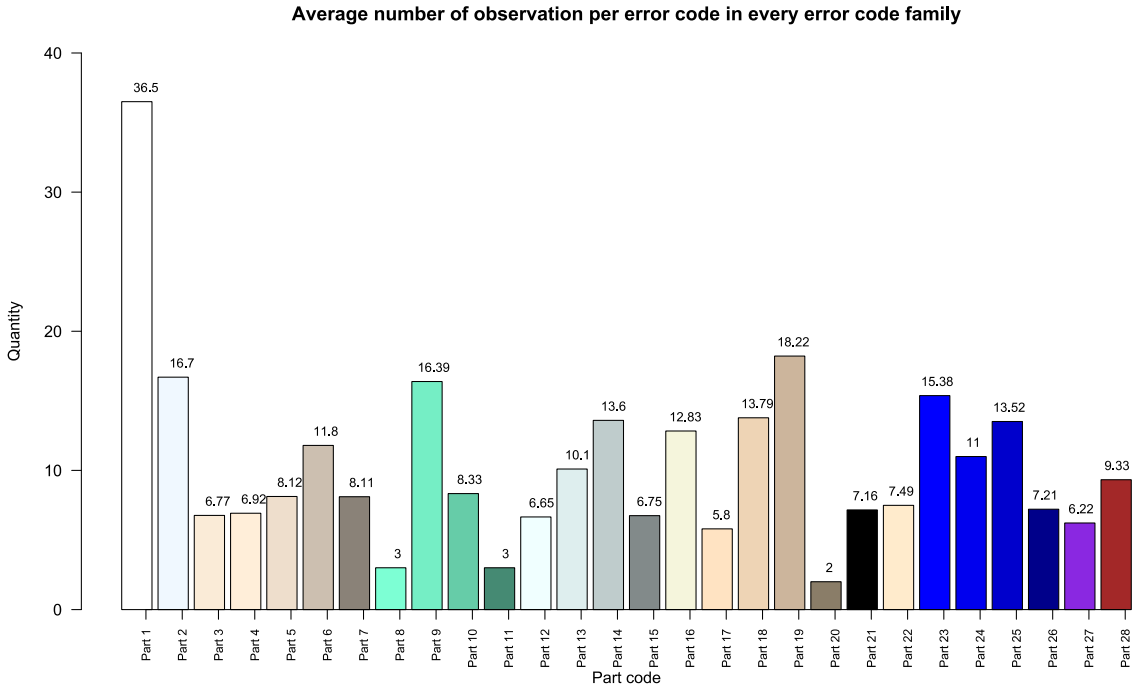


Figure 14 Average number of observations for every error code of each part type for the filtered Supplier dataset

4.2.1.1 Structured data in the dataset

We now take a look at different structured data fields that can be used to complement the features extracted from the text reports and as a result improve accuracy. We aim to find features whose values are heterogeneous enough to help discriminate among error codes.

4.2.1.1.1 Regarding mileage

To see the distribution of the mileage values we present these values as box plots in Figure 15. We see that for some part types there are many outlier values and a very short interquartile ranges (IQR) while for others the opposite is true. Yet in all cases the median values (thick lines inside the boxes) seem to remain low, somewhere beneath the 50,000 km mark. This is not particularly helpful, since across the range of values that mileage has, there does not seem to be a clear pattern to allocate a new observation to a particular part type. However, because of the scale distortion provoked by the presence of outliers, it is hard to determine how similar the median values really are. This is important because it indicates that half of the observations for every part type have values below that of the median. If these values are indeed different enough, it could serve as a good discriminating feature.

To verify this, we plot the median values in Figure 16. In this case it is clear that they progressively grow from almost 150 to a little more than 28000 km. We can expect difficulties in predicting to which error code does a part belong to, based on the mileage. Moreover, let us not forget these are median values which do not represent the full variability of mileage within every part type. As a consequence it is very likely that values close to the limits of the 1st or 3rd quartile in one part type (or error code family) could be mistakenly taken as belonging to another error code family where they also fit. As a conclusion, it can be expected to contribute very little to the overall performance of the classification algorithm.

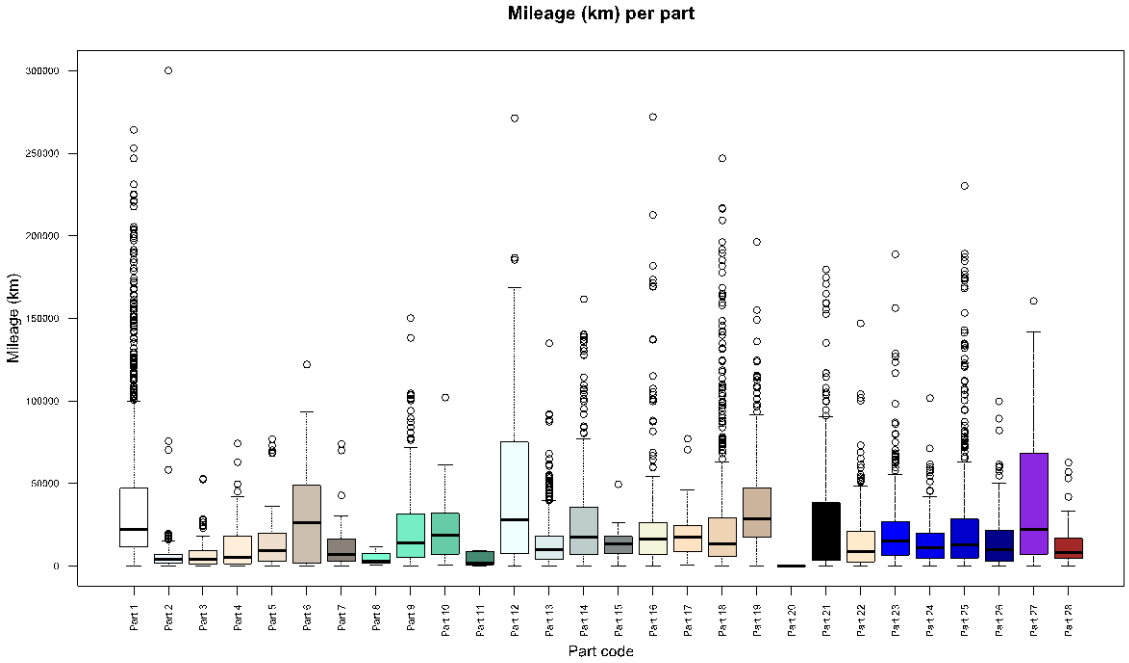


Figure 15 Box plots of mileage values per part type for the filtered Supplier dataset

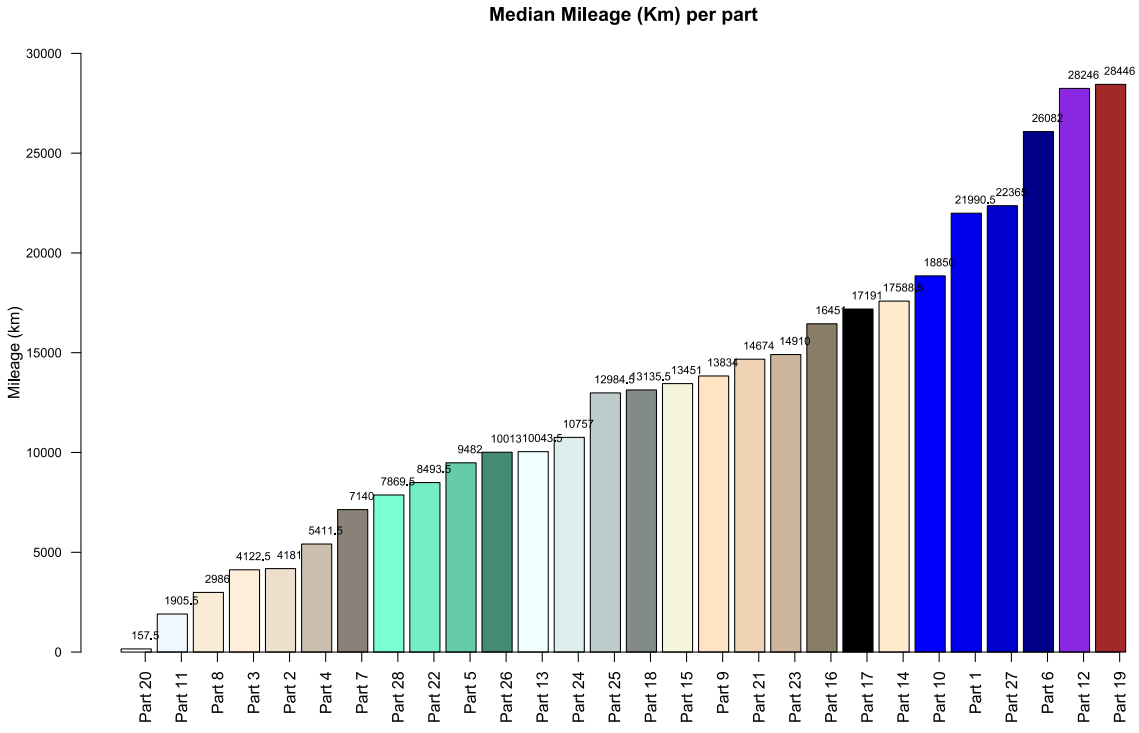


Figure 16 Ordered median mileage values per part type for the filtered Supplier dataset

4.2.1.1.2 Regarding time

Figure 17 shows the observations grouped by the 519 error codes considered in the filtered Supplier dataset on the y axis and grouped by the months in their repair dates on the x axis. Therefore, horizontal lines on the plot show the apparition of error codes during the whole time considered in our dataset.

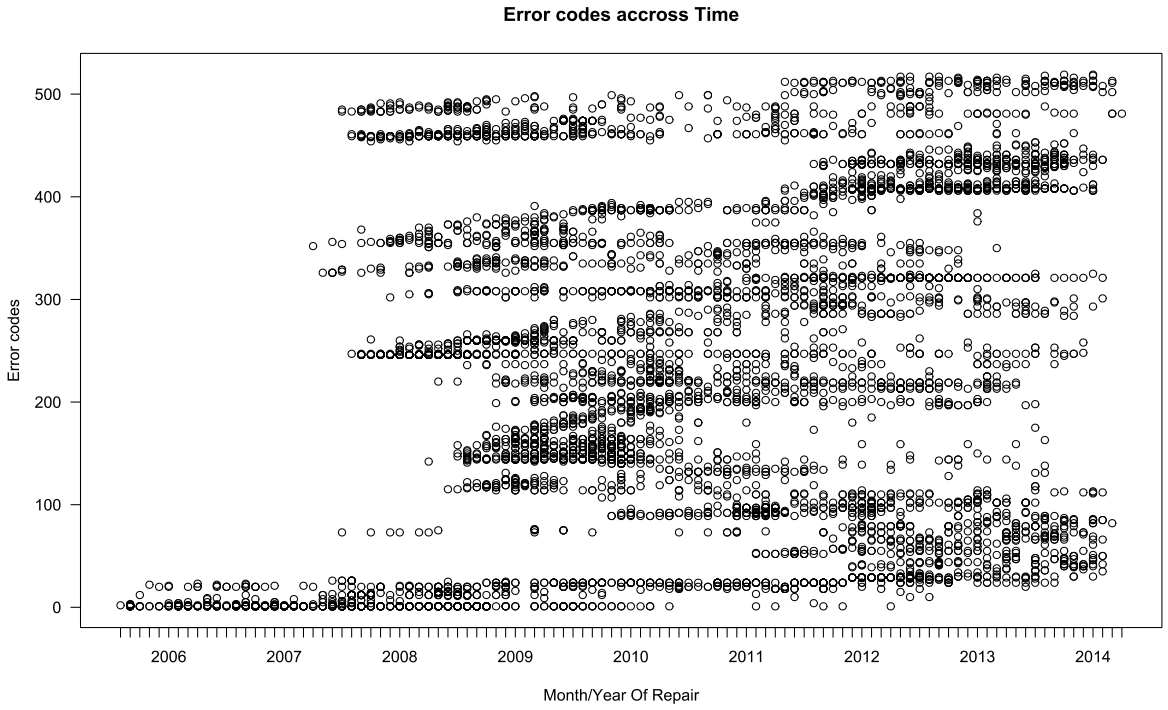


Figure 17 Error codes grouped by their month of repair for the filtered Supplier dataset

Despite of the big amount of observations, a clear trend is present. As we move over the error codes ordered in alphabetical order (as they are in the plot), the more likely it is that the error code was identified at a later month. Following this logic, error codes with an earlier alphabetical position are more likely to be assigned earlier in time. The exceptions to this trend are the first 50 error codes or so, since they seem to be present across all the time period, albeit less frequently as time goes by. In a similar vein, observations around years 2009 and 2011 can be particularly difficult to categorise, given the fact that almost all error codes have some elements present at that moment in time.

To gain more clarity on this pattern, Figure 18 shows the same data (repair date months) as box plots grouped by part type. Here we see the variability we expected among part types if we look at the distribution of IQRs over the total time span. The length of IQRs varies as well, something that indicates that the discrimination problem we expected to see around years 2009 and 2011 may not be as tough as originally thought, as observations from each part type tend to concentrate at slightly different moments in time within this particular period.

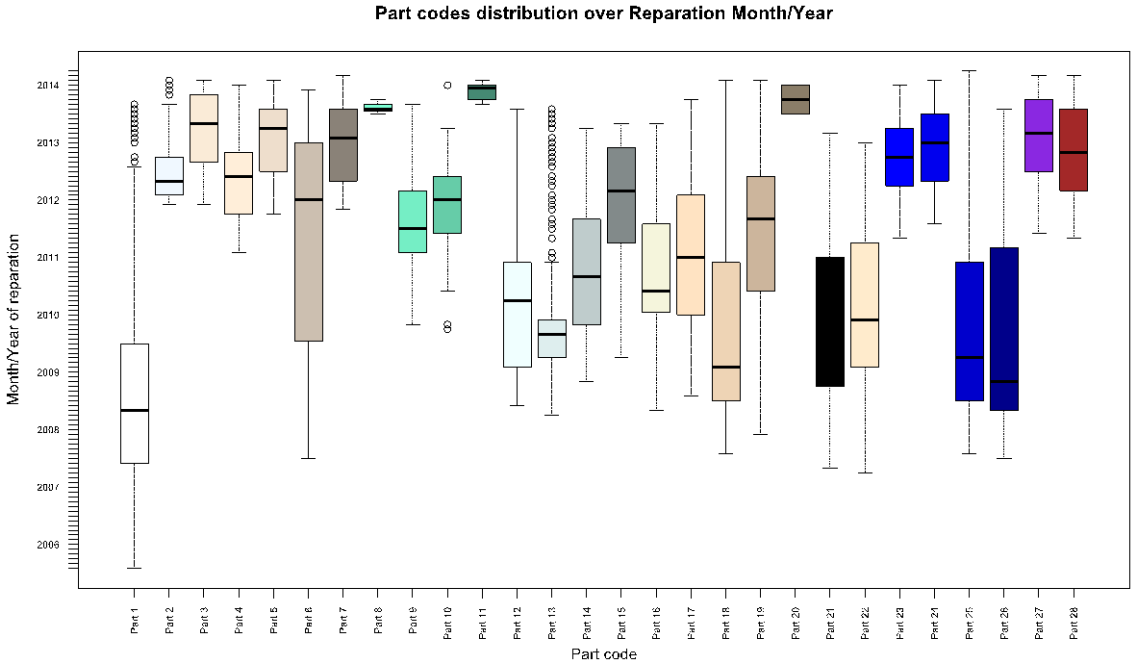


Figure 18 Months of the repair date grouped by part type for the filtered Supplier dataset

A similar situation occurs if we examine the distribution of observations according to the admission-to-drive dates they have registered. This is shown in Figure 19. The IQRs for each part type are just as spread over the whole period of study as in the case of repair dates. In some case the IQRs are also shorter, favoring the concentration of observations around a particular point in time, and as a consequence avoiding overlaps with other part types, thus helping classification. Examples of this are parts 8 and 11, 4 and 5 or 14 and 15.

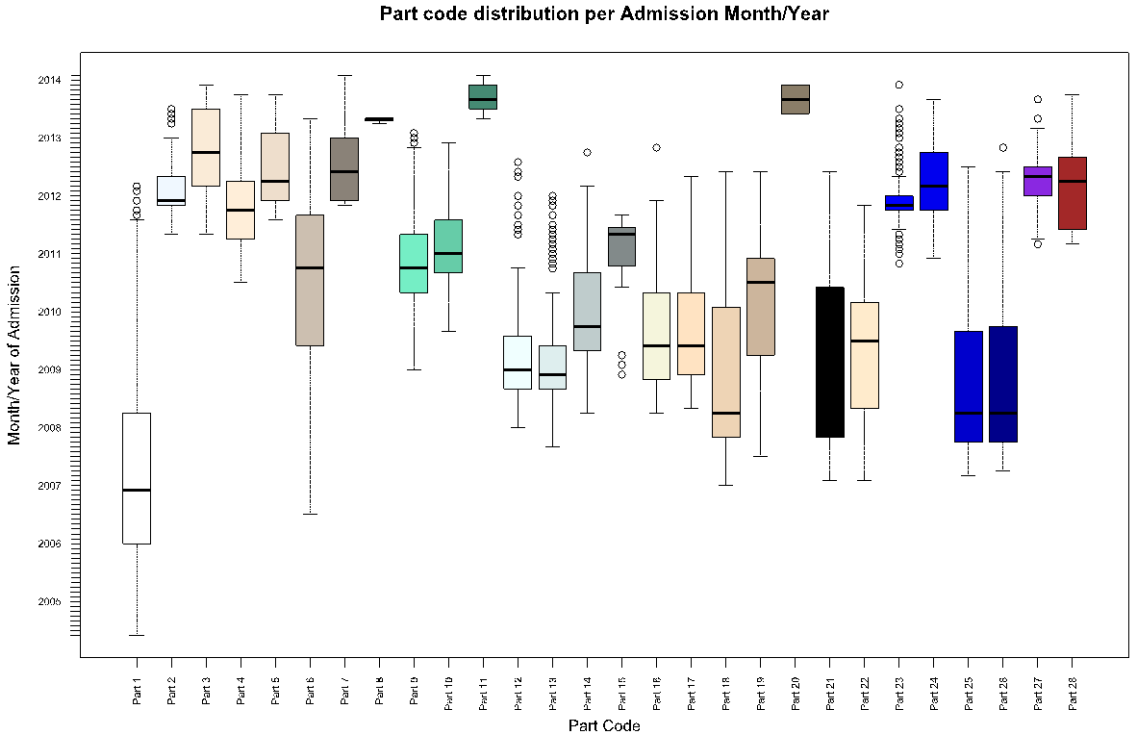


Figure 19 Months of the admission-to-drive date grouped by part type for the filtered Supplier dataset

All in all, these features can prove useful to increase the performance of the classification algorithm. Even though this is due to the particular way observations spread over time for our specific scenario, it highlights the importance that temporal data can have to improve the classification of unstructured text data.

A more revealing feature comes from calculating the amount of days from the day a car is first admitted to circulate until the day it is taken to repair, which we will call "driving time". Figure 20 shows the quartile distributions of this calculation arranged by part type. Contrary to what happens with mileage, here observations are very similar to one another within their part type (shown in the more or less compact IQRs) and different enough in comparison to observations from other parts (median values vary more or less evenly over a range of 500 days).

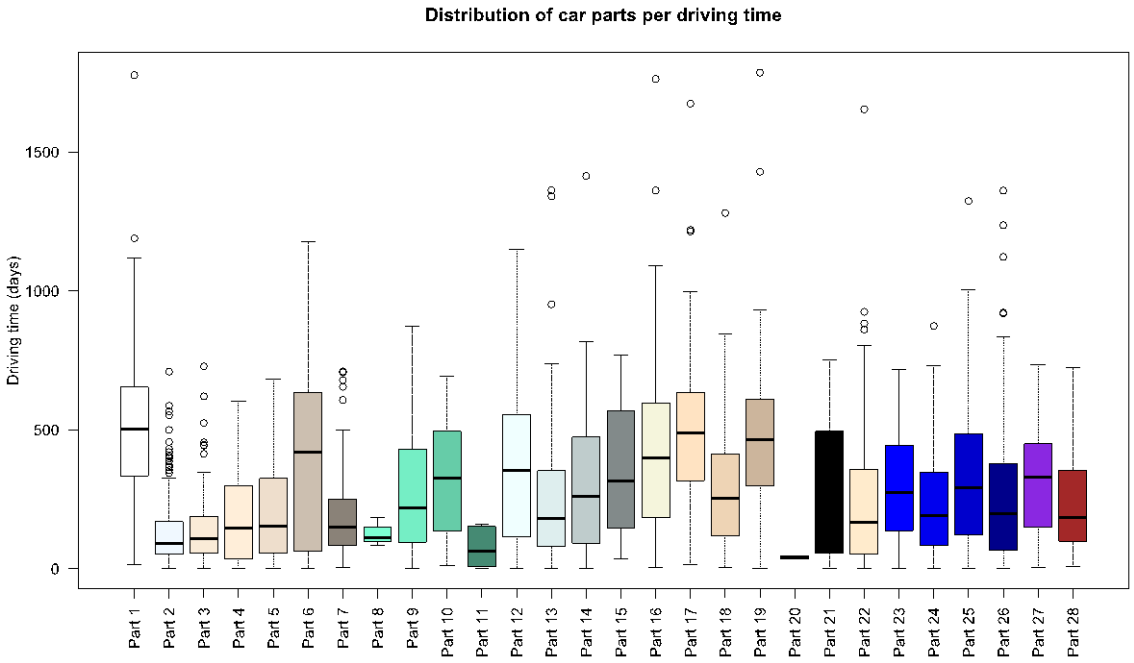


Figure 20 Box plots of driving times per part type for the filtered Supplier dataset

Based on these observations, and even though this feature does not promise to add much as much discrimination as the previous one, we can also consider the driving time as an additional feature. It is safe to include this feature along with the repair date because, despite of being a derived calculation between the admission-to-drive and repair dates, it has low correlation with their effects (-20% and 12% respectively).

4.2.1.2 Text reports in the dataset

In order to explore the nature of the texts in the dataset and the effects of different pre-processing sequences, we represent each report as a feature vector made of single-word terms. Based on this representation we perform two types of pre-processing. One that is blind to the language of the texts and another one that applies additional processing steps based on the identified language in each report (only English and German are valid options). Both pre-processing sequences are implemented in R, in a setup described in section 5.2.

The simpler *language-blind* approach takes every text report (document) in the dataset (collection or corpus) and 1) turns it into lowercase letters, 2) filters English and German stop words, 3) removes all numbers, and 4) removes all punctuation signs.

The *language-oriented* pre-processing approach takes every text report (document) in the dataset (collection or corpus) and 1) turns it into lowercase letters, 2) identifies the document's language, 3) filters only the stop words of the identified language, 4) removes all numbers, 5) removes all punctuation signs, and 6) stems the remaining terms in the document according to the identified language. The reason

to apply lowercasing before anything else is to improve the language detection rate, since the corresponding component is based on character n-gram frequencies where only a subset *s* is used (known as the Cavnar and Trenkle approach (Hornik et al. 2013)), probably leaving out many n-grams with uppercase letters.

Applying the simpler pre-process we obtain a document term matrix (DTM) with 8219 terms coming from all 6293 documents. Documents have a median sequence length of 33 terms, with a maximum of 95 and a minimum of 8 terms. As expected, we have high dimensionality issues despite of filtering stop words.

Meanwhile, the *language-oriented* pre-processing yields two separate document terms matrices which combined provide 9307 terms out of 5198 documents, more than in the *language-blind* pre-processing scenario, both in absolute terms and per document. This occurs even though terms are stemmed and the language detection discarded some documents that due to their spelling errors and high-content of abbreviations could not be identified either as English or German. As a consequence, we still have high-dimensionality. 3794 English-identified documents contribute 5365 terms, while the German matrix has only 1404 documents from which we obtain 3942 terms. We see then more English documents than German ones and consequently, more English terms than German terms.

In terms of length, English documents have a median sequence length of 27 terms, with a maximum of 90 terms and a minimum of 2. German documents have a median of 22 terms with a maximum length of 69 terms and a minimum of 5. These numbers show that at most, reports are approximately as long as this paragraph, with English documents being longer. This difference can be attributed to the fact that English typically needs more words to express what German can with just one compound word. An example could be the German (nevertheless grammatically incorrect) term "diebstahlschutzaktivierungsfehler" which in English would be written as "Anti-theft alarm activation error". It is also worth noting that stop word removal is significantly more effective in German documents.

Figure 21 shows the thirty most frequent terms from each DTM. We see that based on the language detection and stemming, results vary significantly. While in the *language-blind* case the rank is topped by internal terms and abbreviations with unclear meaning, in the language-focused scenarios we find stems that do belong to each language and are easier to interpret, even if their meaning appears to be vague in the context of a quality process. They revolve around the terms or stems "problem", "customer", "complain", "failure" and "defect".

On top of this lack of specificity, we also find big levels of correlation between the top terms, as shown by the ample interconnectedness among them (Figure 22, Figure 23, and Figure 24). This suggests two things. First, top terms are not particularly useful to discriminate among different categories of error codes since they tend to appear together over more documents than there are for each error code. This can be estimated considering that the highest average of observations per error code shown in Figure 14 is far smaller than, let's say, the 1527 observations where the terms customer and complaint correlate.

Secondly, stemming seems to mitigate some of this high correlation. Terms in the *language-oriented* cases do not show as much correlations as in the *language-blind* case. In fact some show no correlation at all (at least with these terms). This can be perhaps attributed to the fact that abbreviations are not as present as in the *language-blind* case. These also suggest that correlation in general is less present in the *language-oriented* version of the dataset, making it more useful for the classification task.

If we compare the terms extracted for this role to the ones found for the Mechanic role, we see there is little overlap. In the *language-blind* pre-processing datasets we find 25.12% of the Supplier terms also present in the Mechanic dataset. On the *language-oriented* side, despite stemming, the Supplier terms in English also appear 18.54% of the times among the Mechanic English terms, whereas in the German case, this occurs 20.16% of the times. This confirms the notion that in general each role refers to the same observations in very different terms.

Rank	Term	Occurrences
1	<internal term 1>	9374
2	<internal term 2>	6809
3	bedienteil	4446
4	high	4072
5	<internal term 3>	4054
6	mid	3123
7	usa	2980
8	fehler	2441
9	control	2297
10	navi	2206
11	customer	2026
12	steering	1914
13	noise	1858
14	entry	1774
15	issue	1610
16	complaint	1527
17	gerät	1359
18	yoke	1326
19	failure	1310
20	dvdwechsler	1265
21	see	1145
22	test	1125
23	confirmed	1084
24	power	1067
25	dvdchanger	1063
26	found	1060
27	lhd	1054
28	like	1051
29	dvd	1034
30	confirm	1011

Rank	Term	Occurrences
1	custom	1835
2	confirm	1783
3	nois	1586
4	test	1486
5	issu	1405
6	complaint	1360
7	failur	1144
8	yoke	1042
9	like	1000
10	found	936
11	part	891
12	close	889
13	check	884
14	due	826
15	caus	811
16	problem	810
17	rack	713
18	greas	687
19	see	678
20	play	669
21	bench	657
22	judg	656
23	bar	637
24	unit	611
25	acoust	568
26	clearanc	546
27	slave	519
28	defect	518
29	report	518
30	without	501

Rank	Term	Occurrences
1	fehl	2041
2	gerat	924
3	sieh	507
4	bestatigt	455
5	gepruft	405
6	ried	389
7	allgemein	383
8	gca	378
9	qec	367
10	bekannt	366
11	analys	325
12	funktion	307
13	stna	291
14	wurd	247
15	bitt	246
16	befund	241
17	<internal term 2>	216
18	mid	213
19	defekt	205
20	spindl	199
21	festgestellt	195
22	festgestelltna	185
23	laufwerk	180
24	verlang	171
25	beim	166
26	cds	163
27	dvd	144
28	motor	143
29	ergebnis	141
30	japan	136

Figure 21 Top 30 most frequent terms for Language Blind (left), English (centre) and German (right) Pre-Processing for the Supplier dataset

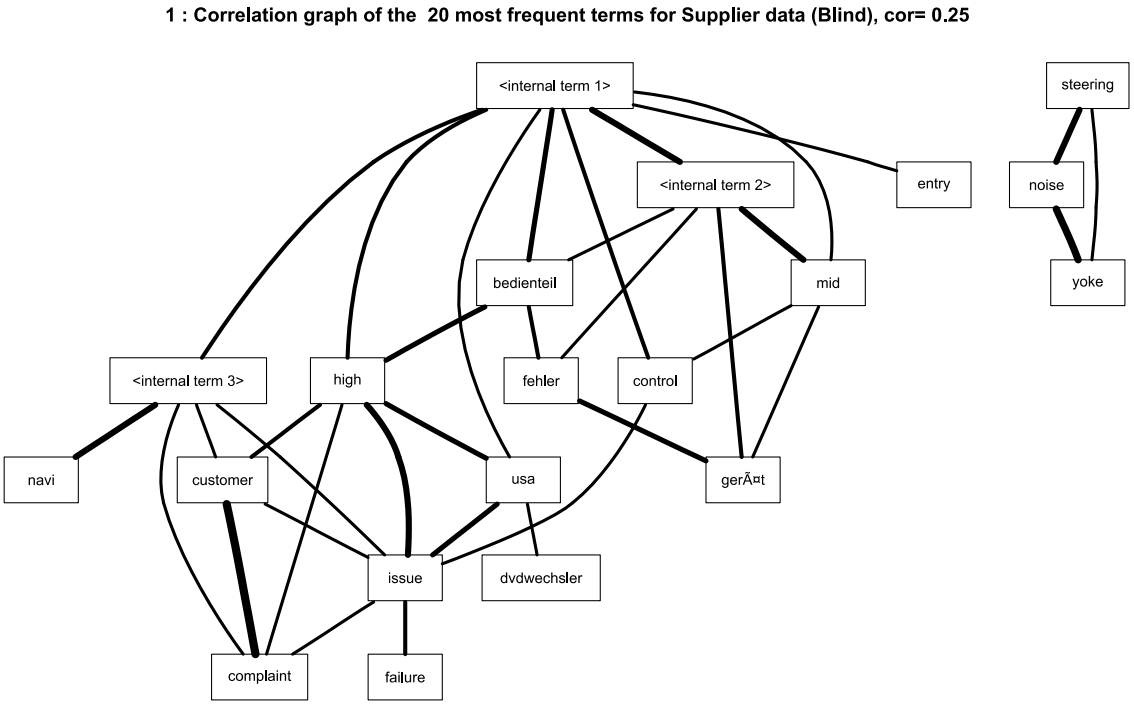


Figure 22 Correlation relationships among the 20 most frequent terms (Supplier dataset, language-blind pre-processing)

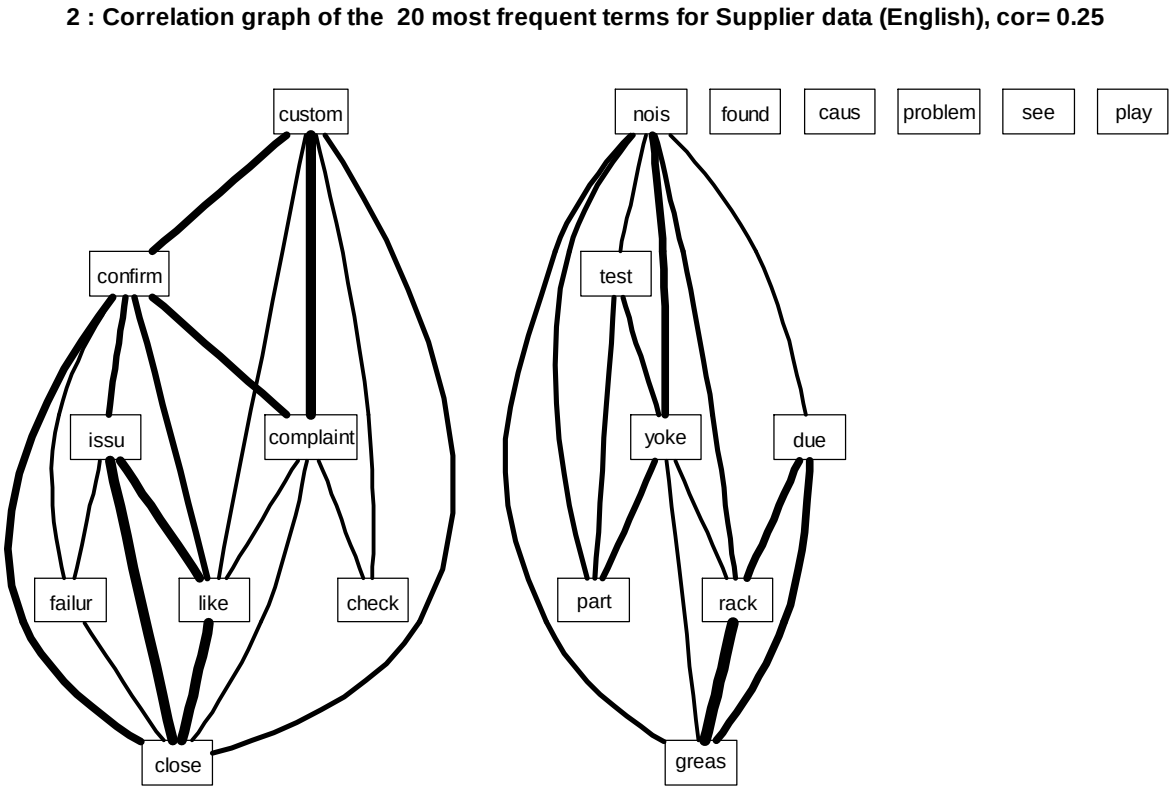


Figure 23 Correlation relationships among the 20 most frequent terms (Supplier dataset, English pre-processing)

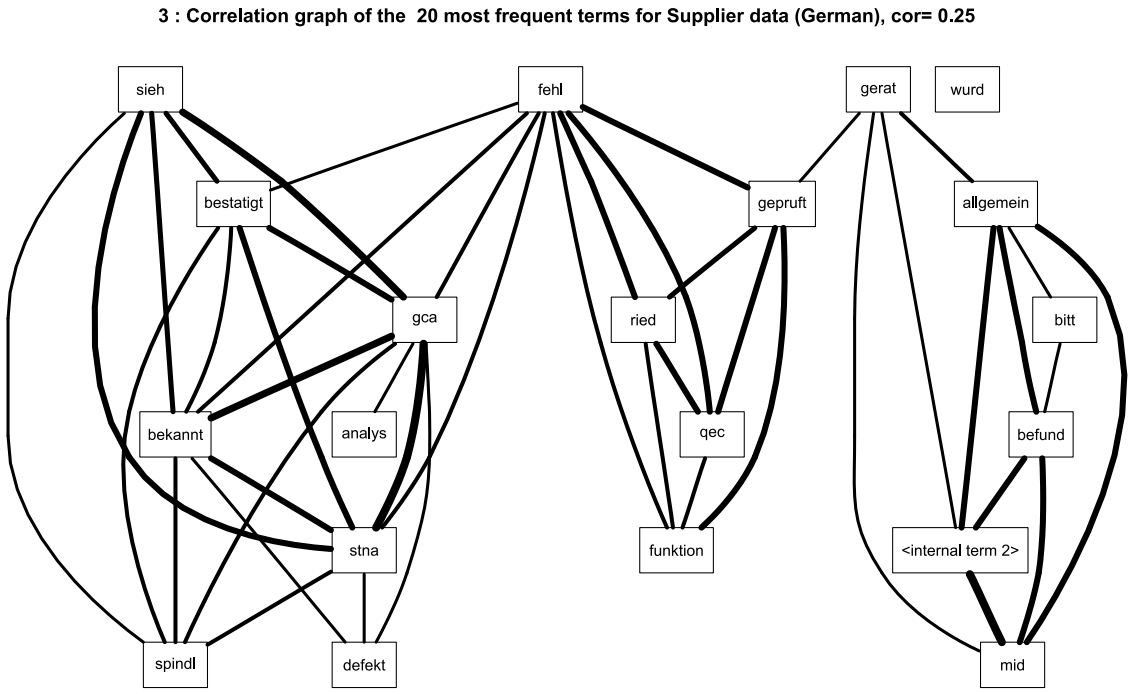


Figure 24 Correlation relationships among the 20 most frequent terms (Supplier dataset, German pre-processing)

To better understand each dataset, we test for the existence of a power-law distribution using a term frequency weighting scheme.

Figure 25 shows both a power-law estimation based on the fitted linear model (black line) and the Maximum-Likelihood Estimation method (see sub section 3.2.6) (orange line). We see as we approach the tail of the distribution that the fitted linear model estimation diverges from the MLE estimation as a result of the noise in the area. This is due to the fact that the estimation is based on the least-square error calculation of a linear regression method, which according to (Clauset et al. 2009) is inaccurate. The MLE estimation does not suffer with this and thus let us determine whether the data follows the power-law distribution in a more reliable way. Using KS tests to probe this assumption, we obtain a p-value of 0.003613, which cannot reject the possibility of data being drawn from other (exponential) distributions. On the other hand, the statistic of 0.035542 suggests a good fit between the fitted and actual distributions.

There are multiple ways to interpret these results, and looking at the estimations made for the data coming from a *language-oriented* pre-process helps to shed some light on the matter.

Testing the estimations made with MLE method for both the English and German sub sets (orange lines in Figure 26 and Figure 27), we find notable differences between languages. The English results (p-value of 0.046017 and statistic of 0.052702) contrast with the German ones (p-value of 0.518804 and statistic of 0.012992). They indicate that the frequency of terms identified as English terms could follow a distribution other than the power-law, while the so identified German terms are more likely to follow the power-law. In both cases however, the fit of an estimated power-law distribution is good. As we show later, because there are more English terms than German terms in the original dataset, it is expected that the behaviour of the *language-blind* pre-processed dataset resembles more the English subset than the German one. A simple inspection to the curvature at the beginning of the plots in all three cases easily confirms it.

Going deeper into the details of the Zipf plots, the curvatures just mentioned represent lower-than-expected frequencies of the most frequent terms, while the noise in the tails can be attributed to sudden changes in frequency values as we go down the rank. These deviations are in accordance to the claims of (Newman 2005) that power-law behaviour is not observed over the whole range of values. The reasons for this are beyond the scope of this thesis, but despite of these deviations, the values of the KS statistics suggest we can assume power-law behaviour in terms of ranks, but not in frequencies.

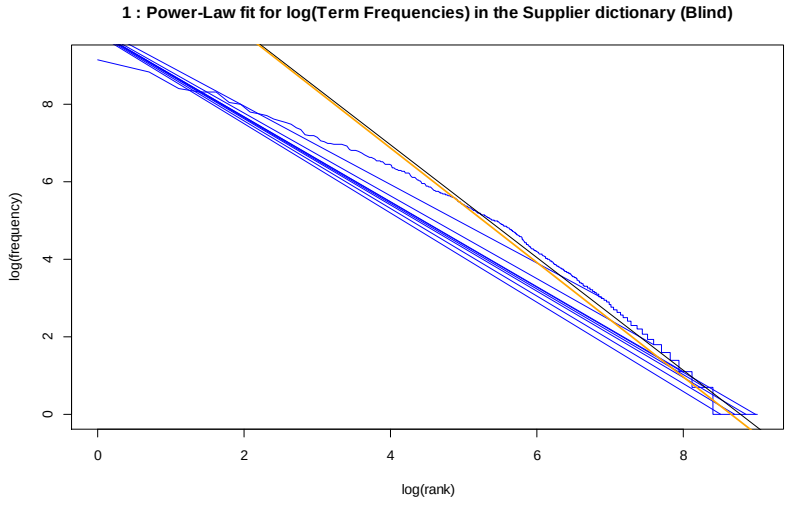


Figure 25 Zipf plot for the term frequency of features in the Supplier corpus (Language Blind)

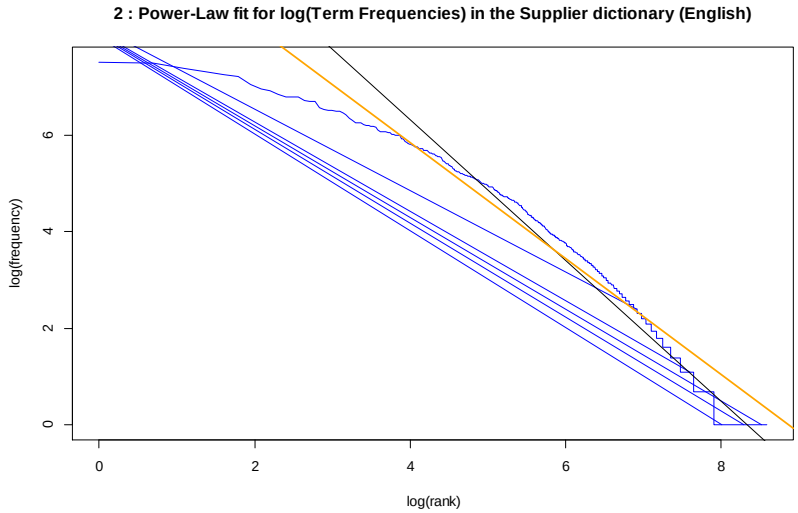


Figure 26 Zipf plot for the term frequency of features in the Supplier corpus (English)

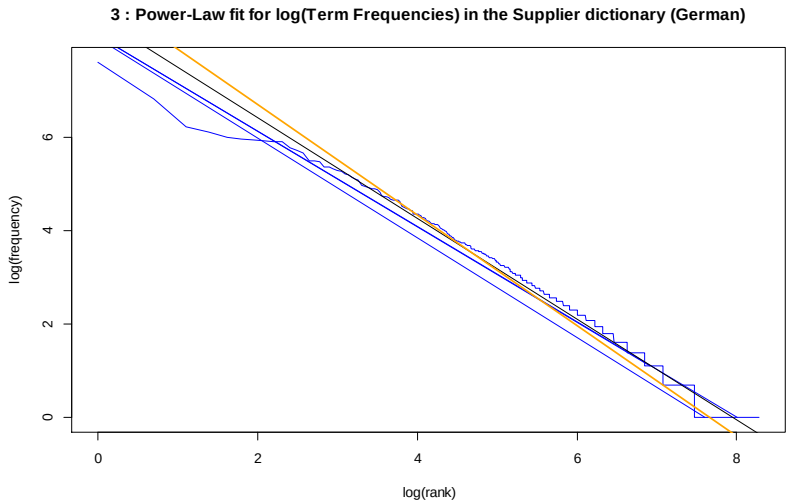


Figure 27 Zipf plot for the term frequency of features in the Supplier corpus (German)

4.2.2 Mechanic Role Dataset

We process this dataset in a similar way as we did with that of the Supplier role, so as to remove the observations with invalid dates or those with an error code that only appears once. From the initial 5624 observations corresponding to 1068 error codes, we remove 178 with invalid dates and 610 that have error codes appearing just once. This leaves us with a filtered dataset of 4836 observations that correspond to 428 error codes. Once again, we see that 393 out of these 428 error codes have less than 30 observations, thus preventing us from removing them.

In general, these numbers already show a significant decrement in the amount of categories that can be classified by only using the mechanics data. However, it is interesting to see that despite of this, the behaviours observed in the supplier dataset remain present here. Figure 28 shows the 150 most frequent error codes for this dataset, which now represents a bit more than a third of all error codes considered. We see again the potential behaviour of a power-law distribution.

The KS tests show again that for a scaling parameter a of 1.8338 and a minimum frequency x of 2, both the distance between fitted and actual distribution is small (KS statistic of 0.0344) and the hypothesis of data being drawn from another distribution is rejected (p-value of 0.6907). This let us conclude again

that despite of being considerably smaller than the supplier dataset, the mechanic data also follows a power-law distribution.

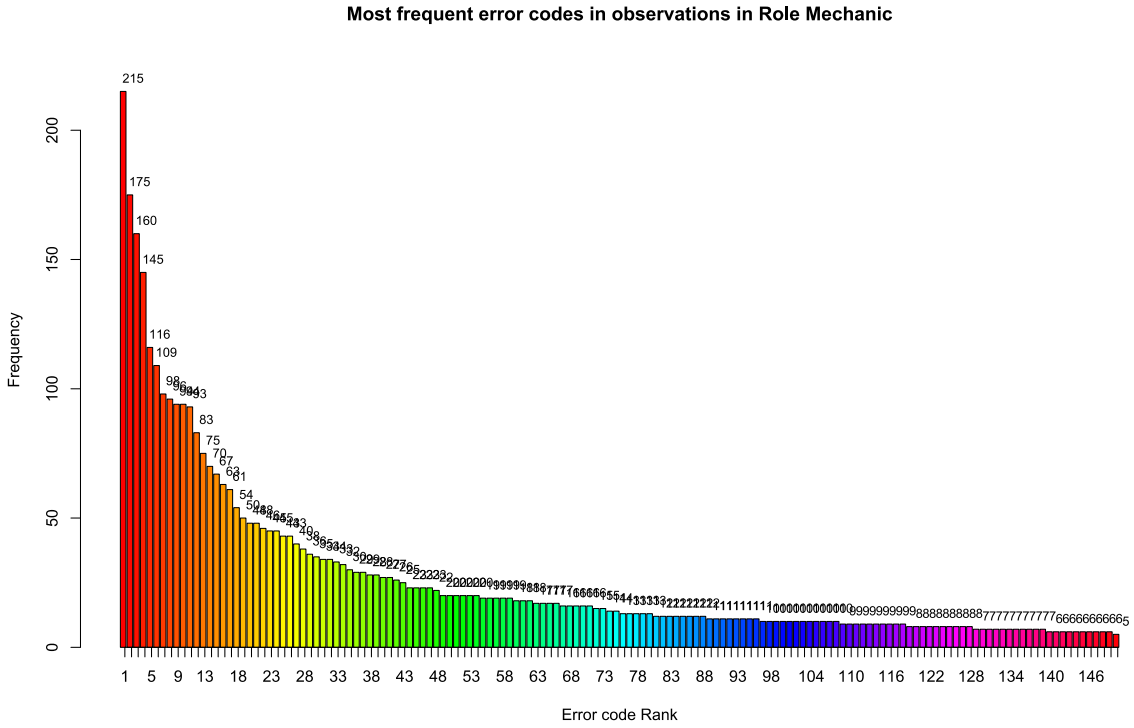


Figure 28 Plot of the 150 most frequent error code for the Mechanic Role (filtered dataset)

When comparing the distribution of error codes among part types before and after filtering the dataset (Figure 29 and Figure 30), we see that this time 4 part types are no longer represented in the filtered dataset. Aside from this change, the variance among part types remains. Again despite of removing 14% of the observations corresponding to 59% of all originally available error codes.

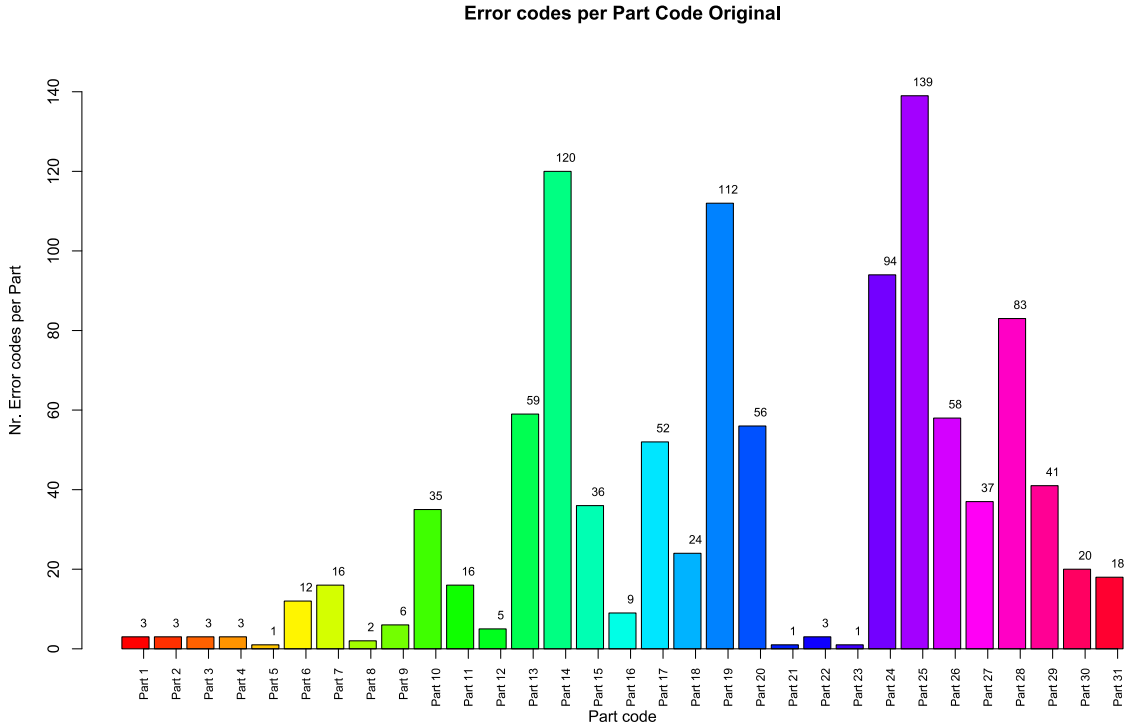


Figure 29 Distribution of error codes per part code in the original Mechanic dataset

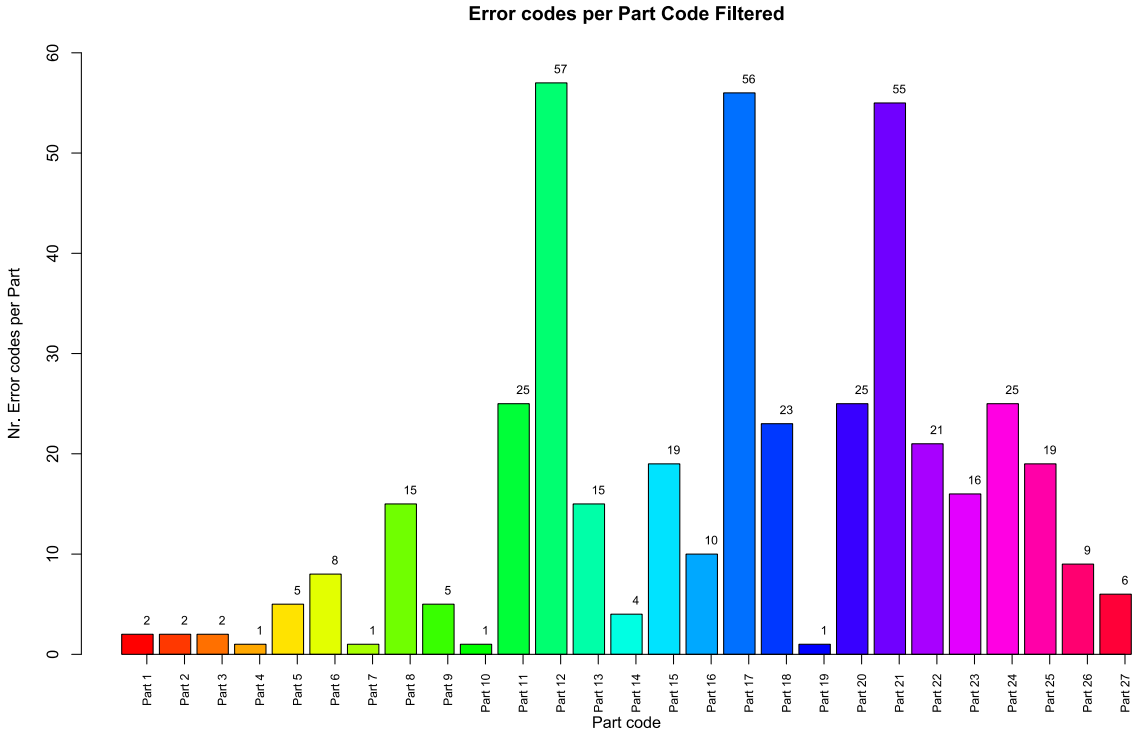


Figure 30 Distribution of error codes per part code in the filtered Mechanic dataset

When it comes to the amount of observations belonging to each part type and how many of those observations on average can be assigned to each error code, we find the same pattern as with the supplier dataset. Up to four part types have very few observations to produce good classification results, while the leftmost part type in the graph has plenty of data to train and test the classification algorithm. For the sake of brevity while still supporting the argument, we present only the average observations per error code for every part type in Figure 31.

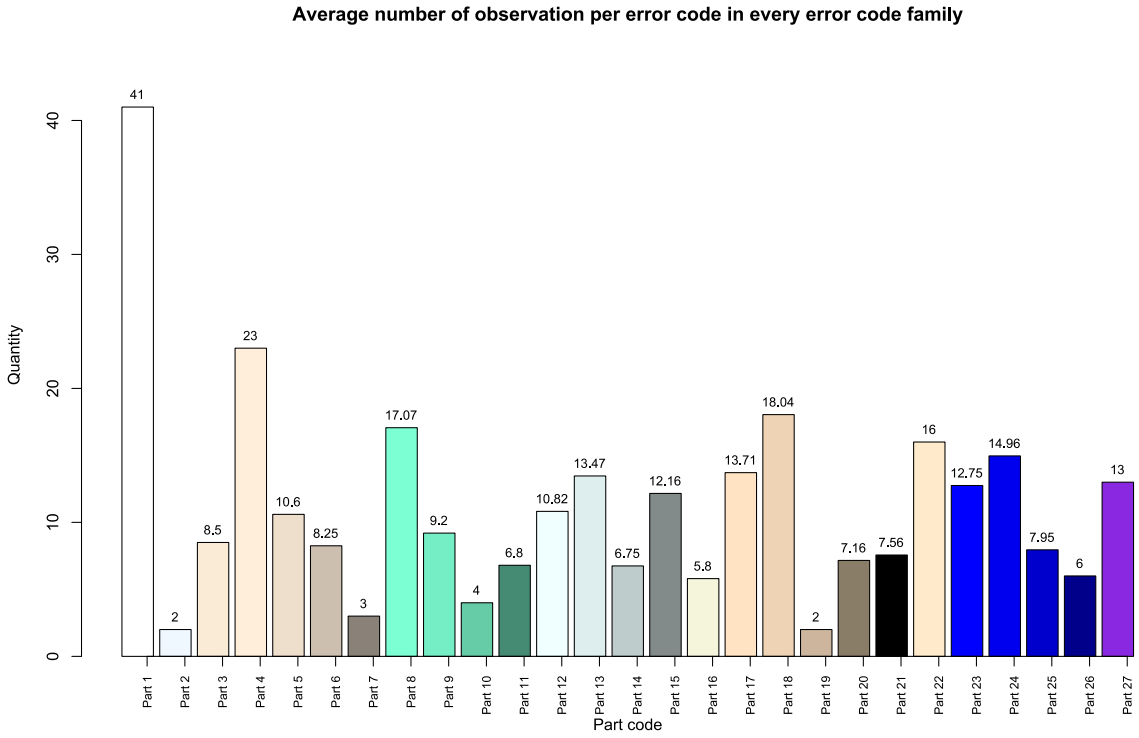


Figure 31 Average number of observations for every error code of each part type for the filtered Mechanic dataset

4.2.2.1 Structured data in the dataset

Since many of the characteristics of the Supplier dataset are also present in this dataset, in this sub section we provide only the key graphics to support the argument and briefly mention their relevance for the classification algorithm. Additional material can be found in chapter 7: Appendices.

4.2.2.1.1 Regarding mileage

Figure 32 shows the median values arranged by part types according to the Mechanic dataset. For most of the part types (except for the last four part types) the continuous behavior shown in the Supplier dataset is also present here. This means that in most cases, the first half of the observations belonging to a part type have mileage values very similar to those of other part types, providing very little variability to easily discriminate among part types. As a consequence, mileage is not a suitable feature to add to the classification.

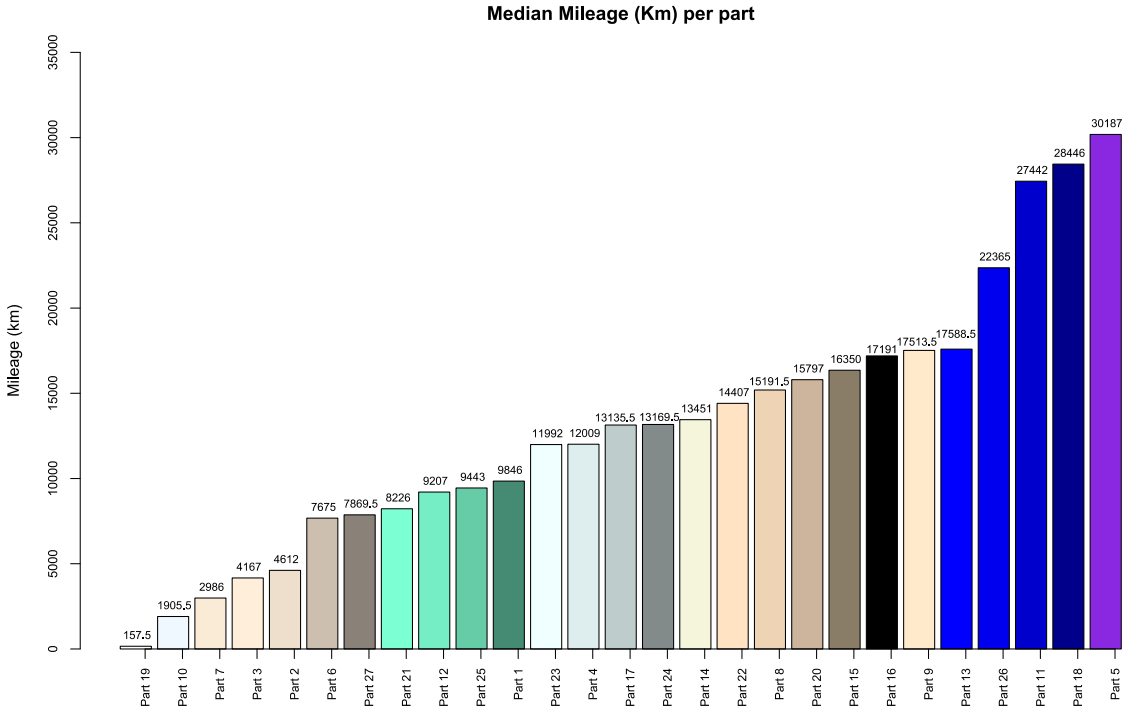


Figure 32 Ordered median mileage values per part type for the filtered Mechanic dataset

4.2.2.1.2 Regarding time

Looking at the way observations are distributed over time in the Mechanic dataset, we see a similar behaviour as the one present in the Supplier role. As the box plots in Figure 33 show, error codes are heterogeneously distributed across the total time period with IQRs for every part type having a compact length, few outliers (in most cases), and almost no alignment of their median values. This means that the first 50% of the observations of every part type are dated earlier than different points in time. This holds to the pattern seen previously and supports the idea of using the repair date as an additional feature for the classification.

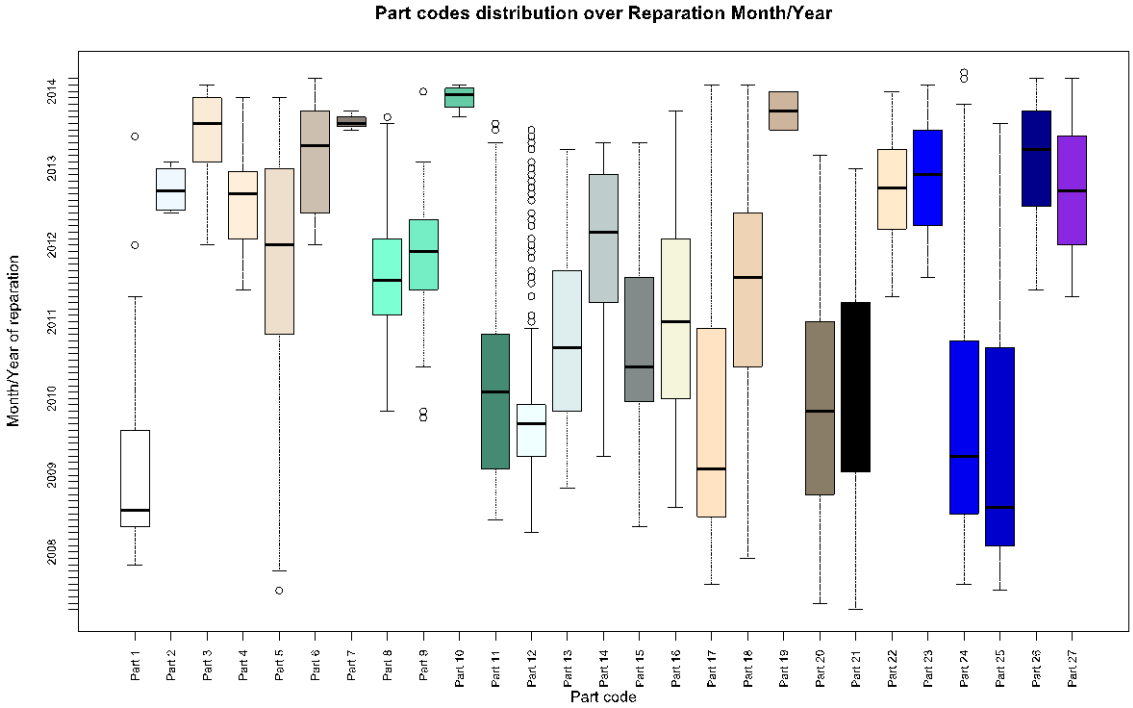


Figure 33 Months of the repair date grouped by part type for the filtered Mechanic dataset

In a similar vein, Figure 34 shows that the distribution of observations according to their admission-to-drive dates also mimics the distribution of repair dates in this dataset. This holds even to the point of having shorter IQRs that avoid overlap, as it can be seen with between parts 6 and 7, or 13 and 14. As a result, the admission-to-drive date also constitutes a good supporting feature in this dataset.

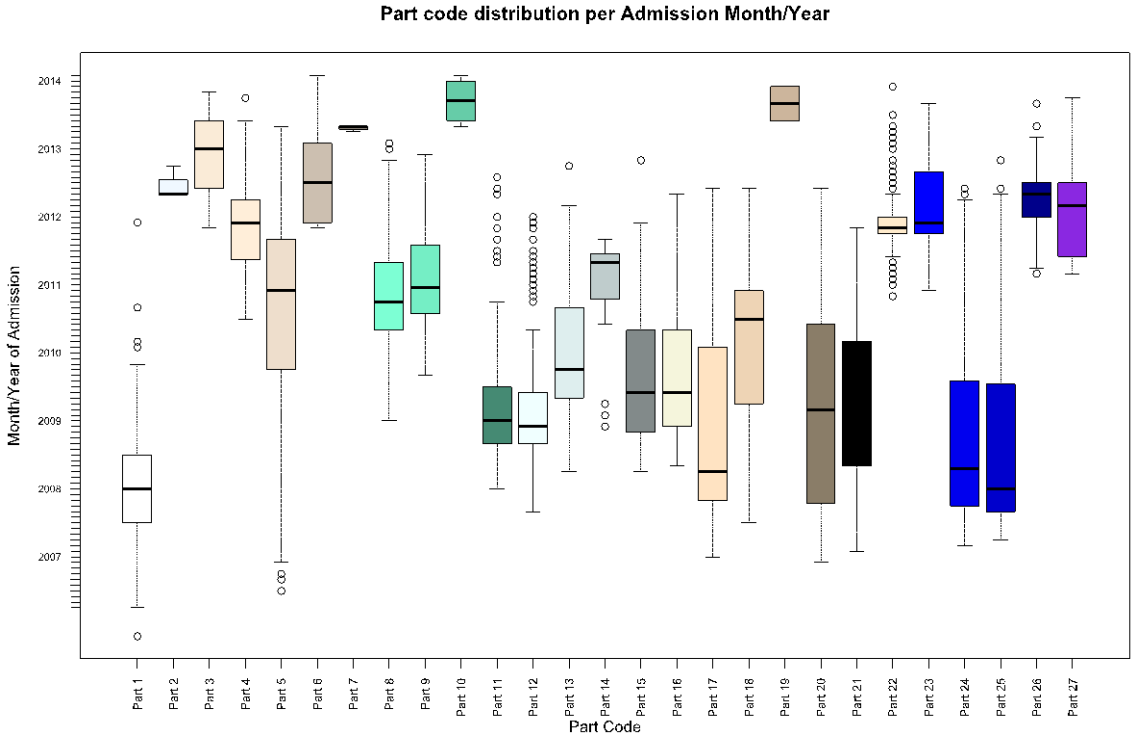


Figure 34 Months of the admission-to-drive date grouped by part type for the filtered Mechanic dataset

Concerning the derivative feature we presented in the Supplier dataset, driving time, Figure 35 shows again a similar situation for the Mechanic role. However, a closer examination and analysis suggest mixed results as a classification feature. On one side, the six part types on the leftmost side of the graph

show less overlapping of their IQRs, as well as more compact lengths. This is positive to increase variability among part types. However, the correlation values between driving time and their original features, admission-to-drive date and repair date, increase to 1.71% and 37.75%. While still far from a range of high correlation, this may bring slight overestimation of effects which may in turn decrease performance.

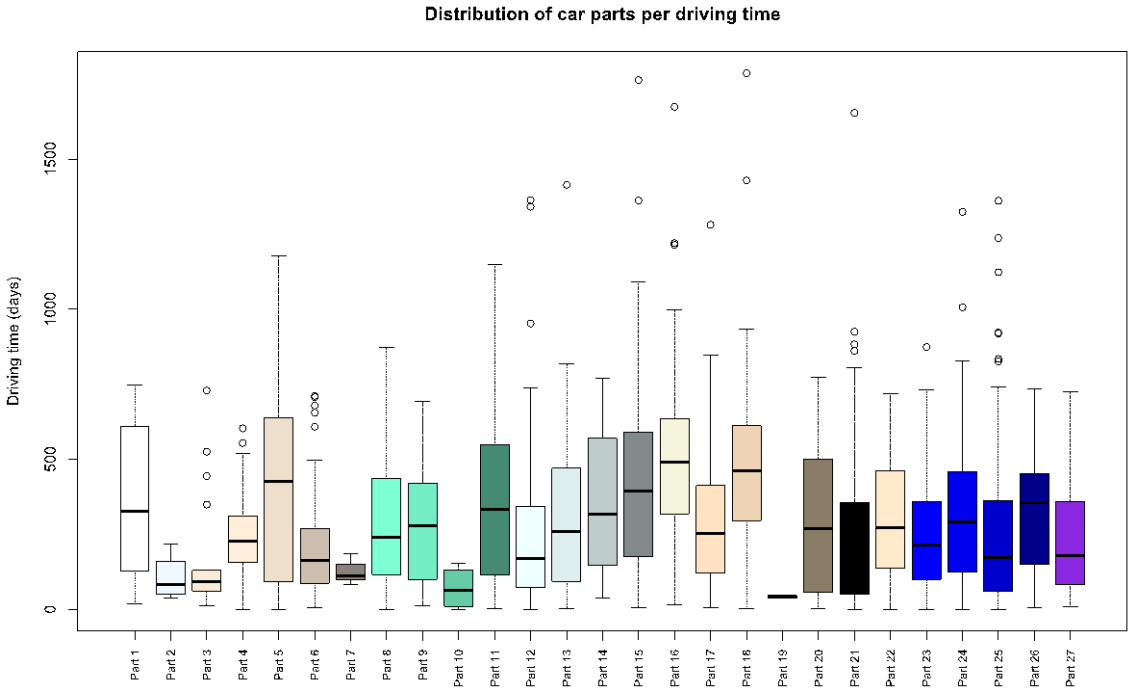


Figure 35 Box plots of driving times per part type for the filtered Mechanic dataset

4.2.2.2 Text reports in the dataset

As with the Supplier dataset, the Mechanic data was pre-processed with two different sequence of tasks. One that is *language-blind* to language and another who adds stemming based on the identified language (either English or German).

In the *language-blind* case, 8989 terms are found out of 4836 documents, while the English DTM contains 3556 terms coming from 1572 documents. Finally, the German DTM has 3023 terms from 2671 documents. This sums up to 6579 terms from 4243 documents from the *language-oriented* pre-processing, which means slightly less terms both overall and per document compared to *language-blind* pre-processing. Also, contrary to the case in the Supplier dataset, here we have more German documents than English ones and yet the number of English terms continues to be bigger.

Regarding document word counts, the *language-blind*-pre-processed documents have a median sequence length of 18 terms, with a maximum of 48 and a minimum of 3 terms. English documents have a median length of 23 terms, with a maximum of 39 terms and a minimum of 1. German documents have a median length of 7 terms, with a maximum of 33 terms and a minimum of 1. We see overall very short documents in this dataset, approximately half the size of their counterparts in the Supplier dataset (using the maximum lengths as reference). Moreover, the minimum value for the English hints to the misidentification of certain reports as English documents. This is confirmed when we see the smallest English document contains the stem “totalausfal”, German for “general failure”.

Figure 36 shows the top 30 terms as obtained from either of the pre-processing approaches. We see again the presence of abbreviations on the top of the *language-blind* results. Meanwhile in the English terms, we see the presence of two stems “command” and “comand”, which evidences the presence of spelling mistakes in the dataset. Something similar occurs in the German rank with stems “imm” and “immer”. Another notable difference is the presence of more terms related to parts in comparison to the ranks of the Supplier dataset. Here we see over all three ranks terms or stems related to radio, dvd, display or audio components.

Additionally, there is again little overlap between the terms or stems found in the Mechanic dataset and those found in the Supplier one. In the *language-blind* pre-processing approach we find a 22.97% overlap, while in the *language-oriented* results overlap ratios are of 27.98% for English and 26.29% for German. The slight increments are due to the fact that overall the Supplier *language-oriented* datasets have more terms than the Mechanic datasets, thus increasing the chances of every term to be found. This relation also explains the decreased overlap in the *language-blind* results.
When looking at the correlations among the top terms (with values of at 0.1) as depicted by lines in Figure 37, Figure 38, and Figure 39, we see similar patterns to those found in the Supplier dataset, even though overall the strength of the correlations (as shown by the thickness of the lines) is not a big. This is particularly noticeable with the German case, where four of the terms have no correlation at all. Still, the fact that top terms tend to appear together in documents across different error codes (as evidenced by the presence of big groups of interconnected terms) suggests that the most frequent terms are not suitable for classification.
In the end, all the differences between Mechanic and Supplier terms make it clear that each role has different perspectives about the same observations.

Rank	Term	Occurrences
1	<internal term 2>	5702
2	<internal term 1>	4601
3	bedienteil	4458
4	<internal term 3>	3775
5	high	2594
6	usa	2503
7	control	2265
8	unit	1229
9	entry	1224
10	mid	1195
11	comand	1136
12	navi	1135
13	radio	1074
14	dvdwechsler	980
15	states	922
16	command	897
17	test	821
18	gelesen	789
19	<internal term 4>	781
20	audio	696
21	<internal term 5>	652
22	<internal term 6>	626
23	dvd	614
24	cds	523
25	defekt	502
26	found	472
27	short	472
28	laufwerk	440
29	<internal term 7>	402
30	fault	372

Rank	Term	Occurrences
1	state	906
2	unit	896
3	test	880
4	command	871
5	comand	812
6	radio	757
7	code	545
8	custom	514
9	perform	491
10	found	468
11	will	467
12	short	466
13	fault	379
14	replac	366
15	client	335
16	inop	327
17	check	293
18	intern	280
19	screen	257
20	function	255
21	player	241
22	time	228
23	work	219
24	system	212
25	edac	210
26	verifi	205
27	audio	198
28	display	196
29	changer	193
30	eject	192

Rank	Term	Occurrences
1	geles	744
2	dvd	541
3	navi	455
4	defekt	405
5	cds	386
6	laufwerk	384
7	fallnr	348
8	radio	340
9	comand	298
10	fehl	282
11	gerat	282
12	audio	250
13	ztw	222
14	lasst	210
15	display	190
16	standig	188
17	funktion	185
18	moglich	177
19	mehr	157
20	zeitweis	155
21	fahrt	151
22	geht	141
23	ausgeworf	138
24	navigation	130
25	beim	122
26	erkannt	121
27	imm	119
28	reset	116
29	immer	103
30	intern	102

Figure 36 Top 30 most frequent terms for Language Blind (left), English (centre) and German (right) Pre-Processing for the Mechanic dataset

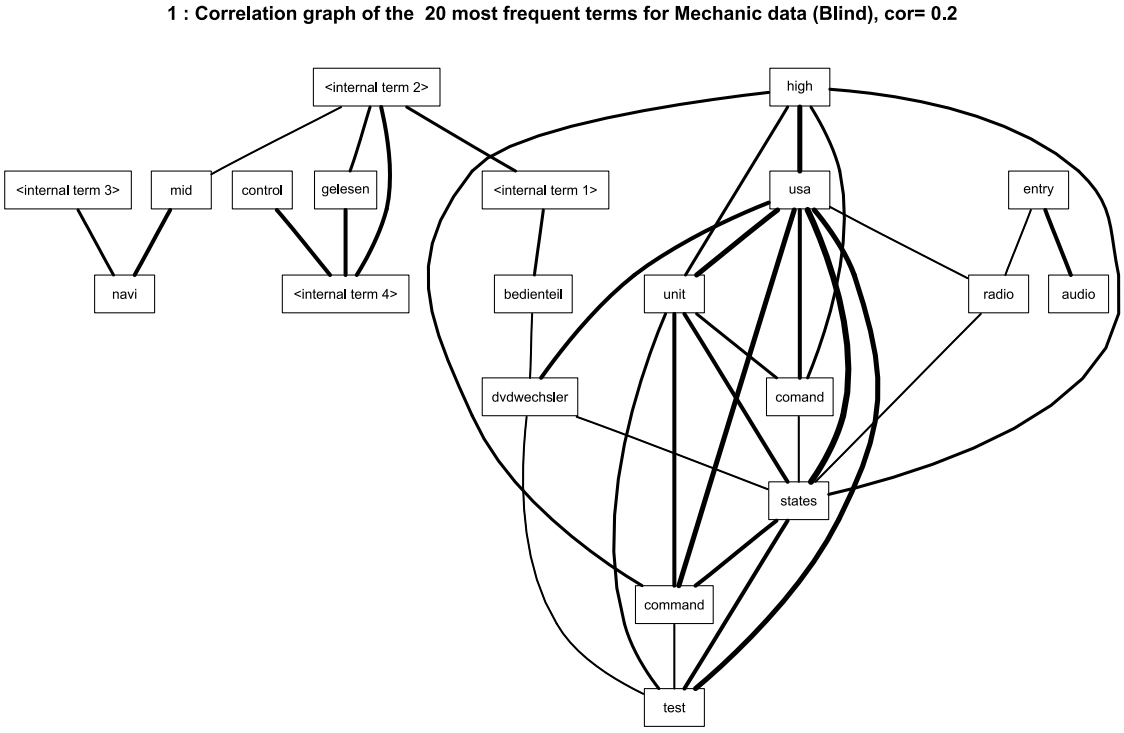


Figure 37 Correlation relationships among the 20 most frequent terms (Mechanic dataset, language blind pre-processing)

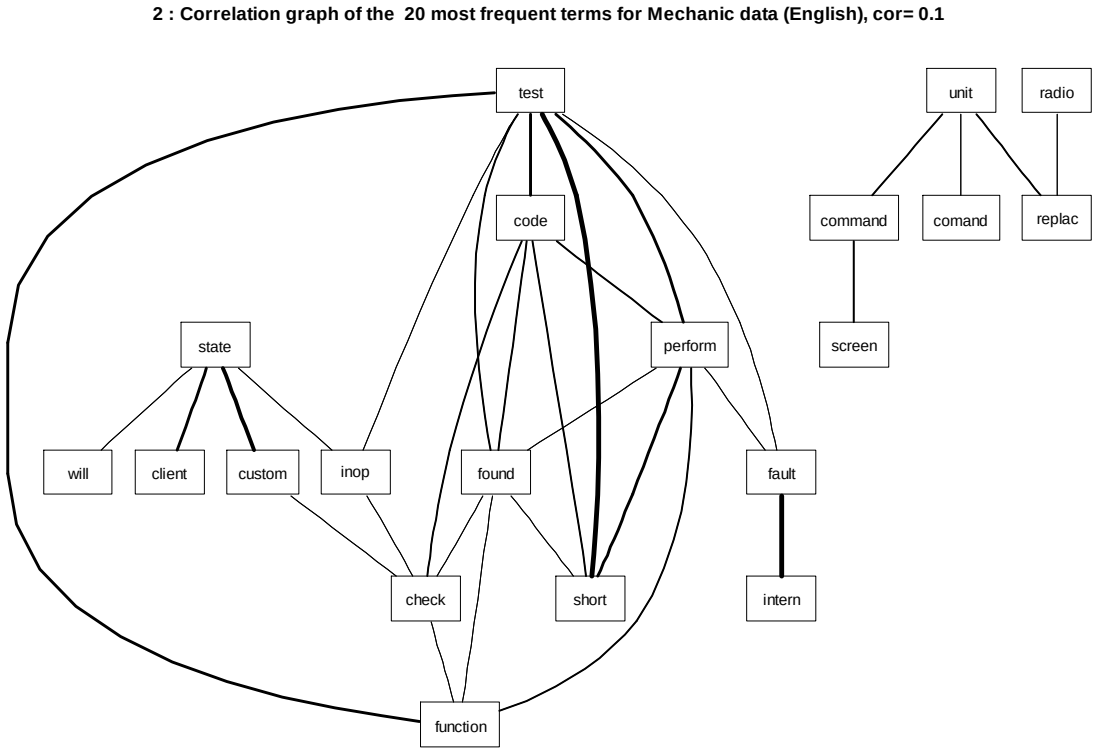


Figure 38 Correlation relationships among the 20 most frequent terms (Mechanic dataset, English pre-processing)

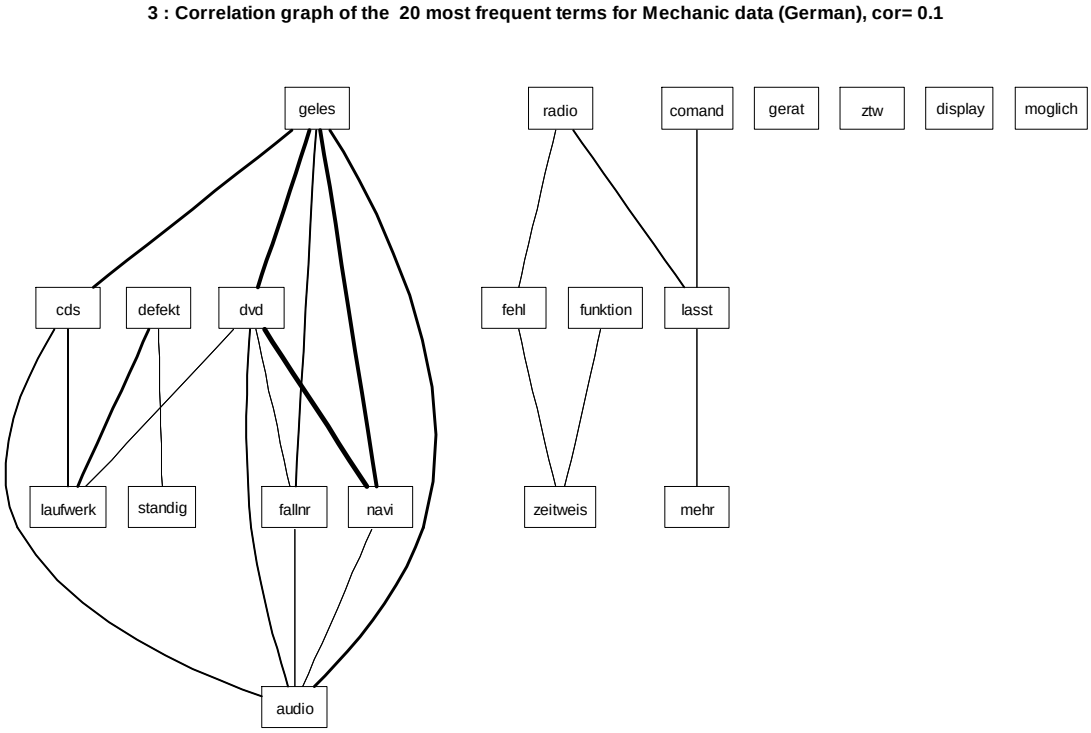


Figure 39 Correlation relationships among the 20 most frequent terms (Mechanic dataset, German pre-processing)

To explore the existence of power-law behaviour in the Mechanic data, Figure 40, Figure 41, and Figure 42 show the estimations using the fitted linear model in black and the MLE estimations in green. Opposite to the case of the Supplier data, differences between each line are not that significant since all distributions resemble more a straight line. At the same time this alone already indicates that data in all cases follows a power-law. KS tests confirm this intuition. In all cases the KS statistic shows very close resemblance between the data distribution and the estimated power-law distribution (0.010203 for the *language-blind* pre-processing, 0.018839 for English data, 0.019112 for German data). Similarly, the hypothesis tests do not support the fact that data could be drawn from another distribution by a good margin (p-value 0.898385 for the *language-blind* data, 0.777331 for the English data and 0.803988 for the German data)

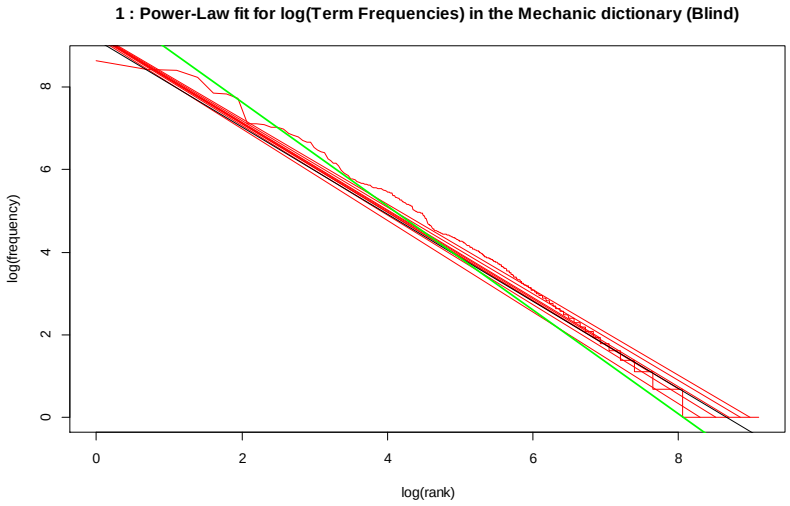


Figure 40 Zipf plot for the term frequency of features in the Mechanic corpus (Language Blind)

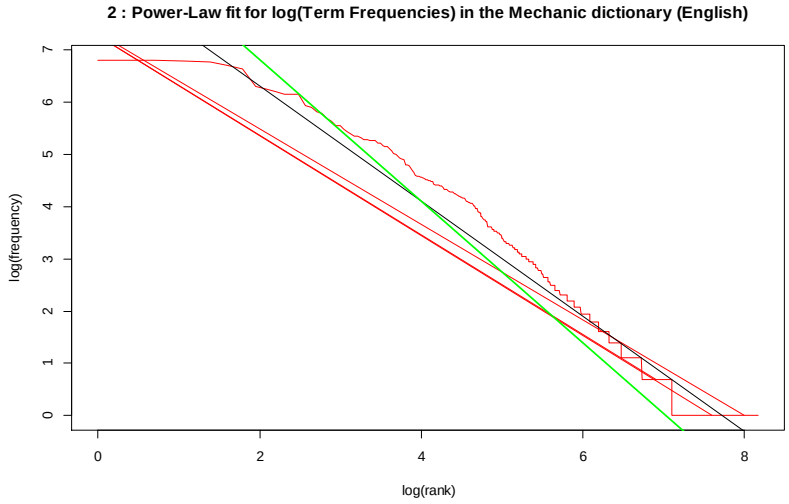


Figure 41 Zipf plot for the term frequency of features in the Mechanic corpus (English)

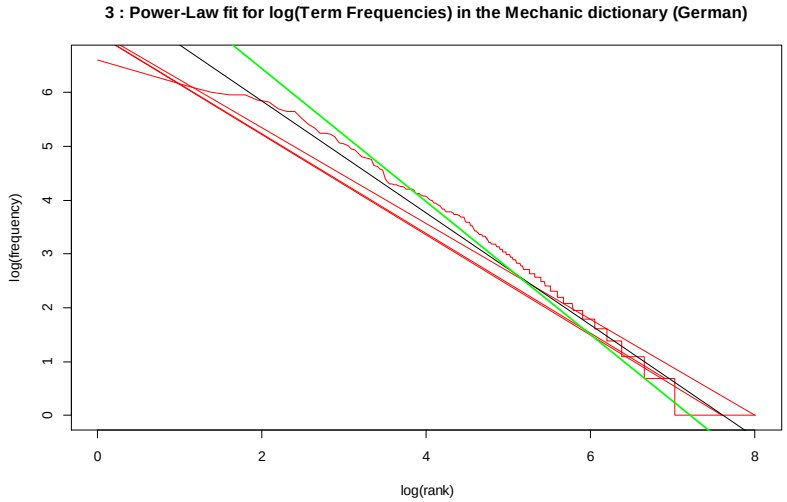


Figure 42 Zipf plot for the term frequency of features in the Mechanic corpus (German)

4.2.3 OEM Role Dataset

The dataset with preliminary reports from the OEM begins with 583 observations, a considerably smaller size in comparison with the datasets from the other two roles. After removing observation with invalid dates or error codes that only appear once, we retain 469 observations corresponding to only 40 error codes. As with the other datasets, we cannot remove error codes with few observations since 36 out of the final 40 error codes have less than 30 occurrences. Regardless of this common issue, the dataset of preliminary reports from the OEM as it is represents only 3.14% of all classification categories (error codes), which definitely makes it unsuitable to obtain meaningful results.

As expected, the visual exploration of the same features for the OEM role showed significant differences in the way observations are distributed. For the sake of brevity, we present a few representative plots in this sub section, while the rest of the material can be found in chapter 7: Appendices.

While overall the OEM dataset is not consistent with the other two, the frequency of its observations still follow a power-law distribution for a scaling parameter of 1.864 and a minimum frequency x of 2. The values of the KS tests (statistic of 0.0924, p-value of 0.8839) support this assertion. Despite of this, the KS statistic shows less fit to the power-law distribution than in the other two datasets, most likely because of the small number of observations.

Regarding the derived feature driving time, correlation values among it and its parent features are still acceptable with -14.61% of correlation with the admission-to-drive date and 30.03% of correlation with the repair date.

Beyond of the data that it actually depicts, Figure 43 summarises all the limitations that disqualify the OEM dataset as a useful one. The first major disadvantage is how little categories of the original set are covered. The 40 error codes contained in this dataset belong to only 9 part types and their 469 observations span over a six years period instead of over a decade as it happens in the other role datasets.

On top of that, the observations that are part of the dataset describe very different behaviours. The IQRs of the parts shown in Figure 43 show a very different distribution to those of the other datasets. Their lengths are very short, their medians tend to align on the same point in time, there are practically no outliers, and the part types that seem to be most prominent do not match the findings in the other roles. We know from the process description (see sub section 1.2.1) that preliminary reports are optional, and as such, the differences in distribution can be explained at least partially by this condition. Consequently, the differences in behaviour of this dataset can be either for real reasons or simply because of missing data. Without knowing the reason why experts from the OEM Company would create a preliminary report or not, we can only conclude that using this data to classify text reports can mean unrelated effects to the model, ultimately hurting the overall performance.

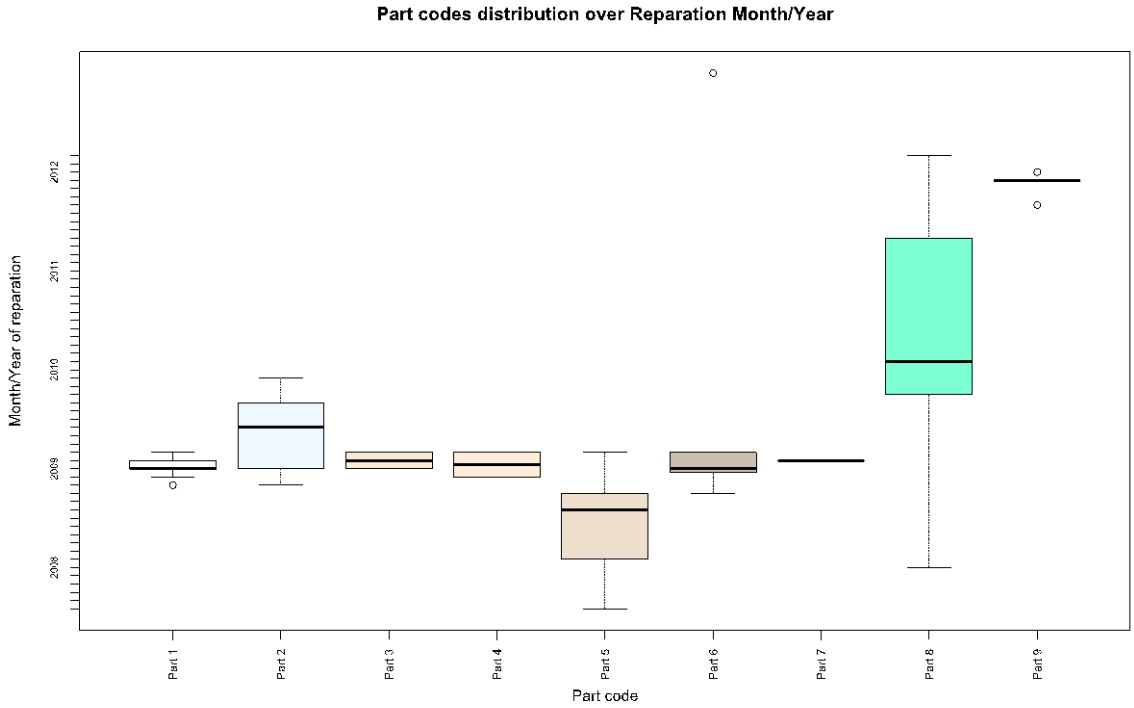


Figure 43 Months of the repair date grouped by part type for the filtered OEM dataset

4.2.4 Summary of the Two Main Roles

Throughout the examination of the three different datasets both on their unstructured text data and the structured fields, we find that the datasets from the Supplier and Mechanic worker seem to be more appropriate for classification purposes since they have a good coverage of the original dataset, have a similar distribution of observations across the part types and their observations span over the whole time frame consistently, enabling the algorithm to be trained with data from the whole period.

When it comes to the text collections from each dataset, these datasets do not look so similar anymore. As Figure 44 shows, Supplier reports are usually longer than their Mechanic counterparts. Despite of this, and the fact of having less reports, the Mechanic dataset has a slightly larger quantity of distinct terms to use in the classification. This can be seen as mechanics writing considerably shorter reports which nonetheless include a bigger variety of words. As a consequence, we can also expect more terms having low frequencies, something that seems to be confirmed with the Zipf plot in Figure 40, since the curvature at the most frequent terms (deviation from the fitted power-law distribution, caused by top

terms that "accumulate" more occurrences than they should according to their rank) is not as strong as in the case of the Supplier dataset (Figure 25).

Metric	Supplier blind	Mechanic blind
Vocabulary size	8219	8989
Number of documents	6293	4836
Maximum report length (tokens)	33	23
Median report length (tokens)	95	39

Figure 44 Comparison of the Supplier and Mechanic document collections (Language-blind pre-processing)

Language Statistics	Supplier	Mechanic
Shannon's entropy for language engineering	71,99%	71,88%
Relative vocabulary size	4,27%	9,91%
Vocabulary concentration	21,34%	32,27%
Vocabulary dispersion	82,15%	92,08%
Average content rate	98,49%	98,94%
Functional words	124	95

Figure 45 Comparison of language statistics for each role

Figure 45 shows an additional perspective about the differences and similarities between our two relevant datasets (in their *language-blind* variants). By looking at the different values of these selected language statistics (see sub section 3.2.5), we can confirm the patterns observed so far either visually or by quantitatively analysing the text corpora.

Starting with entropy, we notice that both datasets have similarly high values. This indicates that both datasets contain mostly words with small frequencies, instead of a few that are very frequent (Bank et al. 2012). This is to be expected given the identified power-law distribution we find in both datasets.

Looking at the relative vocabulary size, we find low values in both cases, albeit with a slight increment in the case of the mechanic dataset. These values indicate a simple language (Bank et al. 2012), probably due in this case to the very specific domain in which text documents were created. Extending a bit more the interpretation of this very domain-specific vocabulary, the slight increase in the case of the mechanic dataset could be attributed to the fact that this role tends to refer more to technical parts than the supplier, as shown in the analysis of the top terms (see sub section 4.2.2.2). However, it is important to notice that because of the stop word removal applied during the pre-processing step, these values may be under estimated (see formula in sub section 3.2.5). Yet, considering the length of the documents and the high average content rate, it is likely these values are not that distant from those of the unprocessed texts. All in all, this supports the document representation as a *bag of words*, where terms are considered by themselves the main source of information to classify documents.

When it comes to vocabulary concentration, high values would indicate that the vocabulary consists of a few words (Bank et al. 2012). In our case, we see that both the supplier and mechanic dataset have low concentrations, meaning that their 10 most frequent terms have relatively low frequencies. We can confirm this by looking at the Zipf plots for both datasets (Figure 25 and Figure 40). The curvature on the left-most side of the plot falls below the adjusted estimated power-law (straight lines in orange and green, respectively). Moreover, the difference in concentration values can also be explained by the same curvatures, since it is less pronounced in the case of the mechanic dataset, meaning their frequencies are comparatively higher (they represent a bigger share of the total amount of occurrences). According to (Bank et al. 2012) this complicates the usage of dictionary and rule-based methods for Natural language processing.

Regarding vocabulary dispersion, we see high levels in both cases with a 10% difference in favour of the Mechanic dataset. Despite of what could be inferred, these metrics are not complementary to vocabulary concentration because vocabulary dispersion does not take term occurrences directly into

account (see sub section 3.2.5). High dispersion values indicate considerable amounts of spelling errors, which in turn can affect named entity recognition or part-of-speech tagging efforts and require the use of feature selection techniques to cope with the noise and burden of unnecessary features (Bank et al. 2012). Beyond confirming the known performance degradation provoked by the use of the mechanic dataset (see sub section 1.2.2), this metric brings a clear measurement of how different the two datasets actually are. This also sheds some light on how significant can be the improvement of *data feature quality* by applying some spelling correction step during pre-processing.

In conclusion, the different metrics and statistics are consistent with the observations made by (Kassner & Mitschang 2016), regarding the superiority of the supplier dataset to achieve higher accuracy in the classification task. We expect this to be the case as well in our instantiation.

In addition to the selected language statistics suggested by (Bank et al. 2012) and discussed so far, a lexical classification of each dataset brings attention to the potential gains that could be made when looking for additional concepts to enrich an existing taxonomy in the terms of each role.

Based on lists of function words for each language (shown in section 7.1 of the Appendix) we can identify a very high proportion of content words (see sub section 3.1.13), suggesting there are almost only meaningful terms. Function words on the other hand, account in both cases for less than 2% of the total terms considered, which translates into roughly 100 terms in each case. Upon closer inspection, these function words are also mostly uncommon, with only 32 (in the supplier dataset) and 21 (in the mechanic dataset) of them being among the top 1000 most frequent terms.

A possible explanation for this lack of function words is the overlap between the lists used to filter stop words and function words. In the case of English, 77.88% of the terms considered function words are also part of the stop words list. In German, 53.52% of the terms belonging to the function words list also appear in the stop words list. This means that several words that would be identified as function words in the lexical classification, are removed in the pre-processing of text corpora to build DTMs.

4.3 Study Object Characterisation

As we see from the previous results, the data that we focus on is different in many aspects to traditional structured data. This translates into several challenges to achieve a successful classification. To properly address them, we describe the properties that constitute a problem for traditional methods, conceptualise them and incorporate them into a study object characterisation.

We begin with the definition of "messy data" provided by (Kassner & Mitschang 2016):

> *"Text which consists of non-standard, domain-specific language, riddled with spelling errors, idiosyncratic and non-idiomatic expressions and OEM-internal abbreviations."*

The properties we can extract from this definition are:

- Non-standard content

  Text does not always follow standard grammar or syntax rules or conventions in terms of punctuation. This translates to challenges for concept recognition based on context identification (see sub section 3.1.12).

- Full of abbreviations/ technicalities

  It is common to find abbreviations in the reports which are understood by human experts but not by standard parsing software, thus reducing the amount of knowledge that can be extracted from this. Evidence of this are the most frequent terms in the *language-blind* pre-processing data for both the Supplier and Mechanic roles. This also represents an issue for certain pre-processing methods such as stemming, which depend on standard vocabulary (see sub section 3.1.9). At the same time the solution may involve named entity recognition or concept recognition techniques (see sub sections 3.1.11 and 3.1.12).

- Domain oriented (even sub-domain)

The text reports come from the automotive domain and quality sub domain. As a consequence, most of the terms revolve around car parts, problems, and customer complaints. It can also be expected that these context limitation makes certain terms change the meaning they normally have in a general context. This again is shown in the top terms ranks for both datasets.

Again this requires the use of concept recognition techniques (see sub section 3.1.12).

- Misspelled

This reflects the notion of data coming from free text under certain conditions that make its quality degrade. Text has typing errors or orthographic mistakes that difficult the pre-processing stage. This is noticeable when looking at term ranks with two versions of the same word, one with some spelling mistake and the correct one. This can represent a problem to properly calculate terms frequencies, which in excess can affect the data distribution on which certain feature selection metrics may depend (see sub sections 3.2.4, 3.2.5, and 3.2.6).

In addition to these, the data exploration shows other characteristics worth-adding to the base definition.

- Brief

We deal with very short text pieces whose maximum length never goes over the 100 words. Depending on the type of pre-processing applied and the dataset in question, this can decrease up to 32 words. On one side, this reduces the complexity of the text, since there is a limit to the depth of expression a text can achieve in so little space. On the other side, interpreting the intended meaning of words becomes more difficult because fewer terms that can serve as cues means it is harder to determine the context in which a word is used, a basic requirement to identify concepts (Schierle & Trabold 2008). Additionally, these differences in length can present a challenge for the weighting scheme employed when it comes to long documents that contain the same term many times (see sub section 3.1.5).

- From different perspectives

Texts are not only written in a particular context but also by different roles in the process. While they are not extremely different from one another, they do present significant differences in several aspects. As shown in the data exploration, documents from the Supplier role are significantly longer than those of the Mechanic. They refer to the same observations in different terms, as shown by the small overlap of terms between the two datasets and the inclination of each role to compose reports based on parts and symptoms or customers and complaints. Additionally, languages have different dominance in each dataset, with English being more common in the Supplier data, and German being slightly more common in the Mechanic data (both in terms of documents). These two phenomena combined induce by themselves a high-dimensionality problem (see sub section 3.1.2).

We can also expect a difference in relevance. For example, since mechanics are the first role in the process, many of the observations here are expected to be preliminary in nature. Symptoms and conditions described here may be superficial consequences of deeper causes to be determined by other roles later in the process. As a result, to avoid providing an algorithm with potentially contradictory input that can reduce its performance, it is better to consider data from different roles as independent inputs that can be provided to different instances of the same algorithm.

- Incomplete components

Data, understood as the combination of text data and the complementary structured data, has missing values that considerably decrease the amount of useful data for classification. This can even lead to redesigns of the modelling approach for the sake of not losing more records. The most notorious example in our data has to do with the dataset from the OEM. Representing 7.45% of the biggest dataset (both after filtering) and with a clearly visible concentration in a particular moment in time, this dataset would unbalance the data of other datasets if combined. Therefore, this role dataset is impossible to compare in a meaningful way to the other two.

Other examples of incompleteness occur at the moment of pre-processing. Reports without a valid date, with text whose language is hard to identify, and whose error code does not occur elsewhere in the dataset need to be removed.

- In one or two languages

Text written in multiple languages presents additional challenges in terms of its pre-processing, be it stemming, stop word filtering, or part of speech tagging. In all these cases, the fact of having to distinguish between English and German requires an additional step where records can be lost because of the inability to properly identify them. We see this happening in the *language-oriented* pre-processing results.

Moreover, the before mentioned tasks as well as others like name entity detection or concept recognition can be more difficult to implement because of the differences in logic they need to adapt to the nature of each language. A clear example is given by (Schierle & Trabold 2008): concepts that are expressed with one word in German may require multiple words in English.

- Many classification categories possible

Moreover, the amount of potential classes available make the classification difficult. This is because many multi classification algorithms were not designed to work with so many categories and despite of modifications to address this issue, they are not efficient enough to scale up to this order of magnitude (see sub section 3.3.4). A notable exception to this case is the Naïve Bayes algorithm, thanks to the Naïve assumption it takes to estimate the probability of a document represented by its feature vector to be part of a given category (see sub section 3.3.3). This assumption which originated from the fact that usually there are many documents to classify, also addresses the problem of having multiple categories. This leads to either looks for ways to reduce the amount of categories to use in the classification algorithm or to use only algorithms that are able to handle this many categories.

- Classification categories do not distribute evenly

As shown in the data exploration section, not all classification categories are equally likely to be used. The first reason is that by design, the error codes that can be allocated to each part type vary significantly, as shown in either Figure 12 or Figure 30. This means that when an error code belongs to a big error code family (part type), all other things being equal, the chances it has to be selected are lower than those of an error code from a smaller error code family.

The second reason has to do with time. As Figure 18 shows, there does not seem to be a clear pattern in the way error codes are assigned at any given point of time, let alone to present signs of seasonality. An example for this is that at the beginning of the time period, all observations can only have an error code belonging to one part type, but later on around year 2011, there are multiple options possible, making the classification in this period of time a lot harder than in the early years of the records. Finally, because of external unknown reasons, certain error codes may have more observations to be trained and tested, as it can be inferred from Figure 13 , Figure 14, Figure 31, and Figure 33.

These three conditions represent a difficult scenario for algorithms based on probabilities, since very common categories can affect the probabilities of very rare ones to be selected (see sub section 3.3.3). Moreover, this requires an elaborate sampling process that random selection cannot fulfil. This could be a problem as well for algorithms where positive and negative training data is needed (see sub section 3.3.4). This skewness is a problem that according to (Forman 2003), only worsens as data grows.

Considering all these properties we can arrive at a definition of messy data that more closely resembles the situation we deal with in our business scenario:

> *"Short texts written by different individuals about a single event in non-standard form, in multiple languages and with spelling mistakes; containing domain-specific language, and jargon abbreviations for the purpose of classifying each event in one many multiple categories."*

Based on the characteristics mentioned in this definition, we can develop a better understanding of the independent variables that operationalise the presumed cause in our research model *Availability of data features*: 1) *quantity of data features* and 2) *quality of data features* (see Figure 4 in chapter 2). We can expect characteristics like misspelling, incompleteness, abbreviations and the lack of proper grammar or syntax to affect the *quality of data features*, just as shown in sub section 4.2.4 with the values of vocabulary dispersion shown in Figure 45, whereas bilingualism, the increased diversity of categories, the existence of multiple roles, and the reduced length may influence the *quantity of data features*, also evidenced by the entropy and vocabulary concentration figures in the same Figure 45.

Knowing in detail the properties and challenges we need to deal with to perform our classification task, we can design a custom method that properly addresses them and as a result provides us with an optimal feature set to explore the effect relationship depicted in our research model (see Figure 4 in chapter 2).

4.4 Method to Select Optimal Classification Algorithm Configuration and Features

A complement to the conceptual architecture in section 4.1, the method to arrive at the feature set that results in the highest accuracy (as defined in sub section 1.2.4) when used with a particular classification algorithm configuration is shown in Figure 46. It covers the course of action taken to build a classification solution (instantiation artefact described in chapter 5) from a generic perspective that can be applied to all possible solutions derived from the use of our conceptual architecture. The steps go from the bottom layer (Feature Extraction) to the top one (Classification algorithm) to review the decisions and actions to go from the selection of a classification algorithm, to the algorithm configuration and feature set that provide the best results in the specified performance metrics.

Eventually, a logical formulation of this method could be implemented in a (semi) automated algorithm explorer to standardise the way this process is performed, facilitating fair comparisons and reducing the time required to do it. This could also be extended to target problems with a similar study object to the one we focus on here (unstructured domain-specific text data).

The steps are:

1. *Select classification algorithm*. Based on a list of requirements derived from the study object characterisation and the chosen metrics to evaluate the classification performance (one which is typically accuracy), a set of candidate algorithms is chosen. We assume there is not perfect match between any basic algorithm and the requirements of the study object. This asks for a comparison regarding their advantages and disadvantages to discard all but one, which will serve as base for the design of configurations later on.
2. *Extract all data features*. This involves retrieving the dataset from its source to then use different components from the Feature Extraction layer in our conceptual architecture to obtain as many features as possible both from the text and structured data parts. On the text part, this requires the use of tokenisation to generate feature vectors (to work in the Vector Space Model) and applying different levels of pre-processing to transform the original unstructured text into a more manageable format. Examples are removing stop words, punctuation signs, lowercasing, or spellchecking. Since the structured data is considered by definition to be in a convenient format, no pre-processing is applied to it.
3. *Choose document representation and weight scheme*. As part of the fundamental choices that needed in the Vector Space Model, the use of representation and weight schemes predisposes the suitability of applying certain feature selection techniques and classification algorithms. While we begin with a term frequency weighting scheme and a single-word-as-term representation, other options can be explored when coming to this step in a second iteration. Options for weights include TF-IDF or binary schemes found in the Feature Selection layer, while document representations could be n-grams of different sizes, phrases or concepts, all of which involve the use of a certain component from the Feature Extraction layer. In both cases, the choice is enacted in the creation of a Document Term Matrix with these characteristics.

4. *Explore data features*. A less concrete step than the previous two, this involves the creation of several graphical representations to look for patterns in their distribution over time, statistical distribution, value ranges and other quantitative aspects that can be observed by grouping the datasets under different perspectives. Using domain-specific notions or combining structured data is also encouraged, e.g. verification of valid dates, meaningful amount of observations, etc. The purpose is to find feature behaviours that can help discriminate between the classification categories and to identify challenges that the selected algorithm needs to deal with. In addition to this, a quantitative exploration may consists in the calculation of relevant metrics that summarise the text content in particular and the data set in general concerning certain aspects, e.g. term correlation, distribution fit, median document lengths, vocabulary size, vocabulary dispersion, etc.
5. *Assess possibility to derive features* and *calculate derivative features*. In combination with the feature exploration, this step aims to obtain additional features by integrating some of the original features. This includes calculating intervals between dates, averages or variances of certain values, normalised values based on other certain structured data (per kilometre, per day) and any other measure that makes sense in the domain context. The resulting new feature should be tested for correlation with its source features to avoid overestimating effects when building the classification model.
6. *Analyse features' utility for the algorithm*. This step is meant to deal with the problems of high dimensionality. Based on the patterns observed and the values obtained in the data exploration, and the application of one or multiple feature selection techniques (filters with evaluation metrics, statistical or dimension-based techniques), a decision must be taken regarding how many features should be considered and how will they be selected. While it still involves a certain amount of trial and error, the correct interpretation of the findings made so far should help to narrow down the feature selection techniques attempted.
7. *Assess dataset coverage*. Before making a final decision on which features to preserve to train a classification model, it is important to verify that the intended feature subset still covers most of the observations in the dataset. If this is not the case, and the amount of observation cannot be used to classify the majority of the categories, it is necessary to rethink the way data is processed starting from the document representation and weighting scheme.
8. *Select most suitable feature subset*. If the feature subsets are representative of the dataset and significant to support the execution of the classification algorithm, the feature selection techniques used to arrive to them are applied definitively to proceed. The plural form in the previous sentences is intentional, because at this point there is still reasonable doubt about which one is the feature selection technique that can contribute to the best performance results, something to be discovered later on in the method.
9. *Design algorithm configurations*. Algorithm configurations are built around *k* binary design choices concerning the selection, use, or way to use any of the elements considered in the Feature Extraction and Feature Selection layers of the conceptual architecture and that can affect the classifier's performance. Every choice becomes then a factor with two levels, high or low, present or absent. This results in 2^k configurations to be tested representing all possible combinations of factors' levels.
10. *Compare algorithm configurations' performance*. Every configuration is run at least twice to obtain performance metrics' values for each one of them. Doing so not only enables the statistical testing of the observed performance levels, it also reduces the likelihood of accepting inaccurate performance levels obtained by chance, since the inconsistency of each trial is easy to detect. This however, does not substitute proper sampling methods, such as cross-validation or stratification, to protect against "lucky sampling" effects.
11. *Choose final configuration*. The collected performance data can be used as input of a 2k experiment design to evaluate the magnitude and significance that every factor (or architectural choice) has on the performance metric values. In this way it is possible to identify the components that improve the most the final result to focus more on them and how they do it at

the expense of others which make little difference. It is important to remember, that this results suggest a best configuration based only on the two levels each factor has, not on all the possible levels the factor can actually have. An example for this would be choosing between two weight schemes, even though our conceptual architecture considers at least three.

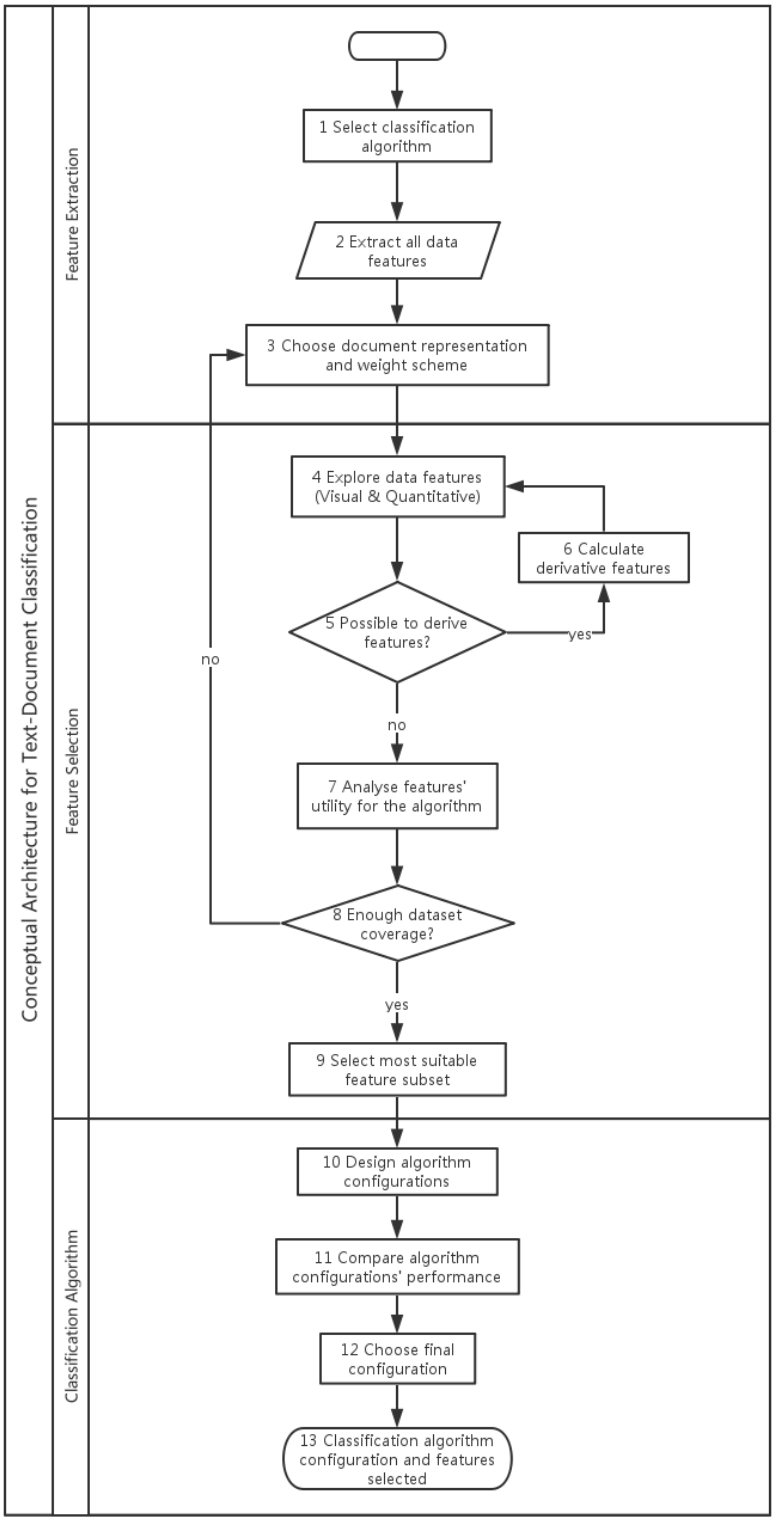


Figure 46 Method to select the best classification algorithm configuration and feature subset given the selected classification algorithm

5 Experiments and Evaluation

In this chapter we exemplify the application of the framework proposed in chapter 4 to build a classification solution for our application scenario. We go through each of the steps described in the method introduced in section 4.4 while also explaining additional details regarding this instantiation whenever necessary. We present the results obtained with our solution and discuss their performance in comparison to the previously implemented solution from (Kassner & Mitschang 2016). Finally, we evaluate the artefacts that led to the creation of our solution in light of the Design Science methodology.

5.1 Classification Algorithm Selection

As the first step from our method, we create a list of requirements that an algorithm should fulfil to handle the particular properties of our study object as identified by both the characterisation in section 4.3 and the problem description in sub section 1.2.2.

We look for an algorithm with the following characteristics

1. *Time efficient*. We look for an algorithm that can be comparable to the processing time per report of the equivalent k-NN implementation. This means having values around the 0,14 seconds per report of the *bag of words* approach from the k-NN implementation.
2. *Can handle many features*. A consequence of the *bag of words* approach, where every word is considered a feature, there is a considerable amount of features even in brief texts as ours to build a feature vector. While feature selection techniques can help partly address this abundance of features, the selected algorithm should still be able to deal with enough features as to keep the subset representative of the original reports.
3. *Can handle many classification categories*. The selected algorithm should be able to perform multi class text classification (see sub section 3.3.2) with hundreds of categories.
4. *Robust to data skewness*. Despite the evident skew of data towards some categories, the selected algorithm should be able to maintain a reasonable performance and to overcome the expectable errors in estimations to classify very uncommon categories.
5. *Easy integration of unstructured and structured features*. The selected algorithm should be able to combine structured and unstructured features in the same classification model with little or no transformation of either kind of data.
6. *Generation of multiple category suggestions*. Instead of just offering a single most likely category, the selected algorithm should provide several category alternatives to classify each document. This is to support the work of a human expert as described in sub section 1.2.2.

5.1.1 Algorithm Selection Rationale

(Khan et al. 2010) describe the k-NN, Naïve Bayes and Support Vector Machines as the typical algorithms of choice for text classification. However, since the k-NN is already implemented for our application scenario, we replace it with the decision trees algorithm to make our comparison. We discuss each algorithm's advantages and disadvantages.

5.1.1.1 Naïve Bayes

This model is known for its simple implementation, for which no adaptation of the document term matrices is needed. The composing feature vectors of the matrix are used directly to calculate probabilities for each classification category. Moreover, thanks to its Naïve assumption when calculating the conditional probability of belonging to a category given the document vectors (see sub section 3.3.3), it can scale to handle big amounts of data and categories. Additionally, the algorithm is robust to failures in the calculation of probabilities due to small training sets, a useful property

considering the limited amount of observations some error codes have (see sub sections 4.2.1.1 and 4.2.2.1). Finally, according to (Liu et al. 2013), the use of token (or term) frequency is an effective scheme for statistical algorithms, such as this one.

On the negative side, we can consider its average performance in terms of accuracy compared to other algorithms, something that nevertheless can be improved with multiple adaptations (Khan et al. 2010). Perhaps the integration of structured data as just another feature among a myriad others obtained from text can lead to a weak influence of the structured data features, but this is to be seen in experimentation.

5.1.1.2 Support Vector Machines

Known as a top performer for text classification, this algorithm can handle a big amount of testing/ input data efficiently thanks to its hyperplane representation of positive and negative category spaces (see sub section 3.3.4). Also, when dataset are highly dimensional, it is possible for this algorithm to transform this highly dimensional feature space into a simpler representation using kernel functions.

However, its greatest weakness appears when dealing with extreme multi class classification problems such as ours. Since it is designed as a binary classification algorithm, it needs to run as many times as there are categories, thus leading to an inefficient execution time and complex adaptations to provide the necessary positive and negative training data without skewing the samples towards the negative side (since at any given moment all categories are negative training data except one). Using feature selection mechanisms to address this result in performance degradation.

In addition to this, configuring additional components to enable the algorithm to deal with high-dimensionality data (such as kernel functions or a slack variable) represent an additional burden that is not needed with other algorithms.

5.1.1.3 Decision trees

Known for its speed and scalability, this algorithm can easily integrate structured data features (something that also helps address overfitting), given the fact that in essence it works as a chain of if-then rules (see sub section 3.3.5). It however has the risk of either being too efficient to classify by using a small amount of features and overfitting the training data or being very complicated as it keeps growing along with the dataset. This last option also sacrifices its main advantage: intelligibility by humans. Moreover, by design it is supposed to assign just one category to each report. Adapting it to provide a list of suggested categories (as needed by our application scenario) entails then additional complexity.

5.1.2 Final Selection

As we can see, given the particular characteristics of our study object and application scenario, specifically that 1) this is an extreme multi class classification, and that 2) we aim to provide a list of suggested categories instead of a single one, the Naïve Bayes algorithm stands out as the most straightforward alternative to test our method and conceptual architecture. By choosing it, we can redirect the focus from the classification algorithm alone, to the complementary feature extraction components and feature selection techniques that also form part of our conceptual architecture.

5.2 Technical Setup

Figure 47 shows the environment where the experiments run. In white we show the necessary software components that serves as foundation for our instantiation, while the created components are coloured in grey. Arrows indicate the flow of data from its source to the classifier logic. We describe this setup in a bottom-up fashion.

Our environment is a 64-bit Lubuntu server version 4.8.2-19 with a quad-core CPU running at 3.2 Ghz, 100 Gb of storage and 46 Gb of RAM. It hosts all our components and is available exclusively for these

experiments. Data is stored in several tables in a Postgres database (version 9.5.3). It is loaded to the R environment with the RPostgresql interface package.

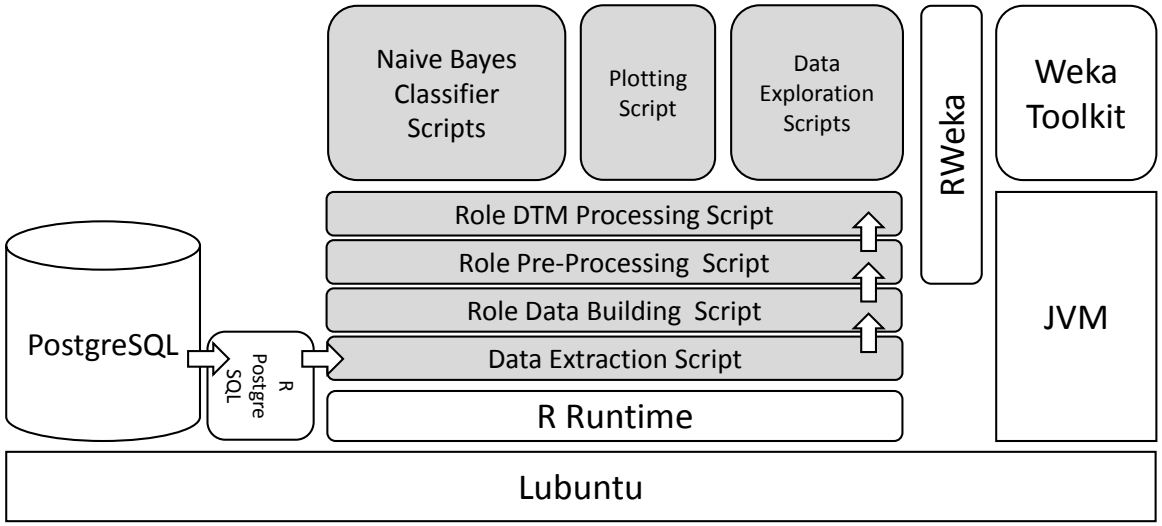


Figure 47 Technical Setup for the implementation of the Naive Bayes classifiers

We run an R environment in version 3.3.0 with the following additional packages installed (dependencies not included):

1. *RTextTools*: To test multiple classifiers in a simplified manner. Version 1.4.2
2. *igraph*: To check for power-law fit of text data. Version 1.0.1
3. *Rgraphviz*: To plot correlations of terms as a graph with links of different strengths. Version 2.14.0
4. *tm*: To pre-process the text reports corpora (stop words removal, stemming, removing numbers). Version 0.6-2
5. *textcat*: For language detection. Version 1.0-4
6. *RWeka*: Interface to use the Weka Naïve Bayes classifier. Version 0.4-27
7. RPostgresql: To retrieve data rows from the source database. Version 0.4
8. *qdap*: To make lexical classification. Version 2.2.4
9. *sampling*: To do stratified sampling. Version 2.7
10. *qualityTools*: To run the 2k Experiments and plot effects. Version 1.55

Weka toolkit in version 3.9.0. Classes are accessed with the RWeka interface package.

We also have a Java virtual machine version 1.8.0_91 to support the execution of the Weka toolkit.

The Naïve Bayes classification solution is composed of several R scripts that perform some part of the steps to arrive at the classification train and execution. They follow a "pipeline" design where a script can be replaced with other similar ones, for example to create feature vectors with terms made of two words, or to perform different kinds of pre-processing. The scripts in the solution include:

1. *Data extraction script*: Retrieves data for each role from the database and stores it into a corresponding data frame (R data structure).
2. *Role data building script*: It filters observations with invalid dates, removes observations whose error codes appear only once (singletons), builds explicit labels of the error codes, aggregates

text reports into a single field (for the roles whose reports are split in multiple fields), and calculates the derivative features *standard mileage* (mileage per day) and *driving time* (days elapsed from the moment a car is first admitted to drive to the moment when it is taken to repair).

3. *Role pre-processing script*: It creates a text corpus (data structure from the tm package) and performs the two kinds of pre-processing. The *language-blind* pre-processing which 1) turns it into lowercase letters, 2) filters English and German stop words, 3) removes all numbers, and 4) removes all punctuation signs. Meanwhile, the *language-oriented* pre-processing 1) turns it into lowercase letters, 2) identifies the document's language, 3) filters only the stop words of the identified language, 4) removes all numbers, 5) removes all punctuation signs, and 6) stems the remaining terms in the document according to the identified language.
4. *Role DTM processing script*: It builds document term matrices, tests for the existence of power-law behaviour, and obtains lists of all terms ordered by frequency to ease further calculations.

The last three scripts all depend on the results of the previous scripts to achieve different purposes, they are independent from each other.

5. *Naïve Bayes classifier script*: Performs the classification task with the given parameters for role, use of structured data, weighting scheme, pre-processing, and number of terms. It also calculates accuracies with lists of 1, 5, 15 and 25 suggested categories.
6. *Plotting script*: Generates graphics of each role dataset based on their document term matrix representations. It plot correlation among top terms, Zipf plots of term frequencies and simple plots of terms ordered by frequency.
7. *Data exploration scripts*: Generates graphics of each role dataset regarding structured data and the distribution of reports across categories and time.

5.3 Extract All Data Features

We begin by retrieving records for each role from the Postgres database and loading them into R. At this point, data is organised into rows of results from an SQL query in the form of a R data frame. The extraction of features from this data frame occurs at different stages of the script pipeline depending on the kind of feature. Structured data is obtained at the *role data building script*, simply parsing extracted text strings into date formats or mileage values into numeric formats. Text features are obtained in the *role DTM processing script* by default using a single-word-as-term representation and term frequency weights.

5.4 Choose Document Representation and Weight Scheme

The document representation in all cases is set to single words as terms since we are exploring the spectrum of feature selection in the *bag of words* approach. For the weight scheme we use both term frequency and term frequency- inverse document frequency (TF-IDF) to account for the differences in documents' lengths. These two weighting schemes indicate more than just the presence of a word in a document, they also account for multiple mentions, thus allowing more fine-grained probability calculations, a feature selection technique based on frequencies, and the use of language statistics to explore the dataset (as shown in sub section 4.2.4).

5.5 Data Exploration

In this step we include the complementary steps *derivation of features* and *feature assessment* for the classification task. Data exploration is covered in detail in section 4.2. Based on this exploration we determine to use only the supplier and mechanic datasets. As structured data features we decide to employ the admission-to-drive date and driving time for both roles, for reasons covered in the same section.

Regarding the vast amount of unstructured data features (those obtained from text), we select features according to their frequency values and the power-law distribution they follow. Based on the assumption that neither very frequent features (since they tend to be present in many documents) nor very uncommon ones (since they appear in a few documents) help discriminate among the various categories

available, we select one thousand features (or terms) following a 80/20 Pareto principle. We exclude the first top terms that account for 20% of the total occurrences in the dataset and use the following thousand. With this we also expect to avoid the common correlation issues at the head of a power-law distribution that can difficult classification (see sub section 3.2.6).

When it comes to sub setting the language-oriented matrices, we also begin our selection after the top terms accounting for 20% of the occurrences in each language, but the thousand features are obtained in proportional rates from each language to preserve their original representation: 58%/42% in favour of English for the Supplier dataset and 54%/46% in favour of English in the Mechanic dataset.

In addition to this subset selection, we employ the whole set of features to have reference values in accuracy and processing time to compare against.

5.6 Coverage

The feature selection technique just described in the previous section allows us to achieve a good coverage in terms of occurrences while drastically reducing the processing time. As Figure 48 shows, while the thousand terms chosen represent 10% to 15% of all distinct terms in their respective document term matrices, they account for 63% to 75% of all term occurrences in the reports. Even though these values can still be optimised to obtain a better trade-off between the number of distinct terms and the number of occurrences included, the selection is considered sufficient to exemplify the utility of a power-law based selection.

Configuration type	Feature space (total number of terms)	1000 terms share as distinct terms	1000 terms share as per occurrences
Supplier Language-blind	**8219**	**12,17%**	**70,64%**
Supplier Language-oriented	**9307**	**10,74%**	**63,05%**
Mechanic Language-blind	**8989**	**11,12%**	**75,24%**
Mechanic Language-oriented	**6579**	**15,20%**	**66,98%**

Figure 48 Coverage of occurrences in different configuration types for different sub setting criteria

It is worth mentioning that there seems to be a clear distinction in coverage between the two kinds of pre-processing in favour of the *language-blind* variant.

5.7 Naïve Bayes Algorithm Configurations

There are five different choices that we consider to design the configurations, given the fact that they can alter the accuracy performance if we choose one of the two proposed levels for each of them. They are:

1. *Role*: Using text reports from the supplier or mechanic datasets. While the difference in performance due to role data already has evidence from the previous k-NN implementation, we keep this factor to be able to observe the effects of other factors in each dataset.
2. *Weight scheme*: We compare the term frequency and term frequency – inverse document frequency schemes to verify the effectiveness of the latter in giving more relevance to uncommon features.
3. *Pre-processing type*: We compare the *language-blind* and the *language-oriented* approaches to estimate the effect of different degrees of pre-processing in the classification results.
4. *Use of structured data*: We evaluate the impact of using structured data (driving time and admission-to-drive date) in the classification results.

5. *Number of terms*: To assess the effectiveness of the power-law based selection technique compared to executing the classification with all terms.

These factors result in 32 (2^k, with k=5) configurations to run our Naïve Bayes classifier, comprising all possible combinations of factors' levels.

5.8 Algorithm Configuration Results

As a result of the first four scripts in the "R pipeline" (see section 5.2) we have 12 Document Term Matrices to serve as input for the 32 configurations. Each role has 6 DTMs with the following differences among them:

1. Language-blind pre-processing, term frequency weights.
2. Language-blind pre-processing, TF-IDF weights.
3. English-oriented pre-processing, term frequency weigths.
4. English-oriented pre-processing, TF-IDF weights.
5. German-oriented pre-processing, term frequency weights.
6. German-oriented pre-processing, TF-IDF weights.

We run each configuration with its corresponding DTM (or DTMs for the *language-oriented* configurations). 80% of the documents are used to train the model (*training*) and the rest is used for *testing*. This split is made with a stratified sample based on the total amount of documents labelled in each category. In the case the amount of documents for the error code is too small to allow a 4:1 split (only two reports), one document was used for training and the other one for testing.

Due to the language detection step, some error codes in the *language-oriented* DTMs may have only one document; this despite of the singleton removal made by the *role data building script* (see section 5.2). This is the combination of two phenomena: 1) the language detector component may assign the two reports of the same error code to different languages or simply may not assign one of them to any language, and 2) the removal of singletons occurs before the language detection step. Since the solution to this report loss heavily relies on the improvement of language detection components, we proceed with these datasets and leave enhancements for a future time (see section 2.8). In these cases the only report available is used for training the classifier.

We present results according to the role dataset used, starting with summary tables containing the exact values for all configurations. Then, to improve readability and analysis, we present graphs grouping configurations by weight scheme and subset used.

5.8.1 Supplier Dataset

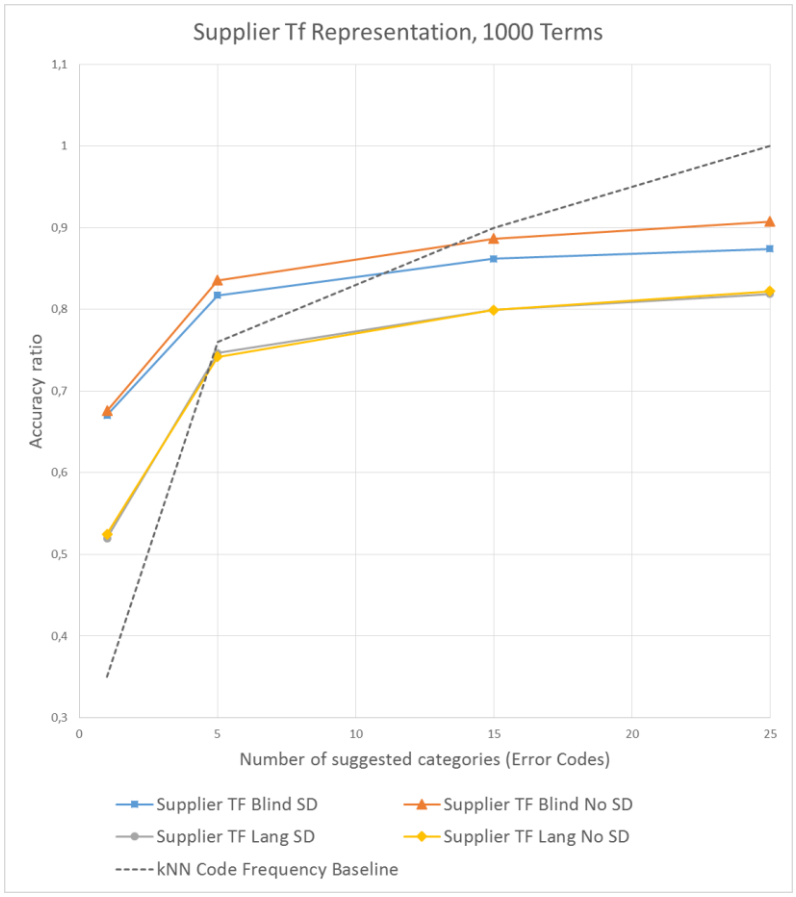
Figure 49 shows accuracy levels and processing times (considering only the testing time) of all 16 configurations corresponding to the supplier role. In addition, the *code frequency baseline* used in the k-NN implementation (see sub section 1.2.2) is also shown for comparability.

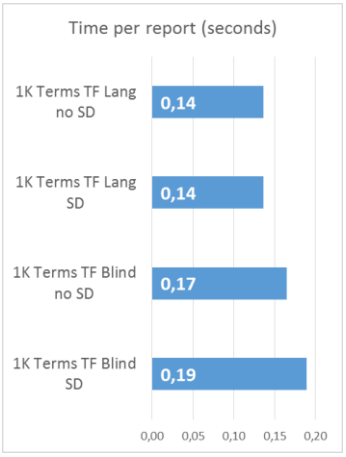
One of the first differences that stand out is the remarkable variance in processing time of the configurations using a subset of a thousand terms compared to those that use the whole feature set. While all configurations with a subset have processing times below 0.2 seconds per report, times for the rest of configurations are above two minutes. These values make the subset configurations comparable to the *bag of concepts* approach from the k-NN implementation with regards to processing time but not in terms of accuracy.

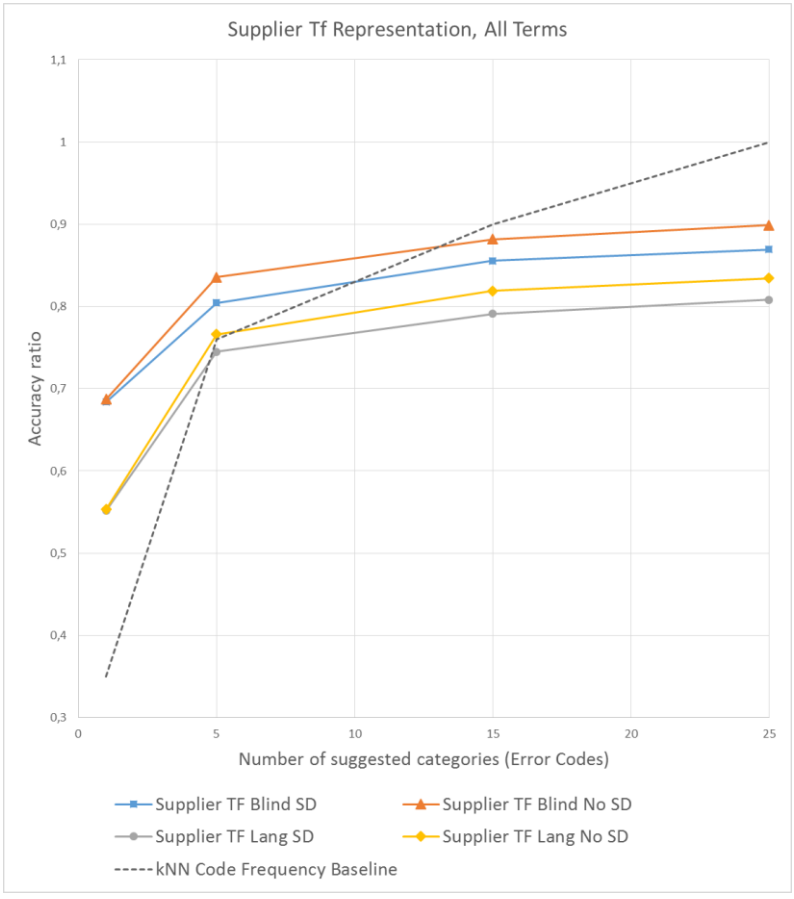
Concerning the accuracy levels and how they fare against the *code frequency baseline*, we see that the classifier performs better than the baseline in all configurations only for the first suggestion of error codes (accuracy at 1). With a list of 5 suggestions, 10 out 16 configurations still perform better. However with lists of 10 suggestions or more, the baseline outperforms all configurations.

Feature Selection	Weight Scheme	Pre-Processing	Use of Structured Data	Accuracy at 1	Accuracy at 5	Accuracy at 15	Accuracy at 25	Total classification time (minutes)	Time per report (secs or min)
kNN Baseline Code Frequency				35%	76%	90%	100%	NA	NA
1000 Terms	Term Frequency	Language-blind	With Structured Data	67,0%	81,7%	86,2%	87,4%	4,68	0,19
1000 Terms	**Term Frequency**	**Language-blind**	**No Structured Data**	**67,6%**	**83,5%**	**88,7%**	**90,8%**	**4,08**	**0,17**
1000 Terms	Term Frequency	Language-Oriented	With Structured Data	52,0%	74,7%	79,9%	81,9%	3,37	0,14
1000 Terms	Term Frequency	Language-Oriented	No Structured Data	52,5%	74,2%	79,9%	82,2%	3,37	0,14
1000 Terms	TF-IDF	Language-blind	With Structured Data	61,5%	79,4%	83,8%	86,0%	4,09	0,17
1000 Terms	**TF-IDF**	**Language-blind**	**No Structured Data**	**61,8%**	**82,3%**	**88,4%**	**90,6%**	**4,05**	**0,16**
1000 Terms	TF-IDF	Language-Oriented	With Structured Data	53,0%	73,8%	79,4%	81,0%	3,43	0,14
1000 Terms	TF-IDF	Language-Oriented	No Structured Data	52,0%	73,2%	78,4%	80,3%	3,42	0,14
All terms	Term Frequency	Language-blind	With Structured Data	68,4%	80,4%	85,6%	86,9%	54,94	2,22
All terms	**Term Frequency**	**Language-blind**	**No Structured Data**	**68,7%**	**83,5%**	**88,2%**	**89,9%**	**55,28**	**2,24**
All terms	Term Frequency	Language-Oriented	With Structured Data	55,2%	74,5%	79,1%	80,8%	51,75	2,10
All terms	Term Frequency	Language-Oriented	No Structured Data	55,3%	76,6%	81,9%	83,4%	52,86	2,14
All terms	TF-IDF	Language-blind	With Structured Data	70,3%	82,6%	87,1%	88,5%	55,30	2,24
All terms	**TF-IDF**	**Language-blind**	**No Structured Data**	**71,0%**	**85,0%**	**89,1%**	**90,4%**	**56,62**	**2,29**
All terms	TF-IDF	Language-Oriented	With Structured Data	54,8%	75,9%	80,7%	82,4%	52,78	2,14
All terms	TF-IDF	Language-Oriented	No Structured Data	55,0%	76,4%	80,8%	82,4%	52,96	2,14

Figure 49 Results for the algorithm configurations with Supplier data







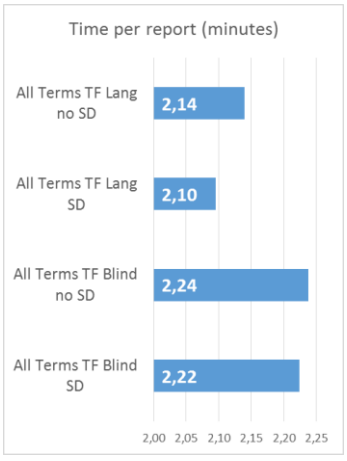
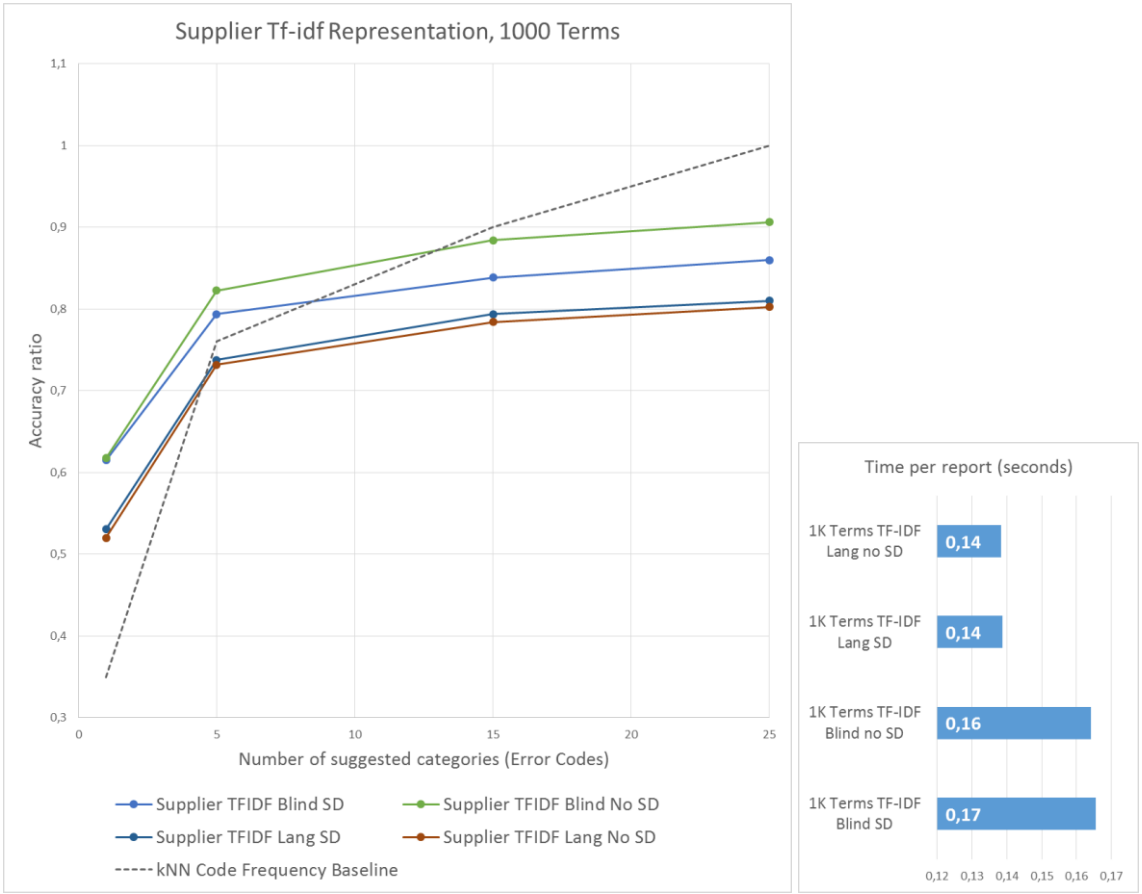


Figure 50 Accuracy and processing time plots for the Supplier set with TF weights

Looking at a graphical representation of the term frequency part of the Supplier results (Figure 50), we can see a clear distinction between *language-blind* and *language-oriented* configurations, both in terms of accuracy and time. While *language-oriented* configurations take less time to classify the same amount of reports, they do so with a consistently lower accuracy.

Accuracy at the first suggestion varies in this group of configurations (both graphs) between 52% and almost 69%. With the largest list of suggestions, accuracy ranges between roughly 82% and 90%. While the range of variation decreases steadily as the list of suggestions grows, the more remarkable change occurs as the list grows from 1 to 5 elements. All configurations strongly raise their accuracy in this interval, particularly the *language-oriented* configurations that raise approximately 20%.

When looking at the effects of using structured data, contrary to expectations, it tends to lessen accuracy, with its negative effect growing as the list of suggested error codes (categories) increases. From the processing time perspective, the effect is unclear, given that otherwise identical configurations can increase, decrease or maintain their processing time with the use of structured data. However, regarding pre-processing, *language-blind* configurations seem to take slightly longer to complete. Also interesting to point out, the use of a feature selection technique (upper graph) seems to neutralise the effect in accuracy of structured data in the *language-oriented* configurations, something that does not occur with the *language-blind* counterparts.



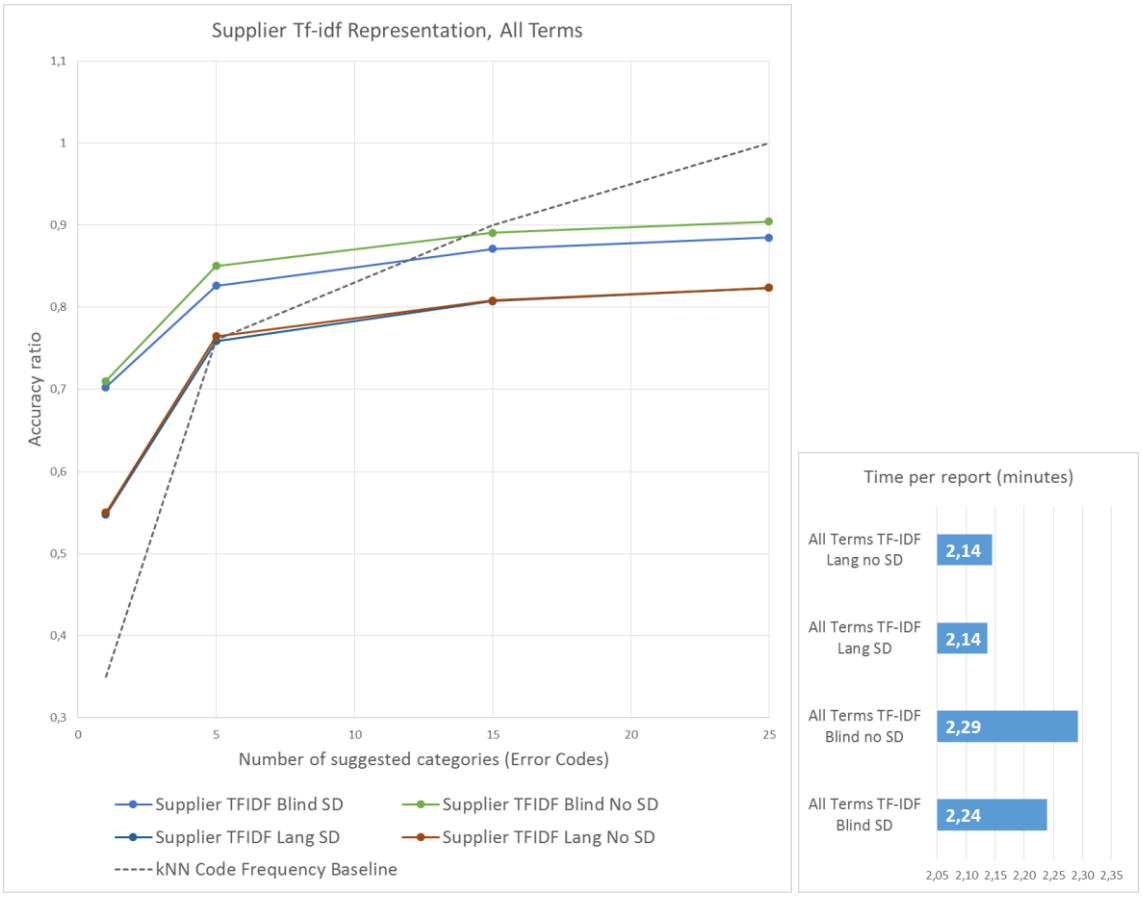


Figure 51 Accuracy and processing time plots for the Supplier set with TF-IDF weights

The portion of results with TF-IDF weights, shown in Figure 51, presents a similar behaviour to that of the TF configurations in many respects: the performance distinction between *language-blind* and *language-oriented* configurations remains, so does the processing time difference between configurations with a term subset (upper graph) and those using all terms (lower graph), and the sharper accuracy increment from 1 to 5 recommendations in all configurations, with particularly higher values for the *language-oriented* kinds. Finally, when looking at processing times, the difference between *language-blind* and *language-oriented* configurations remains visible.

However, we can also appreciate several differences. With this weight scheme, *language-blind* configurations achieve a slightly higher accuracy using all terms in the feature set (lower graph), but also see a greater loss when using only a thousand terms instead (upper graph). Moreover, the effects of structured data seem to be neutralised for all *language-oriented* configurations, not just those with a feature sub set.

In conclusion, we see that the best configurations tend to be those with a language-blind pre-processing and with no use of structured data, regardless of the weight scheme and feature subset. However, with just one suggested category (accuracy at 1), there are losses in their accuracies when using a feature subset (upper graphs in both figures Figure 50 and Figure 51), which may be as big as 9.2% or as small as 1.1% depending on the weight scheme used. This strongly contrasts with the situation observed at accuracies with the longest list of suggestions (of the same configurations), where the use of a feature subset actually increases values up to 0.9%. Since our goal is to provide overall good lists of suggestions to human experts, this supports the idea of selecting a portion of all available features to perform classification.

5.8.2 Mechanic Dataset

Figure 52 shows accuracy levels and processing times (considering only the testing time) of all 16 configurations corresponding to the mechanic role, along with the *code frequency baseline* from the k-NN implementation for ease of comparison.

When comparing accuracy levels to the baseline, we immediately notice the stark difference in quality of the mechanic dataset compared to its supplier counterpart. Only three configurations have better accuracies when suggesting a single error code. Every other scenario is dominated by the baseline. Even if their processing times are overall better than those of the supplier dataset (with some configurations even crossing the 2 minutes threshold), their bad accuracy performance make this dataset unsuitable for classification.

Feature Selection	Weight Scheme	Pre-Processing	Use of Structured Data	Accuracy at 1	Accuracy at 5	Accuracy at 15	Accuracy at 25	Total classification time (minutes)	Time per report (seconds)
kNN Baseline Code Frequency				35%	76%	90%	100%	NA	NA
1000 Terms	**Term Frequency**	**Language-blind**	**With Structured Data**	**38,2%**	**62,4%**	**75,9%**	**79,7%**	**2,66**	**0,14**
1000 Terms	Term Frequency	Language-blind	No Structured Data	30,9%	59,2%	76,2%	81,7%	2,62	0,14
1000 Terms	Term Frequency	Language-Oriented	With Structured Data	20,9%	41,5%	55,7%	62,0%	2,38	0,12
1000 Terms	Term Frequency	Language-Oriented	No Structured Data	12,4%	31,2%	47,6%	54,4%	2,35	0,12
1000 Terms	**TF-IDF**	**Language-blind**	**With Structured Data**	**33,5%**	**53,2%**	**67,2%**	**72,1%**	**2,62**	**0,14**
1000 Terms	TF-IDF	Language-blind	No Structured Data	27,5%	48,3%	63,1%	68,7%	2,55	0,13
1000 Terms	TF-IDF	Language-Oriented	With Structured Data	21,2%	40,5%	55,7%	63,0%	2,38	0,12
1000 Terms	TF-IDF	Language-Oriented	No Structured Data	13,1%	31,1%	45,2%	55,0%	2,38	0,12
All terms	**Term Frequency**	**Language-blind**	**With Structured Data**	**39,1%**	**62,4%**	**76,2%**	**80,1%**	**39,57**	**2,08**
All terms	Term Frequency	Language-blind	No Structured Data	33,4%	62,2%	77,8%	83,0%	40,08	2,10
All terms	Term Frequency	Language-Oriented	With Structured Data	21,3%	43,5%	56,7%	63,8%	25,89	1,36
All terms	Term Frequency	Language-Oriented	No Structured Data	14,7%	33,3%	47,9%	54,9%	25,60	1,34
All terms	**TF-IDF**	**Language-blind**	**With Structured Data**	**38,6%**	**62,2%**	**74,9%**	**79,4%**	**39,73**	**2,08**
All terms	TF-IDF	Language-blind	No Structured Data	33,5%	59,5%	76,2%	81,7%	39,16	2,05
All terms	TF-IDF	Language-Oriented	With Structured Data	21,4%	41,2%	55,3%	60,6%	25,74	1,35
All terms	TF-IDF	Language-Oriented	No Structured Data	15,3%	32,6%	47,8%	53,2%	25,18	1,32

Figure 52 Results for the algorithm configurations with Mechanic data

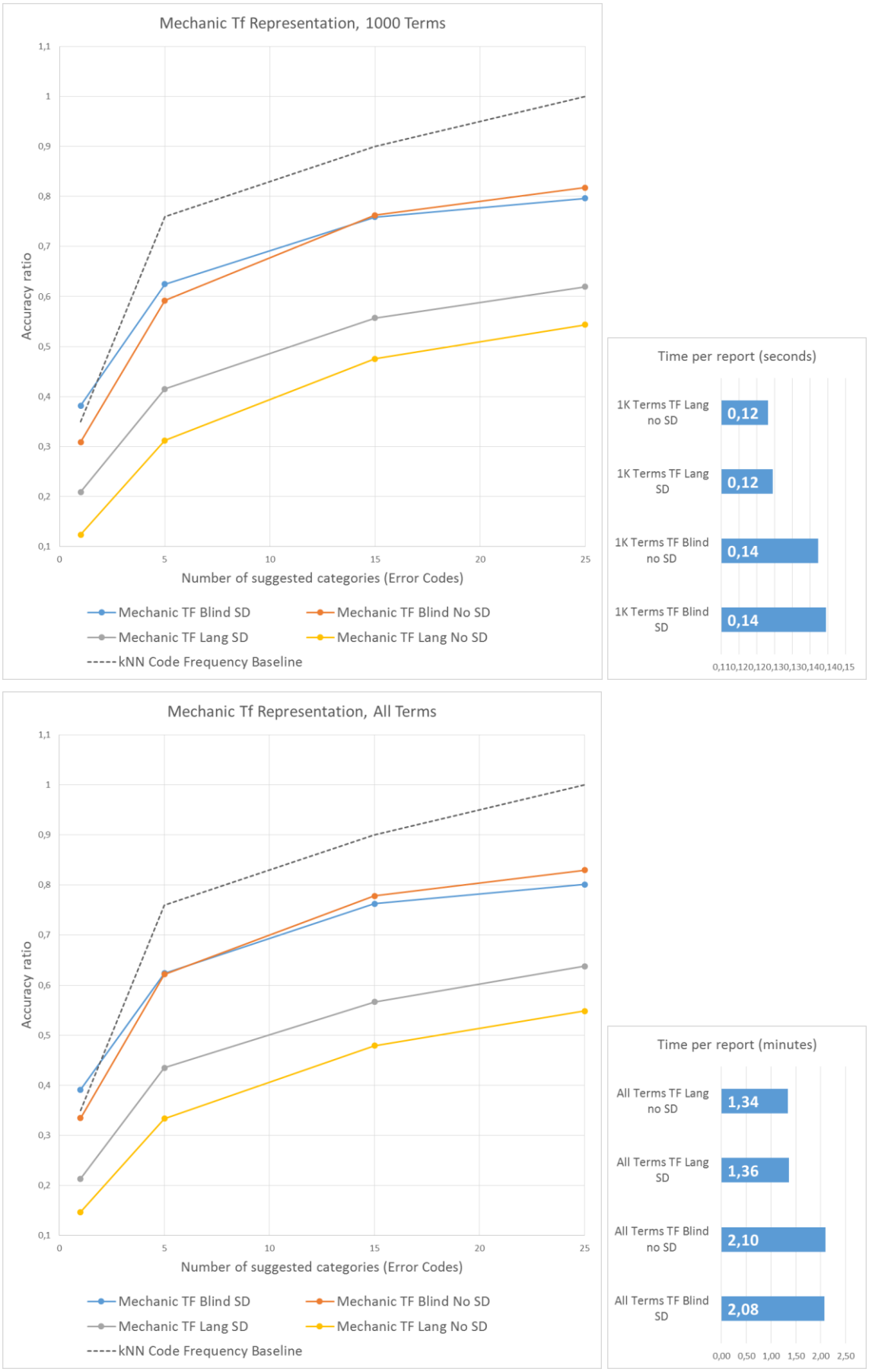


Figure 53 Accuracy and processing time plots for the Mechanic set with TF weights

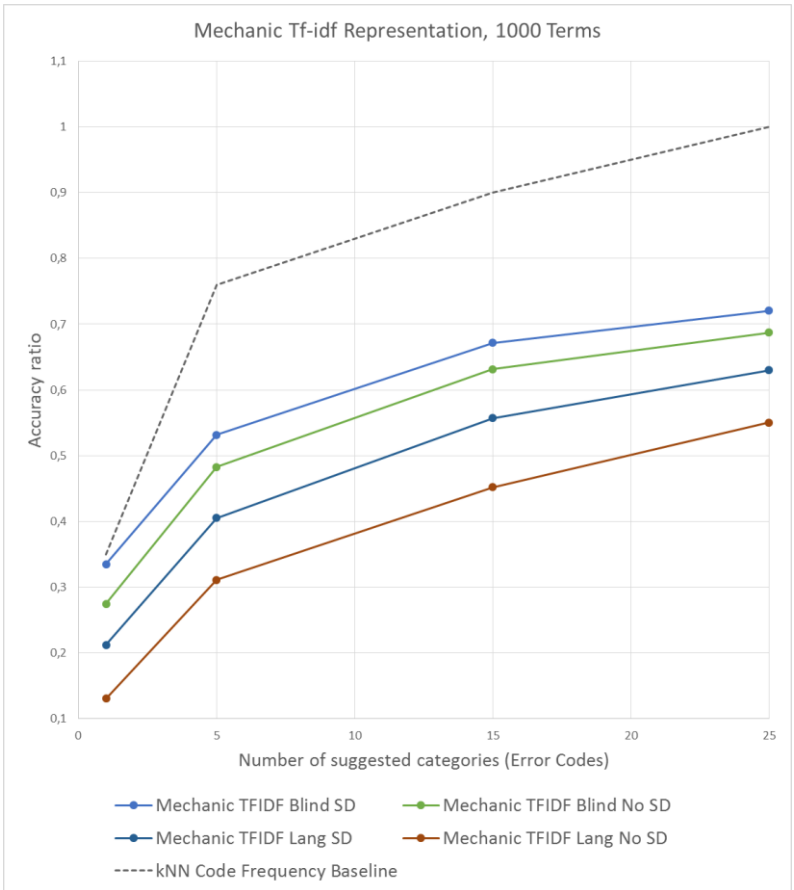
When looking at the visual representation of the mechanic results with term frequency weights (Figure 53), we can appreciate different patterns to those in the supplier dataset. Beyond the generalised accuracy fall, the use of structured data seems to have totally different effects, this time making a clear contribution to accuracy in *language-oriented* configurations and twisting performance in *language-blind* ones. With suggestion lists of just one element, the use of structured data in *language-oriented* configurations can mean improvements from 6.6% to 8.9%, in favour of the configurations using a feature subset (upper graph). With 25 suggested categories, the improvements go from 7.6% to 8.9%, but this time in favour of the configurations using all features (lower graph).

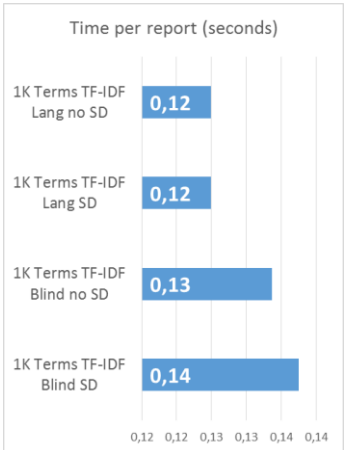
As mentioned earlier, the impact of structured data in the *language-blind* configurations is more complex. At suggestions of a single category, using structured data improves accuracy from 5.7% to 7.3% in favour of the configurations using a feature subset (upper graph). However, when looking at accuracies with suggestions of 25 elements, the effect is the opposite: using structured data now decreases values by 2% to 2.9% also in favour of the configurations using a feature subset (upper graph), meaning less accuracy loss.

Overall, this behaviour seems to benefit most of the time the configurations using a feature subset (upper graph), showing them as more stable options across the different accuracy cut-offs, even though their performance is slightly worse than those using all features available (lower graph).

Considering the elapsed time to execute each configuration, although the two patterns found in the Supplier dataset still hold (1. *language-oriented* configurations being faster than *language-blind* ones, 2. Configurations with feature subsets (upper graph) being faster than configurations using all features (lower graph)), the difference between the *language-blind* and *language-oriented* configurations seems to be smaller. Possibly because of the reduced size of the dataset compared to the Supplier one.

All in all, the mechanic dataset does yield worse results than the supplier dataset. We can imagine that having less documents (that also happen to contain less words), in addition to the many other characteristics summarised in sub section 4.2.4 are the causes behind this reduced performance. Determining whether this is true or not, is left to future research.





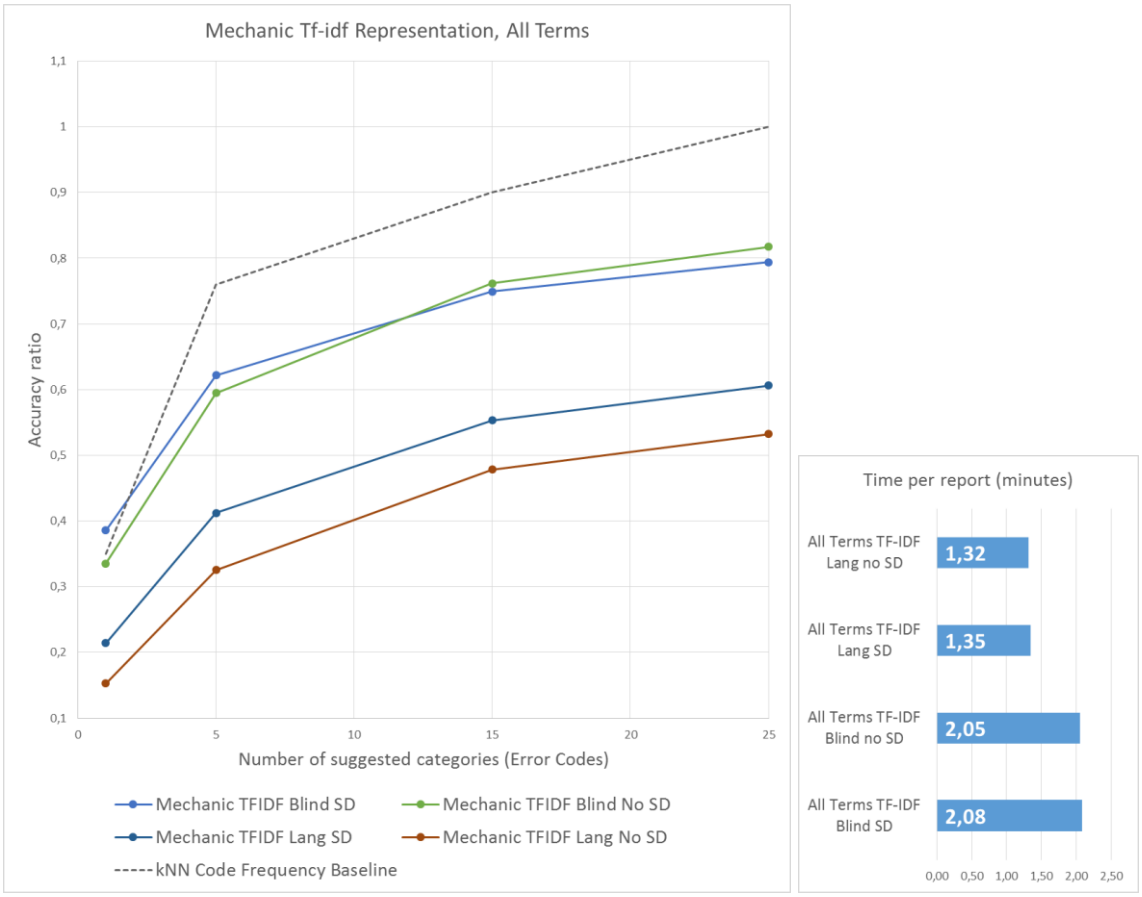


Figure 54 Accuracy and processing time plots for the Mechanic set with TF-IDF weights

Finally, Figure 54 shows the results for the configuration with TF-IDF weights. In this group, the *language-blind* configurations using a feature subset (upper graph) suffer a sharp loss in accuracy compared to their full feature set counterparts (lower graph). This results in all configurations using a feature subset (upper graph) to perform below the level of the *code frequency baseline*. This strong reduction does not happen however with the *language-oriented* configurations that also use a feature subset (upper graph).

Concerning the use of structured data, all configurations with TF-IDF weights tend to be benefited by its use, with the sole exception of the *language-blind* configuration using all features in the set (lower graph), which loses its advantage over the similar configuration without structured data at 25 suggestions.

When it comes to processing time, the behaviours observed in the mechanic configurations with term frequency weights (Figure 53) are still observed, implying that weight schemes do not have an impact on processing time.

5.9 Algorithm Configurations Evaluation

As evidenced by the results analysis of the previous section, it is hard to determine the real significance of using one configuration instead of another very similar. Accuracy values do not have a clear trend variation over different configurations, with the most problematic factors being the use of structured data and the use of a feature selection technique to subset the feature set. Depending on the dataset used, the accuracy cut-off specified or the kind of pre-processing used, these two factors can have increased, reduced, or inverse effects on the accuracy levels. Besides, by analysing execution data of just one run, we are exposed to attribute significance to variations due to chance.

To address these issues and to be able to more effectively determine the way each factor affects accuracy, and even whether a factor has a real effect at all, we run four 2^k experiments with two replicates each. Two experiments deal with the supplier dataset and accuracy cut-offs at 1 and 25, whereas the other two use the mechanic dataset and the same cut-off levels. By targeting accuracy at the two extreme cut-offs we can appreciate changes in the way each factor affects the classification performance.

The 2^k design is useful for this particular situation, as it provides an efficient way to test effects and interactions of multiple factors (Montgomery 2013). In it, every factor is given an uppercase letter and every level is arbitrarily considered high or low. For our particular application scenario, as stated in section 5.7, we maintain the distinction of role data as a way to clearly differentiate the impact of other factors in accuracy, not to estimate the impact of the dataset itself. The reason for this is that both the results of the previous k-NN implementation and the results shown in section 5.8 provide plenty of evidence to support the notion that the supplier data is indeed better for the report classification than the mechanic data. This leads us then to consider four factors: A) Weight schemes, with term frequency as high level and TF-IDF as low; B) Pre-processing, with *language-blind* as high level and *language-oriented* as low; C) Structured data use, with "usage" as high level and "no usage" as low; and D) Feature selection technique (Subset) with 1000 terms subset as high level and no subset (use all terms) as low.

With each of the four experiments we can estimate the effects of each factor and the significance of these effects, in other words, how much variability in accuracy can be attributed to changes in a given factor (Montgomery 2013). For this (test significance) we run an analysis of variance (ANOVA) on an linear model that we assume factors follow at least in the range considered within their levels, something safe to do since the linear model can hold even if the assumption is very approximate (Montgomery 2013).

For each experiment, we mention the statistically significant factors as well as their estimated effects. We then take a look at the way these factors change depending on the accuracy cut-off considered. Additional support material for each experiment can be found in section 7.2 of the Appendix.

5.9.1 Supplier Data Experiments

For the accuracy cut-off at 1, with an adjusted r-square of 98.5% (the amount of variability that can be explained by the model, adjusted for the number of factors), we find that *pre-processing* and *subset* are the most significant factors, followed by *weight*. *Language-blind* pre-processing is estimated to augment accuracy by 6.87%, while selecting 1000 terms is estimated to reduce accuracy by 2.28%. Using term frequency weights is estimated to increase accuracy 0.63%, something that is already negligible.

If we look at Figure 55 Effect estimates with Supplier data for Accuracy at 1Figure 55 we can see there are some factor interactions that are also significant, however, their effects are so small that for practical purposes they can also be neglected. If we consider the kind of contributions our significant factors do, we can find that the best configuration for accuracy at 1 cut-off is that with *language-blind* pre-processing, using *all terms* with *term frequency* weights, regardless of the use of structured data. We can confirm this by looking at the corresponding results on Figure 49. Yet if we take the processing time into account, we see that for a net loss in accuracy of 1.6% (selecting the feature subset and using *term frequency* as weight scheme), we can classify every report in at least 632.73 times less time, going from 2 minutes 13 seconds to only 0.19 seconds.

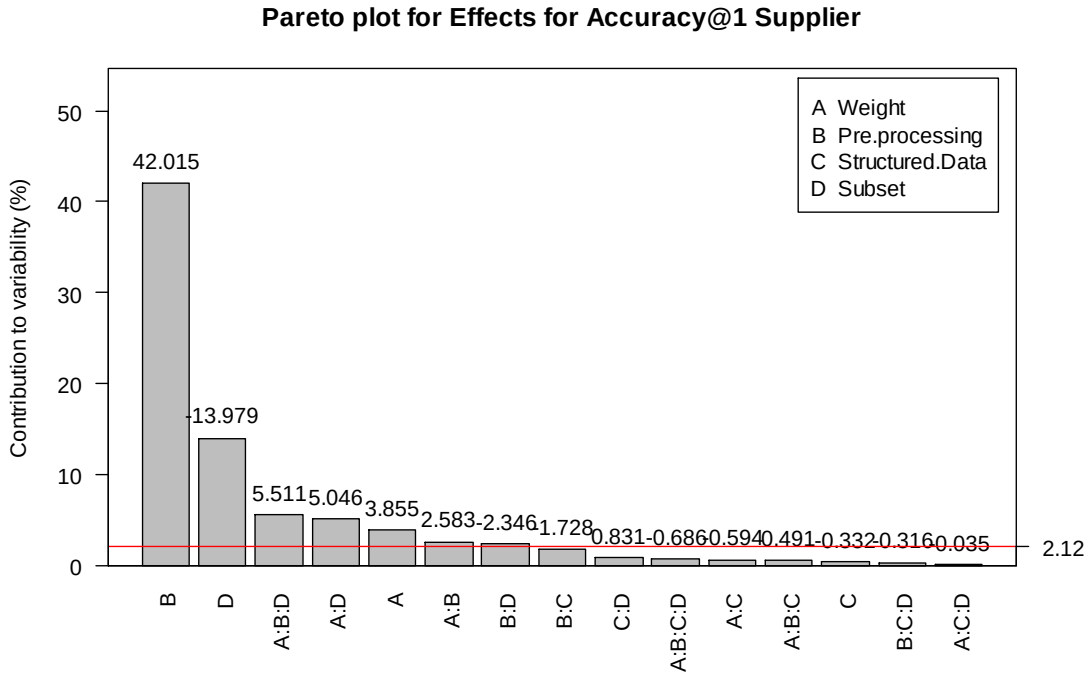


Figure 55 Effect estimates with Supplier data for Accuracy at 1

For the accuracy cut-off at 25, with an adjusted r-square of 96.72% (the amount of variability that can be explained by the model, adjusted for the number of factors), we find that *pre-processing* and *structured data* are strongly significant factors, *subset* is significant and *weight* is a little significant. *Language-blind* pre-processing is estimated to augment accuracy 3.58%. *Using structured data* is estimated to decrease accuracy by 1.03%. *Selecting a 1000 terms* is estimated to reduce accuracy by 0.46%, while using *term frequency* weights is estimated to increase accuracy only 0.22%, making the last two factors practically negligible.

Interactions between *pre-processing* and *structured data*, and *weight* and *subset*, while significant (as shown in Figure 56) have again negligible effects to be considered. Using the same graph to find the best configuration for the accuracy cut-off at 25, we conclude that a *language-blind* configuration, without using *structured data*, *selecting 1000 terms*, regardless of the weight scheme used should bring the best results. We can confirm this in Figure 49. Moreover, the processing time per report remains low in 0.17 seconds.

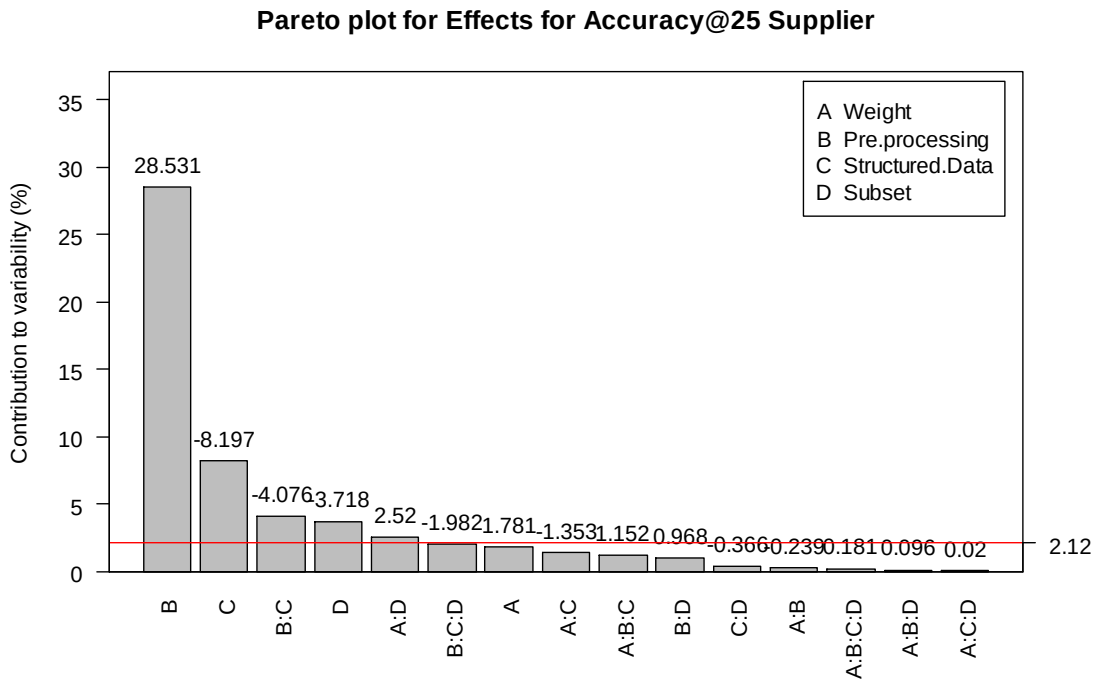


Figure 56 Effect estimates with Supplier data for Accuracy at 25

Examining Figure 57 and Figure 58 is possible to compare which effects are most relevant when trying to achieve higher accuracies with just 1 or 25 category suggestions. The first thing to notice is that in both cases, the most decisive factor is pre-processing, with a clear preference for the *language-blind* kind. In contrast, weighting schemes seem to have overall very little impact to improve accuracy, remaining steadily in favour of term frequency weights by a very little margin.

On the other end of the spectrum, the use of structured data and feature selection techniques have a very different effect depending on the accuracy we strive for. In traditional classification scenarios, where the objective is to obtain a single classification category per element (Figure 57), the effect of structured data is minimal and the reduced processing times achieved by using a subset of the total amount of features comes with a high price.

In our application scenario (Figure 58), the objective is different and so are the options to achieve it. The negative and now moderate effect of using a subset of all features can be easily compensated by properly configuring the pre-processing of text data, selecting the right weight scheme and including useful structured data. By doing this we can achieve a nearly optimal trade-off between accuracy and processing time. To further improve it, it would be necessary to refine the choices made in every factor that contributes to accomplish this trade-off, following the direction of the most useful level (as long as there are still options available). In this case, it means exploring configurations with even lighter pre-processing and looking for better ways to select the 1000 terms considered in the subset.

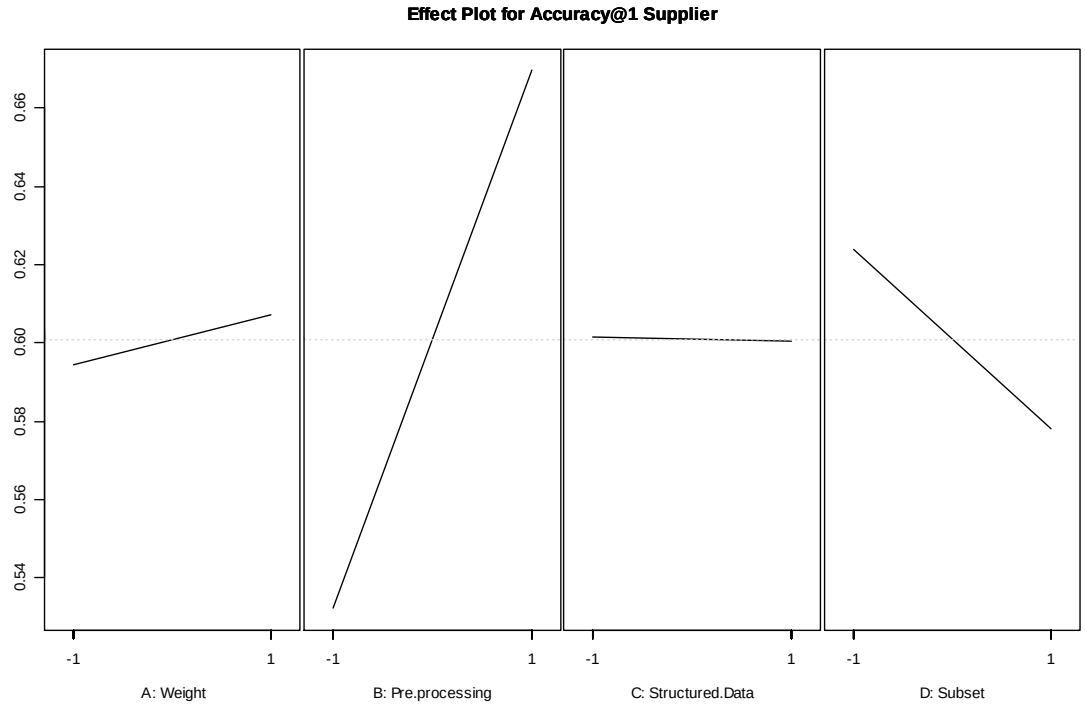


Figure 57 Variations in Accuracy levels at 1 by factor (Supplier data)

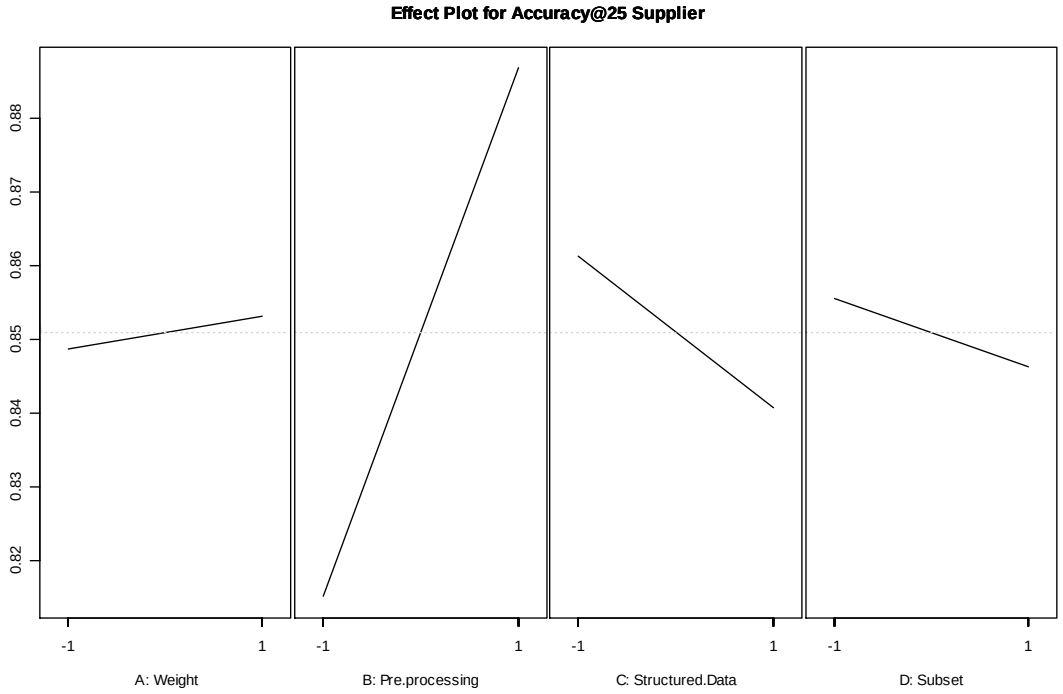


Figure 58 Variations in Accuracy levels at 25 by factor (Supplier data)

5.9.2 Mechanic Data Experiments

For the accuracy cut-off at 1, with an adjusted r-square of 99.2%, we find that all factors (weight, pre-processing, structured data and subset) are strongly significant. *Language-blind* pre-processing is estimated to augment accuracy by 8.61%. *Using structured data* is estimated to increase accuracy by 3.11%. *Selecting 1000 terms* is estimated to reduce accuracy by 1.26%, and using *term frequency* weights is estimated to augment accuracy by 0.63%, once more a negligible gain. Once more,

interactions among factors, although significant, are discard on the basis of their negligible contributions.

According to Figure 59, the best configuration for the accuracy cut-off at 1 uses *language-blind* pre-processing, *structured data*, and *all terms* regardless of the weighting scheme. This can be confirmed in Figure 52. Although, similar to the supplier case, if we consider the processing time it is better to use *term frequency* as the weighting scheme and to select 1000 terms, achieving a net loss in accuracy of 0.63%, and reducing the processing time per report from 2 minutes 4 seconds to only 0.14 seconds.

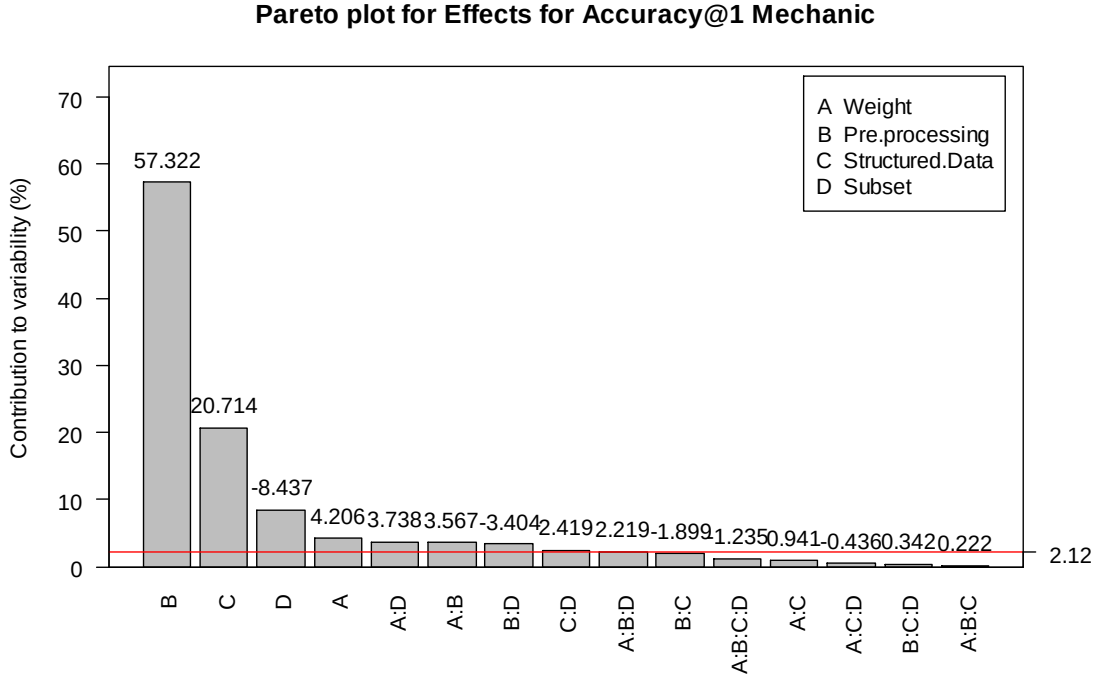


Figure 59 Effect estimates with Mechanic data for Accuracy at 1

For the accuracy cut-off at 25, with and adjusted r-square of 99.49%, we find that all factors are strongly significant. *Language-blind* pre-processing is estimated to augment accuracy by 10%, *using structured data* is estimated to augment accuracy by 1.97%, using *term frequency* weights does the same by 1.43% and selecting a *feature subset* of 1000 terms reduces accuracy by 1.32%.

In addition, all interactions except that involving all factors are significant to various degrees. Considering only those with non-negligible contributions, the interaction *pre-processing-structured data* is estimated to decrease accuracy by 2.21%, the interaction *weight-pre-processing* is estimated to increase accuracy by 1.21%, the interaction *weight-subset* is estimated to increase accuracy by 1.12%, the interaction *pre-processing-subset* is estimated to decrease accuracy by 1.38%, and the interaction *weight-pre-processing-subset* is estimated to increase accuracy by 1.39%.

Based on Figure 60, the best configuration for the accuracy cut-off at 25 (considering the effects of factors and interactions) uses *language-blind* pre-processing, no *structured data*, and *all terms* with *term frequency* weights. This can be confirmed in Figure 52. Once more, however, by taking a loss of 1.32% in accuracy at this cut-off level, the processing time per report is reduced 857.85 times, passing from 2 minutes 6 seconds to 0.14 seconds.

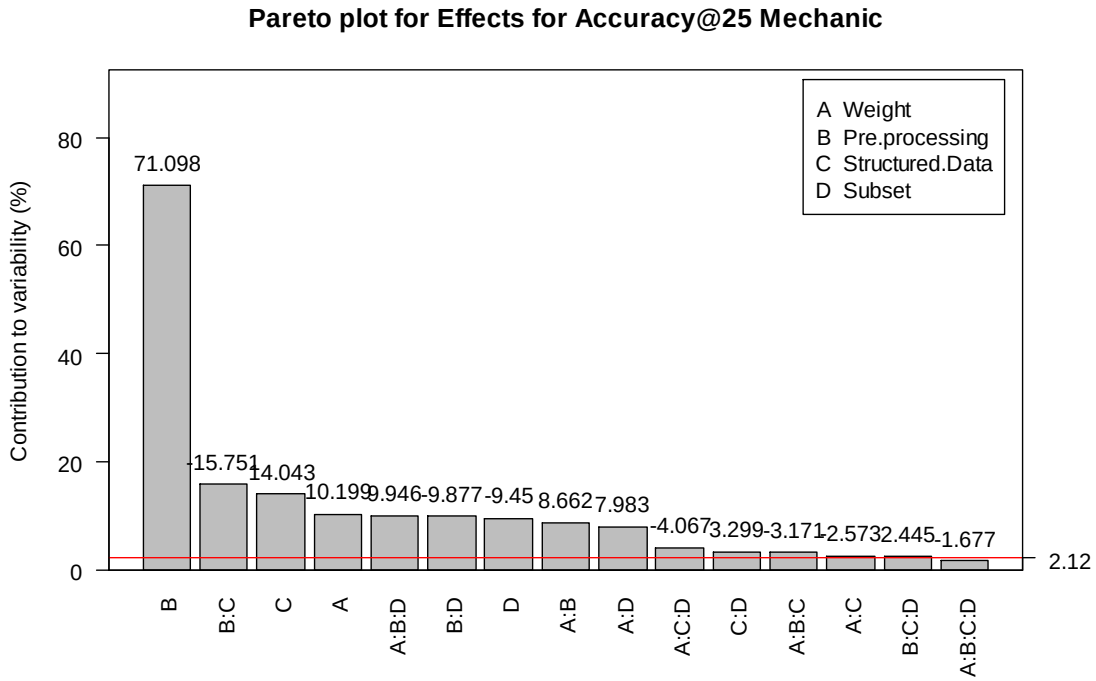


Figure 60 Effect estimates with Mechanic data for Accuracy at 25

With Figure 61 and Figure 62 it is possible to understand how the effect that each factor has on accuracy changes depending on the cut-off we consider. Contrary to the situation with the supplier dataset, effects remain more or less the same at 1 and at 25, letting *pre-processing* prevail as the dominant factor, and the *language-blind* type as the best choice. This confirms the finding in the supplier experiments of aiming to find lighter pre-processing approaches that nonetheless improve the effectiveness of feature selection techniques.

Aside from this, it is interesting to note that for the mechanic dataset the use of structured data does increase accuracy at both cut-off levels, whereas for the supplier dataset, it is either irrelevant or counterproductive. Determining the reason for this inverse behaviour is beyond the scope of this work.

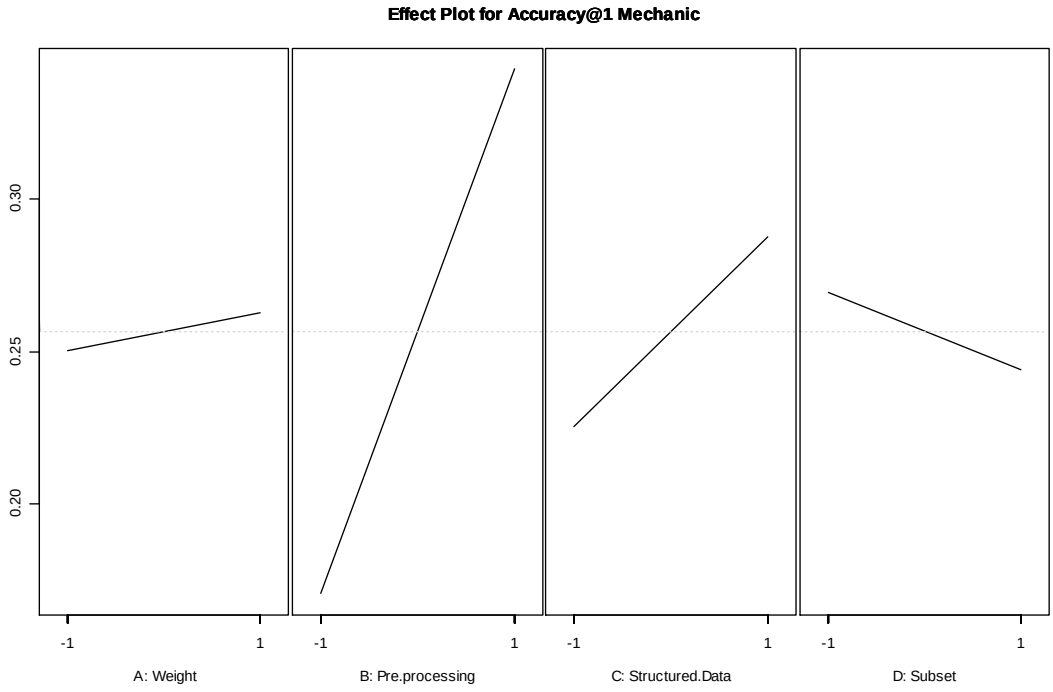


Figure 61 Variations in Accuracy levels at 1 by factor (Mechanic data)

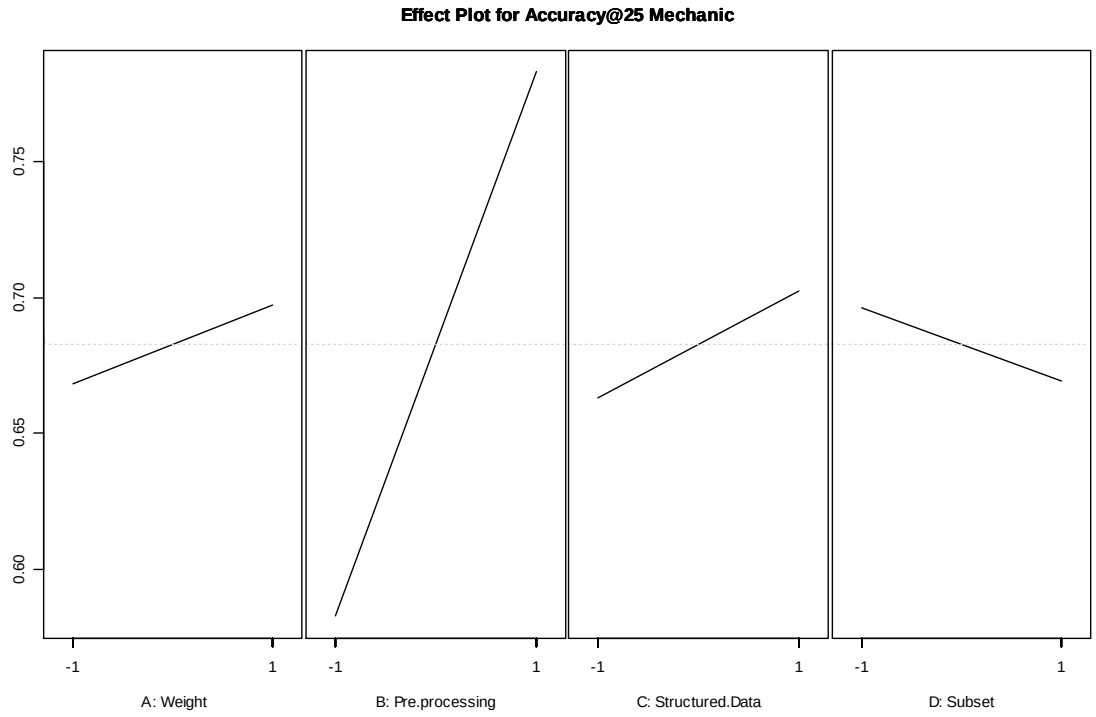


Figure 62 Variations in Accuracy levels at 25 by factor (Mechanic data)

5.9.3 Selection of the Classification Algorithm Configuration and Features

Using only the information obtained by applying our framework to optimise data features and classification algorithms (see chapter 4), it is possible to select the algorithm configuration and features that attain the best performance for our application scenario.

From the performance results shown in section 5.8, it is clear that the supplier dataset contains higher quality features than the mechanic dataset, thus reducing the selection of algorithm configurations to those using the former. As for the corresponding experiments (see sub section 5.9.1), each one suggested slightly different configurations, encouraging the use of all terms on one side and suggesting the same for the 1000 terms subset on the other. One experiment disregards the specific choice of structured data while the other does something similar with the weighting schemes.

However, when considering processing time (excluded by design from the experiment, since there cannot be two response variables for the same model), there is only one configuration that both experiments recommend. It is that with *language-blind* pre-processing, no use of structured data and the feature subset with term frequency weights. This configuration achieves 90.8% of accuracy with 25 suggestions and processes each report in 0.17 seconds.

5.10 Artefacts Evaluation

In terms of the Design Science methodology, the contributions of this research work comprise three artefacts: 1) the Conceptual Architecture for Text Classification (model artefact), 2) the optimisation method for classification algorithm configurations and features (method artefact), and 3) the Naïve Bayes-based classification configurations (instantiation artefact). In this section we evaluate each one of them to assess their utility, quality, and efficacy (Hevner et al. 2004).

5.10.1 Instantiation Artefact Evaluation

The collection of classification algorithm configurations can be evaluated from three different perspectives: 1) In relation to its performance as a classification solution (dynamic analysis), 2) in

comparison to other classification solutions (optimisation), 3) as a tool to identify the way factors and features affect accuracy (controlled experiment).

As a classification solution, the instantiation is subject to the metrics defined in sub section 1.2.4. At its best, the instantiation can provide a relevant error code within a list of 25 elements to a human expert in a little more than 90% of the cases, spending 0.17 seconds in the process (see sub section 5.9.3). Although there surely is room for improvement in terms of accuracy, we can consider the artefact useful enough to satisfy the needs of the application scenario.

When compared to similar solutions, the criterion used is that of optimisation. In other words, to verify if the artefact can provide a better solution than previous instantiations. If we compare the current instantiation with the corresponding (*bag of words*) configurations of the previous k-NN implementation by (Kassner & Mitschang 2016), we notice a combination of results. Using the mechanic dataset, our instantiation performs overall better with an accuracy cut-off at 1, however it underperforms with any higher accuracy cut-off. Results on the supplier dataset are no better. This time the instantiation underperforms the bag-of-words configurations of the k-NN implementation in all accuracy cut-offs. However, the trend reverses when considering processing time. The instantiation's configurations using a feature subset of 1000 terms have processing times ranging from 0.12 to 0.19 seconds per report, overcoming the k-NN implementation's reference values of 0.5 seconds per report and 0.3 seconds per report for configurations with stop word removal.

Finally, as a tool to test the effects of factors and features (in combination with a 2k experimental design), the artefact's utility derives from its pipeline design that favours modularity. This modularity, represented by the composing scripts, enables the replacement of one file for another that performs the same processing steps (e.g. pre-processing, document term matrix building), although with a slightly different logic (e.g. different weight schemes, other pre-processing components). Thanks to this, it is possible to run similar configurations based on the same data processing scripts, speeding up the process of creating very similar configuration that vary just in one design choice.

5.10.2 Model Artefact Evaluation

The first evidence of the ability of the Conceptual Architecture to help design solutions for the particular problem of our application scenario is the existence of the instantiation artefact and its ability to classify documents in an accurate and time efficient manner. As such, the instantiation provides what (Hevner et al. 2004) call *proof by construction* about the utility of the conceptual architecture.

We can also evaluate the completeness of this artefact by examining the way the architecture addresses each of the variables considered in our research model, which is at the core of the solutions we design for our problem.

The most straightforward relationship happens between the architecture's three layers and the moderating variables for *Feature Extraction*, *Feature Selection* and *Classification Algorithm*. The components in each layer offer alternatives to explore a wide range of pre-processing approaches, feature selection strategies, and algorithm configurations so that the impact of each of these variables can be tracked across their full range of values, other things held equal.

The *availability of structured data* is also acknowledged by the conceptual architecture in that it integrates structured data sources as a complement to extract features or to select them. Examples of components making use of this integration could be the concept recognition component or the dimension based selection techniques.

When it comes to the independent variables *quantity of data features* and *quality of data features*, although they are not depicted as elements of the architecture, it is clear that they are closely related to the selection of components in the feature extraction and feature selection layers. Components like spellchecking or punctuation removal are included based on the need to increase *feature quality* as part of the process of designing a solution. Something similar occurs between filters in the feature selection layers and *feature quantity*, albeit with a different focus based on the way feature quantity affects accuracy and processing time (see chapter 6).

Finally, the associations presented so far make it clear the conceptual architecture's orientation towards the improvement of the dependent variables *classification algorithm accuracy* and *classification algorithm processing time*. Without it, the purpose and utility of the architecture cannot be understood. This orientation can be demonstrated by the existence of several redundant components (weighting schemes, evaluation metrics, n-gram extraction or tokenisation) in every layer that perform the same kind of processing step, but whose presence is meant to provide flexibility in their use. This redundancy allows the construction of a configuration that can overall optimise for the target dependent variables by choosing complementary components that compensate the affectations of other components towards accuracy in favour of benefiting processing time, and vice versa.

5.10.3 Method Artefact Evaluation

Analogous to the case of the conceptual architecture, a compelling argument about the method's utility to guide the building process of a classification solution is the fact that it already did. The evidence for this *proof by construction* is shown in sections 5.1 to 5.9.

In addition to this, it is possible to imagine some additional application scenarios where the application of this method (along with the conceptual architecture) can also prove useful without any modification. We refer to scenarios where a business process is designed around the classification of text data, with text containing domain-specific vocabulary.

The first one is an insurance policy management process where an expert analyst needs to read customers' change requests to the terms of their contracted policy, which can lead to changing the policy's amount coverage, the scope of assets it protects, or even the insurance company (in the scenario of an insurance broker). One can think of simple "approved" or "rejected" categories that classify each customer request based on the contents of the text. More refined alternatives can include more specific categories detailing instead the degree of attention the request needs, so as to bring the most delicate cases to the attention of the expert analyst instead of letting him or her find them among all requests. While this may require more effort in labelling enough documents to train for all potential categories, the method is expected to handle this multi-class classification task as well, since it does not depend on the amount of categories to provide useful results. Moreover, structured data like the value of the assets covered by the policy, the customer's financial standing and the amount of years he or she has been renovating the policy are good examples of additional features that could be used in combination with text features.

Another scenario can be identified in a support desk process with phone conversation transcripts as input. In it, the text can be commented by a call centre executive to specify (using domain-specific jargon) the problem and symptoms discussed in the conversation. These two pieces of text would constitute a data bundle which can then be classified according to a priority scale so that the case (or support ticket) be assigned to one of different escalation support levels. The training data would need labels concerning the different priority levels, which depending on the company's policies may involve several categories. Complementary structured data that can be used in this scenario would be customer details retrieved from a CRM system using the customer's number, which is typically registered in a support phone call as part of the conversation protocol.

6 Conclusions and Future Research

At the beginning of chapter 2 we introduced a research model (see Figure 4) to explore the creation of classification solutions that could address the problem of our application scenario (see sub section 1.2.2). Given that our goal (see sub section 1.2.3) is to understand the way elements in the research model relate to one another, we present in this chapter the conclusions we can draw about these relationships based on the results and evaluation presented in sections 5.8 and 5.9. Additional remarks concerning how to continue studying any of these relationships are presented right after the relevant relationship.

Quality of data features can be perceived through visual and statistical exploration as shown throughout section 4.2. We find that a good summary for these data exploration, and as a consequence a good measure for feature quality are the language statistics suggested by (Bank et al. 2012) (e.g. vocabulary concentration, vocabulary dispersion, vocabulary relative size, and entropy) as well as others, like correlation. As expected, the relation of feature quality to classification performance is directly proportional: the higher the quality of the data features, the better the classification performance.

Moving into *quantity of data features*, we find this independent variable can be measured in terms of the number of distinct terms (vocabulary size) or in terms of the occurrences each term has. This leads to two corresponding measures of dataset coverage. However, as our feature selection technique shows, it is better to opt for the occurrence based coverage, as it allows to subset the amount of features with consideration to the power-law distribution their frequencies show. This proves to be of key importance to remove highly correlated terms that can lower classification performance. This also infers that the relation of data quantity with classification performance is not a linear one. It does not have to do only with providing the least amount of terms that account for the most amount of occurrences, but also with choosing a subset with this characteristics that also deals with correlation. In other words, objective is to have a middle point, with not so few features that they do not describe the dataset correctly (thus degrading accuracy) but also with not that many that it increases processing time.

Concerning *feature extraction* mechanisms, the pre-processing approaches proved to be the single most significant factor to alter accuracy, with minimal affectations to performance. Supporting evidence are the similar times between *language-blind* and *language-oriented* configurations shown in section 5.8. The difference in their effects seems to dictate that less complex pre-processing leads to better accuracy with marginally longer processing times.

It is important to remember that each of these approaches is an aggregation of several components. Therefore changes in accuracy attributed to either approach tell us more about the components that differentiate each approach than about the common components (e.g. stop word removal, lowercasing). As a result, it is worth pursuing further analysis of the effect each separate component has or even better, ways to estimate this without performing all the required classifications.

The effects of *Feature selection* can be controlled by using different techniques, such as the statistical sub setting described in section 5.5. In general this moderating variable tends to degrade accuracy but sharply increase performance, thus highlighting the need to use it appropriately and in combination with feature extraction and feature selection components that compensate for this and positively impact overall classification performance.

Another aspect of feature selection are (the confirmation of) weight schemes. These however showed little impact on accuracy, at least when it comes to the two schemes tested. Because of this, the choice of term frequency weights (the better choice in terms of accuracy) mostly plays a compensating function to handle the accuracy drop caused by selecting a sub set of all available features.

The *classification algorithm* as a moderating variable seems to set the limits for the effects of the other moderating variables, the upper limits in particular. For example, not even the best configuration of the Naïve Bayes algorithm shown in section 5.9.3, can surpass the equivalent configurations (*bag-of-words*) from the k-NN algorithm. Further exploration of the Naïve Bayes variant with similar pre-processing choices to those of the k-NN implementation could confirm this assumption.

When looking at the impact of using *available structured data* on the algorithm performance, we do see an effect, however results do not provide a clear direction for it, as in combination with supplier data it

tends to degrade accuracy, while the opposite occurs when the mechanic data is used. Possible causes for this can be related to 1) the dataset used, 2) the way the algorithm integrates structured data into calculations, or 3) the nature of the structured data itself. Concerning the dataset employed, the cause could either lie on the quality or quantity of the dataset in question, since we know the mechanic dataset is both smaller and of lesser quality. This view is the one supported by our results. However it could also be that the impact of structured data heavily depends on the way an algorithm ponders different kinds of features into its calculations, for example giving more weight to features that have more importance in the context of the application scenario. Finally, there is also the possibility that the structured data features used in this work were not of sufficient quality to really impact classification performance. Finding mechanisms to assess the quality of structured data features, or augmenting the share of features coming from structured data could shed light on the truth this hypothesis may hold.

Finally, while the *amount of categories* (our only confounding variable) is given as part of the requirements and conditions of our application scenario, it does represent an indirect limitation to increasing classification performance. This is because it restricts the classification algorithms that can be employed.

This work has brought attention to the way the different variables in our research model can be configured to design increasingly performing solutions. A key observation in this respect is that although the dataset used, the feature extraction and feature selection choices can all have sound effects on the final accuracy and processing time of a classification solution, they cannot overcome the performance range established by the classification algorithm in use. This stresses the need for future research on this topic to delve into the broad scope of possible solutions based on different classification algorithms. Suffice to mention one example of this future work:

The SVM algorithm can be adapted to multi-class classification problems, turning it into a set of binary classification tasks as numerous as there are categories to classify. This decomposition of the classification problem can in theory enable the algorithm to be run in a distributed computing network, where each node processes a classification category as a standard SVM algorithm (see sub section 3.3.4). Each node requires observations labelled for its category to serve as positive train data, whereas every other observation in the dataset can be used as negative train data. This split can be expected to lead to skewed training sets, which may need to be resolved as a feature selection problem, using the necessary elements from our Framework to Optimise Data Features and Classification Algorithms (comprising the Conceptual Architecture and Optimisation Method). While this implementation is certainly expensive in terms of computing power and possibly processing time, the potential accuracy gains make it an interesting research endeavour.

Additionally, as stated in the beginning of this document (see sub section 1.2.2), the focus when exploring the effects of the different elements of our research model has remained on the side of the *bag of words* approach. Yet, a complementing element to this research is the exploration of a *bag of concepts* approach that can provide information about a particular feature extraction component: the concept annotation. While the emphasis is this work has been to count existing words in every document to do calculations with them, a concept annotator identifies the relevant word or set of words that represent a concept in this particular domain and replaces them with a concept identifier. This implies a representation of text reports as a collection of identifiers. This significant transformation brings the advantage of discovering relationships previously hidden across languages and behind synonyms. Previously implemented resources (needed to implement this approach) are based on the UIMA framework (see sub section 3.4.3), which then sets the context for further work to be done in this direction.

In a nutshell, it is through the exploration of alternative classification algorithms and additional feature extraction and feature selection components that the work of quality experts in our application scenario can be better supported. The aim in all cases is to raise the existing 90% of accuracy with 25 suggested categories to higher ratios needing less suggestions and the same or shorter execution times. The better this conditions are satisfied, the more useful and transparent our technological solution becomes in the great scheme of things, along with the benefits for employees and customers this entails.

References

Bank, M., 2013. AIM - A Social Media Monitoring System for Quality Engineering. Available at: http://nbn-resolving.de/urn:nbn:de:bsz:15-qucosa-115894.

Bank, M. et al., 2012. Textual Characteristics for Language Engineering. In N. Calzolari et al., eds. *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul: European Language Resources Association (ELRA), pp. 515–519. Available at: http://www.lrec-conf.org/proceedings/lrec2012/pdf/182_Paper.pdf.

Blum, A.L. & Langley, P., 1997. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2), pp.245–271.

Cavnar, W.B., Trenkle, J.M. & Mi, A.A., 1994. N-Gram-Based Text Categorization. *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pp.161–175.

Chih-Wei Hsu, Chih-Chung Chang, and C.-J.L., 2008. A Practical Guide to Support Vector Classification. *BJU international*, 101(1), pp.1396–400. Available at: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

Clauset, A., Rohilla Shalizi, C. & Newman, M.E.J., 2009. Power-law Distributions in Empirical Data. *SIAM Review*, 51(4), pp.661–703.

Cooper, D. & Schindler, P.S., 2011. Thinking Like a Researcher. In *Business Research Methods*. McGraw-Hill/Irwin, pp. 52–77.

Damljanovic, D., Stankovic, M. & Laublet, P., 2012. Linked Data-Based Concept Recommendation : Comparison of Different Methods. In E. Simperl et al., eds. *9th Extended Semantic Web Conference, ESWC 2012*. Heraklion: Springer, pp. 24–38.

Dasgupta, a et al., 2007. Feature selection methods for text classification. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp.230–239. Available at: papers2://publication/uuid/666861F4-34A5-45D1-ABD9-50768335A4E1.

Feinerer, I., Hornik, K. & Meyer, D., 2008. Text Mining Infrastructure in R. *Journal Of Statistical Software*, 25(5), pp.1–54. Available at: http://www.jstatsoft.org/v25/i05.

Ferrucci, D. & Lally, A., 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4), pp.327–348.

Forman, G., 2003. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, 3, pp.1289–1305.

Freire, N., Borbinha, J. & Calado, P., 2012. An Approach for Named Entity Recognition in Poorly Structured Data. In E. Simperl et al., eds. *The Semantic Web: Research and Applications. 9th Extended Semantic Web Conference, ESWC 2012*. Heraklion: Springer, pp. 718–732.

Gandomi, A. & Haider, M., 2014. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2), pp.137–144. Available at: http://dx.doi.org/10.1016/j.ijinfomgt.2014.10.007.

Giorgetti, D. & Sebastiani, F., 2003. Multiclass Text Categorization for Automated Survey Coding. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*. ACM, pp. 798–802.

Gupta, A., 2011. *New Framework for Cross-Domain Document Classification*. Naval Postgraduate School.

Hall, M. et al., 2009. The WEKA data mining software. *ACM SIGKDD Explorations Newsletter*, 11(1), p.10. Available at: http://portal.acm.org/citation.cfm?doid=1656274.1656278.

Hänig, C., 2012. *Unsupervised Natural Language Processing for Knowledge Extraction from Domain-specific Textual Resources*. University of Leipzig.

Hänig, C., Bordag, S. & Quasthoff, U., 2008. Unsuparse: Unsupervised parsing with unsupervised part of speech tagging. In N. Calzolari et al., eds. *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. Marrakech: European Language Resources Association (ELRA), pp. 1109–1114. Available at: http://www.lrec-conf.org/proceedings/lrec2008/pdf/286_paper.pdf.

Hevner, a. R., March, S.T. & Park, J., 2004. Design Science in Information Systems Research. *MIS Quarterly*, 28(1), pp.75–105. Available at: http://dl.acm.org/citation.cfm?id=2017217.

Hofmann, M. & Chisholm, A., 2016. *Text Mining and Visualization: Case Studies Using Open-Source*

Tools, CRC Press. Available at: https://books.google.de/books?id=JfQYCwAAQBAJ.

Hornik, K. et al., 2013. The textcat Package for n-Gram Based Text Categorization in R. *Journal of Statistical Software*, 52(6), pp.1–17. Available at: http://www.jstatsoft.org/v52/i06.

Hotho, A., Nürnberger, A. & Paaß, G., 2005. A Brief Survey of Text Mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 20, pp.19–62. Available at: http://www.kde.cs.uni-kassel.de/hotho/pub/2005/hotho05TextMining.pdf.

Johnson, N.J., 1978. Modified t Tests and Confidence Intervals for Asymmetrical Populations. *Journal of the American Statistical Association*, 73(363), pp.536–544.

Jurka, T.P. et al., 2013. RTextTools: A Supervised Learning Package for Text Classification. *R Journal*, 5(1), pp.6–12. Available at: http://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=20734859&AN=90616103&h=50cyR6MarAY9WE/sMi2d4KY2rxKU7dzzMDzuuThqpFgz4Zxe3x3+d7WcPDSIcW+g0t7yaqx3AXpG8xNmnd35Jg==&crl=c.

Kassner, L. et al., 2014. Product Life Cycle Analytics – Next Generation Data Analytics on Structured and Unstructured Data. In *9th CIRP Conference on Intelligent Computation in Manufacturing Engineering*. pp. 35–40.

Kassner, L. & Mitschang, B., 2016. Exploring Text Classification for Messy Data : An Industry Use Case for Domain-Specific Analytics Industrial Paper Categories and Subject Descriptors. In *19th International Conference on Extending Database Technology (EDBT)*. Bourdeaux.

Khan, A. et al., 2010. A Review of Machine Learning Algorithms for Text-Documents Classification. *Journal of Advances in Information Technology*, 1(1), pp.4–20.

Kyriakopoulou, A., 2008. Text classification aided by clustering: a literature review. *Tools in Artificial Intelligence*, pp.233–252. Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Text+Classification+Aided+by+Clustering:+A+Literature+Review#0.

Lang, A., Ortiz, M.M. & Abraham, S., 2009. Enhancing Business Intelligence with unstructured data. In J.-C. Freytag et al., eds. *Datenbanksysteme in Business, Technologie und Web, BTW 2009 - 13th Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), Proceedings*. pp. 469–485.

Liu, W., Wang, L. & Yi, M., 2013. Power Law for Text Categorization. *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, 8208, pp.131–143. Available at: <Go to ISI>://WOS:000358605100013.

Montgomery, D.C., 2013. *Design and Analysis of Experiments* 8th ed., Tempe: John Wiley & Sons, Inc. Available at: http://cataleg.uab.cat/record=b1764873~S1*cat.

Murty, M.R. et al., 2012. A survey of cross-domain text categorization techniques. *2012 1st International Conference on Recent Advances in Information Technology, RAIT-2012*, pp.499–504.

Newman, M.E.J., 2005. Power laws, Pareto distributions and Zipf's law. *Power laws, Pareto distributions and Zipf's law. Contemporary physics*, 46(5), pp.323–351. Available at: http://arxiv.org/abs/cond-mat/0412004\nhttp://dx.doi.org/10.1016/j.cities.2012.03.001.

Ogren, P. V. & Bethard, S.J., 2009. Building test suites for UIMA components. *Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing (SETQA-NLP 2009)*, Proceeding(June), pp.1–4. Available at: http://dl.acm.org/citation.cfm?id=1621947.1621948.

Pulvermüller, F., 1999. Words in the brain's language. *Behavioral and Brain Sciences*, 22, pp.253–336. Available at: http://kops.uni-konstanz.de/handle/123456789/10734.

R Core Team, 2001. What is R? *R News*, 1(January), pp.2–3.

Rafeeque, P.C. & Sendhilkumar, S., 2011. A survey on Short text analysis in Web. In *2011 Third International Conference on Advanced Computing*. Chennai: IEEE, pp. 365–371. Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6165203.

Ruotsalo, T., 2012. Domain Specific Data Retrieval on the Semantic Web BT - The Semantic Web: Research and Applications. In E. Simperl et al., eds. *9th Extended Semantic Web Conference, ESWC 2012*. Heraklion: Springer, pp. 422–436. Available at: http://dx.doi.org/10.1007/978-3-642-30284-8_35\npapers2://publication/livfe/id/95124.

Schierle, M., 2011. *Language Engineering for Information Extraction*. Universität Leipzig.

Schierle, M. & Trabold, D., 2008. Multilingual knowledge based concept recognition in textual data. In *Proceedings of the 32nd Annual Conference of the GfKl*. pp. 1–10.

Sebastiani, F., 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), pp.1–47. Available at: http://portal.acm.org/citation.cfm?doid=505282.505283.

Turek, D., 2012. The Case Against Digital Sprawl. *Bloomberg Business*. Available at: http://www.bloomberg.com/news/articles/2012-05-02/the-case-against-digital-sprawl [Accessed March 23, 2016].

Turner, V. et al., 2014. *The Digital Universe of Opportunities: Rich Data and Increasing Value of the Internet of Things*, Framingham. Available at: http://www.emc.com/leadership/digital-universe/index.htm.

de Winter, J.C.., 2013. Using the Student's t -test with extremely small sample sizes. *Practical Assessment, Research & Evaluation*, 18(10), pp.1–12.

Zhang, X. et al., 2011. SVM based extraction of spatial relations in text. *ICSDM 2011 - Proceedings 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services*, pp.529–533.

7 Appendices

7.1 Function Words Lists for English and German

7.1.1 English Function Words List

1. a
2. about
3. above
4. after
5. again
6. ago
7. all
8. almost
9. along
10. already
11. also
12. although
13. always
14. am
15. among
16. an
17. and
18. another
19. any
20. anybody
21. anything
22. anywhere
23. are
24. aren't
25. around
26. as
27. at
28. back
29. else
30. be
31. been
32. before
33. being
34. below
35. beneath
36. beside
37. between
38. beyond
39. billion
40. billionth
41. both
42. each
43. but
44. by
45. can
46. can't
47. could
48. couldn't
49. did
50. didn't
51. do
52. does
53. doesn't
54. doing
55. done
56. don't
57. down
58. during
59. eight
60. eighteen
61. eighteenth
62. eighth
63. eightieth
64. eighty
65. either
66. eleven
67. eleventh
68. enough
69. even
70. ever
71. every
72. everybody
73. everyone
74. everything
75. everywhere
76. except
77. far
78. few
79. fewer
80. fifteen
81. fifteenth
82. fifth
83. fiftieth
84. fifty
85. first
86. five
87. for
88. fortieth
89. forty
90. four
91. fourteen
92. fourteenth
93. fourth
94. hundred
95. from
96. get
97. gets
98. getting
99. got

100. had
101. hadn't
102. has
103. hasn't
104. have
105. haven't
106. having
107. he
108. he'd
109. he'll
110. hence
111. her
112. here
113. hers
114. herself
115. he's
116. him
117. himself
118. his
119. hither
120. how
121. however
122. near
123. hundredth
124. i
125. i'd
126. if
127. i'll
128. i'm
129. in
130. into
131. is
132. i've
133. isn't
134. it
135. its
136. it's
137. itself
138. just
139. last
140. less
141. many
142. me
143. may
144. might
145. million
146. millionth
147. mine
148. more
149. most
150. much
151. must
152. mustn't
153. my
154. myself
155. near
156. nearby
157. nearly
158. neither
159. never
160. next
161. nine
162. nineteen
163. nineteenth
164. ninetieth
165. ninety
166. ninth
167. no
168. nobody
169. none
170. noone
171. nothing
172. nor
173. not
174. now
175. nowhere
176. of
177. off
178. often
179. on
180. or
181. once
182. one
183. only
184. other
185. others
186. ought
187. oughtn't
188. our
189. ours
190. ourselves
191. out
192. over
193. quite
194. rather
195. round
196. second
197. seven
198. seventeen
199. seventeenth
200. seventh
201. seventieth
202. seventy
203. shall
204. shan't
205. she'd
206. she
207. she'll
208. she's
209. should
210. shouldn't
211. since
212. six
213. sixteen

214. sixteenth
215. sixth
216. sixtieth
217. sixty
218. so
219. some
220. somebody
221. someone
222. something
223. sometimes
224. somewhere
225. soon
226. still
227. such
228. ten
229. tenth
230. than
231. that
232. that
233. that's
234. the
235. their
236. theirs
237. them
238. themselves
239. these
240. then
241. thence
242. there
243. therefore
244. they
245. they'd
246. they'll
247. they're
248. third
249. thirteen
250. thirteenth
251. thirtieth
252. thirty
253. this
254. thither
255. those
256. though
257. thousand
258. thousandth
259. three
260. thrice
261. through
262. thus
263. till
264. to
265. towards
266. today
267. tomorrow
268. too
269. twelfth
270. twelve
271. twentieth
272. twenty
273. twice
274. two
275. under
276. underneath
277. unless
278. until
279. up
280. us
281. very
282. when
283. was
284. wasn't
285. we
286. we'd
287. we'll
288. were
289. we're
290. weren't
291. we've
292. what
293. whence
294. where
295. whereas
296. which
297. while
298. whither
299. who
300. whom
301. whose
302. why
303. will
304. with
305. within
306. without
307. won't
308. would
309. wouldn't
310. yes
311. yesterday
312. yet
313. you
314. your
315. you'd
316. you'll
317. you're
318. yours
319. yourself
320. yourselves
321. you've

7.1.2 German Function Words List

1. als
2. am
3. an
4. auch
5. auf
6. aus
7. bei
8. bin
9. bis
10. bist
11. da
12. dann
13. darf
14. das
15. dein
16. dem
17. den
18. der
19. die
20. du
21. durch
22. ein
23. eine
24. einen
25. er
26. es
27. euch
28. euer
29. fragt
30. für
31. haben
32. hat
33. hinter
34. ich
35. ihr
36. im
37. in
38. ist
39. ja
40. kann
41. kein
42. los
43. mein
44. meine
45. mich
46. mir
47. mit
48. muss
49. nein
50. nicht
51. nun
52. nur
53. oder
54. oft
55. ruft
56. sagt
57. sein
58. sie
59. sind
60. so
61. soll
62. um
63. und
64. uns
65. unser
66. unten
67. von
68. vor
69. wann
70. war
71. was
72. wenn
73. wer
74. wie
75. will
76. wir
77. wo
78. zu
79. du
80. er
81. es
82. ich
83. man
84. sie
85. wir
86. an
87. auf
88. aus
89. durch
90. für
91. gegen
92. hinter
93. in
94. nach
95. neben
96. unter
97. vor
98. zu
99. über
100. aber
101. damit
102. ob
103. oder
104. und

105. weil
106. wenn
107. warum
108. was
109. wer
110. wie
111. wo
112. woher
113. wohin
114. dein
115. mein
116. unser
117. aber
118. bei
119. da
120. ein
121. für
122. ganz
123. alle
124. bis
125. dann
126. eine
127. gegen
128. als
129. das
130. einem
131. am
132. dass
133. einen
134. an
135. dem
136. einer
137. auch
138. den
139. eines
140. auf
141. denn
142. einzelnen
143. aus
144. der
145. er
146. des
147. es
148. die
149. diese
150. dieser
151. doch
152. du
153. durch
154. habe
155. ich
156. kann
157. man
158. nach
159. oder
160. haben
161. ihm
162. können
163. mehr
164. nicht
165. ohne
166. hat
167. ihn
168. meine
169. noch
170. hatte
171. ihnen
172. mich
173. nun
174. hier
175. ihr
176. mir
177. nur
178. ihre
179. mit
180. im
181. muss
182. in
183. ist
184. schon
185. über
186. vom
187. war
188. Zeit
189. sehr
190. um
191. von
192. was
193. zu
194. sein
195. und
196. vor
197. welche
198. zum
199. sein
200. uns
201. wenn
202. zur
203. seine
204. unter
205. werden
206. seiner
207. wie
208. selbst
209. wieder
210. sich
211. wir
212. sie
213. wird
214. sind

215. wo
216. so
217. zum
218. also
219. weiterhin
220. ergänzend
221. und
222. sicherlich
223. nochmals
224. wesentlich
225. ausdrücklich
226. nachdrücklich
227. letztlich
228. etwas
229. wenig
230. eher
231. kaum
232. fast
233. wenig
234. teilweise
235. ziemlich
236. nur
237. bloß
238. ziemlich
239. einigermaßen
240. viel
241. sehr
242. beträchtlich
243. ganz
244. höchst
245. völlig
246. gerade
247. ausschließlich
248. genug
249. überaus
250. sehr
251. gar
252. völlig
253. gänzlich
254. höchst
255. zu
256. allzu
257. übermäßig
258. über
259. generell
260. ausnahmsweise
261. offensichtlich
262. andererseits
263. jedoch
264. aber
265. wie
266. als
267. ebenso
268. häufig
269. oft
270. sehr
271. viel
272. weit
273. wenig
274. wohl
275. ähnlich
276. desgleichen
277. gleichfalls
278. einen
279. zwar
280. aber
281. gleichfalls
282. eine
283. beides
284. und
285. überdies
286. außerdem
287. dazu
288. insbesondere
289. erstens
290. zweitens
291. übrigens
292. auch
293. sogar
294. einschließlich
295. samt
296. nebst
297. inklusive
298. schließlich
299. sobald
300. als
301. während
302. bevor
303. bis
304. nachdem
305. darauf
306. dabei
307. zuvor
308. danach
309. währenddessen
310. daraufhin
311. unterdessen
312. damals
313. früher
314. zuvor
315. gleichzeitig
316. zuerst
317. zunächst
318. sodann
319. schließlich
320. endlich
321. später
322. seit
323. während
324. nach

325. vor
326. beispielsweise
327. ähnlich
328. deutlich
329. aber
330. sondern
331. doch
332. jedoch
333. während
334. währenddessen
335. indessen
336. dagegen
337. wohingegen
338. wogegen
339. dennoch
340. hingegen
341. vielmehr
342. jedoch
343. doch
344. zuwider
345. gegen
346. umgekehrt
347. obwohl
348. denn
349. weil
350. da
351. zumal
352. deswegen
353. dadurch
354. darum
355. weshalb
356. weswegen
357. also
358. eben
359. doch
360. nämlich
361. durch
362. dank
363. mangels
364. wegen
365. dass
366. somit
367. mithin
368. also
369. folglich
370. so
371. demzufolge
372. darum
373. daher
374. infolgedessen
375. infolge
376. deshalb
377. deswegen
378. aber
379. obgleich
380. wenngleich
381. obschon
382. obzwar
383. obwohl
384. trotz
385. allem
386. trotzdem
387. gleichwohl
388. doch
389. selbstverständlich
390. aber auch
391. natürlich
392. andererseits
393. sicherlich
394. trotzdem
395. allerdings
396. immerhin
397. zusammenfassend
398. zusammengefasst
399. kurz
400. kurzum
401. abschließend
402. schließlich
403. letztlich
404. schlussendlich
405. laut
406. entsprechend
407. offenbar
408. offensichtlich
409. ebenso
410. sowie
411. soweit
412. gleichsam
413. anscheinend
414. offenkundig
415. augenscheinlich
416. dies
417. bedenkend
418. voraussetzend

7.2 Design Matrices and Analysis Of Variance (ANOVA) for 2k Experiments

7.2.1 Design Matrix and ANOVA for Supplier Dataset with Accuracy At 1

A (Weight)	B (Pre-processing)	C (Structured Data)	D (Subset)	Treatment combination	Replicate I	Replicate II
+	+	+	+	1000 Terms, Term Frequency, Language-blind, With Structured Data	67,03%	66,35%
+	+	-	+	1000 Terms, Term Frequency, Language-blind, No Structured Data	67,57%	67,36%
+	-	+	+	1000 Terms, Term Frequency, Language-Oriented, With Structured Data	51,99%	51,58%
+	-	-	+	1000 Terms, Term Frequency, Language-Oriented, No Structured Data	52,48%	49,72%
-	+	+	+	1000 Terms, TF-IDF, Language-blind, With Structured Data	61,50%	61,29%
-	+	-	+	1000 Terms, TF-IDF, Language-blind, No Structured Data	61,77%	61,50%
-	-	+	+	1000 Terms, TF-IDF, Language-Oriented, With Structured Data	53,05%	50,28%
-	-	-	+	1000 Terms, TF-IDF, Language-Oriented, No Structured Data	51,99%	49,39%
+	+	+	-	All terms, Term Frequency, Language-blind, With Structured Data	68,37%	68,91%
+	+	-	-	All terms, Term Frequency, Language-blind, No Structured Data	68,71%	69,86%
+	-	+	-	All terms, Term Frequency, Language-Oriented, With Structured Data	55,16%	55,16%
+	-	-	-	All terms, Term Frequency, Language-Oriented, No Structured Data	55,32%	55,97%
-	+	+	-	All terms, TF-IDF, Language-blind, With Structured Data	70,26%	69,32%
-	+	-	-	All terms, TF-IDF, Language-blind, No Structured Data	71,00%	70,67%
-	-	+	-	All terms, TF-IDF, Language-Oriented, With Structured Data	54,75%	55,56%
-	-	-	-	All terms, TF-IDF, Language-Oriented, No Structured Data	55,00%	54,02%

Figure 63 2^k design matrix with supplier data for accuracy at 1

```
Call:
lm(formula = yield.lief.1 ~ A * B * C * D, data = fdacc.lief.1)

Residuals:
     Min       1Q   Median       3Q      Max
-0.01381 -0.00328  0.00000  0.00328  0.01381

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.009e-01  1.637e-03 367.141  < 2e-16 ***
A            6.309e-03  1.637e-03   3.855 0.001402 **
B            6.877e-02  1.637e-03  42.015  < 2e-16 ***
C           -5.434e-04  1.637e-03  -0.332 0.744190
D           -2.288e-02  1.637e-03 -13.979 2.19e-10 ***
A:B          4.227e-03  1.637e-03   2.583 0.020030 *
A:C         -9.728e-04  1.637e-03  -0.594 0.560585
B:C         -2.828e-03  1.637e-03  -1.728 0.103248
A:D          8.259e-03  1.637e-03   5.046 0.000119 ***
B:D         -3.840e-03  1.637e-03  -2.346 0.032177 *
C:D          1.361e-03  1.637e-03   0.831 0.417998
A:B:C        8.042e-04  1.637e-03   0.491 0.629844
A:B:D        9.020e-03  1.637e-03   5.511 4.74e-05 ***
A:C:D       -5.691e-05  1.637e-03  -0.035 0.972690
B:C:D       -5.178e-04  1.637e-03  -0.316 0.755800
A:B:C:D     -1.123e-03  1.637e-03  -0.686 0.502407
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.009259 on 16 degrees of freedom
```

```
Multiple R-squared:  0.9923,  Adjusted R-squared:  0.985
F-statistic: 136.6 on 15 and 16 DF,  p-value: 5.522e-14
```

7.2.2 Design Matrix and ANOVA for Supplier Dataset with Accuracy At 25

A (Weight)	B (Pre-processing)	C (Structured Data)	D (Subset)	Treatment combination	Replicate I	Replicate II
+	+	+	+	1000 Terms, Term Frequency, Language-blind, With Structured Data	87,39%	86,65%
+	+	-	+	1000 Terms, Term Frequency, Language-blind, No Structured Data	90,76%	90,63%
+	-	+	+	1000 Terms, Term Frequency, Language-Oriented, With Structured Data	81,88%	79,77%
+	-	-	+	1000 Terms, Term Frequency, Language-Oriented, No Structured Data	82,21%	82,05%
-	+	+	+	1000 Terms, TF-IDF, Language-blind, With Structured Data	85,97%	85,97%
-	+	-	+	1000 Terms, TF-IDF, Language-blind, No Structured Data	90,63%	88,67%
-	-	+	+	1000 Terms, TF-IDF, Language-Oriented, With Structured Data	80,99%	79,77%
-	-	-	+	1000 Terms, TF-IDF, Language-Oriented, No Structured Data	80,26%	80,42%
+	+	+	-	All terms, Term Frequency, Language-blind, With Structured Data	86,92%	88,27%
+	+	-	-	All terms, Term Frequency, Language-blind, No Structured Data	89,89%	90,49%
+	-	+	-	All terms, Term Frequency, Language-Oriented, With Structured Data	80,83%	81,23%
+	-	-	-	All terms, Term Frequency, Language-Oriented, No Structured Data	83,43%	82,70%
-	+	+	-	All terms, TF-IDF, Language-blind, With Structured Data	88,47%	87,46%
-	+	-	-	All terms, TF-IDF, Language-blind, No Structured Data	90,42%	90,29%
-	-	+	-	All terms, TF-IDF, Language-Oriented, With Structured Data	82,37%	81,07%
-	-	-	-	All terms, TF-IDF, Language-Oriented, No Structured Data	82,37%	82,78%

Figure 64 2^k design matrix with supplier data for accuracy at 25

```
Call:
lm(formula = yield.lief.25 ~ A * B * C * D, data = fdacc.lief.25)

Residuals:
     Min       1Q   Median       3Q      Max
-0.01056 -0.00319  0.00000  0.00319  0.01056

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  8.509e-01  1.257e-03 677.103  < 2e-16 ***
A            2.239e-03  1.257e-03   1.781 0.093856 .
B            3.586e-02  1.257e-03  28.531 3.78e-15 ***
C           -1.030e-02  1.257e-03  -8.197 4.04e-07 ***
D           -4.673e-03  1.257e-03  -3.718 0.001870 **
A:B         -3.000e-04  1.257e-03  -0.239 0.814372
A:C         -1.700e-03  1.257e-03  -1.353 0.194870
B:C         -5.123e-03  1.257e-03  -4.076 0.000879 ***
A:D          3.167e-03  1.257e-03   2.520 0.022747 *
B:D          1.217e-03  1.257e-03   0.968 0.347327
C:D         -4.596e-04  1.257e-03  -0.366 0.719363
A:B:C        1.447e-03  1.257e-03   1.152 0.266332
A:B:D        1.205e-04  1.257e-03   0.096 0.924817
A:C:D        2.488e-05  1.257e-03   0.020 0.984448
B:C:D       -2.490e-03  1.257e-03  -1.982 0.064963 .
A:B:C:D      2.280e-04  1.257e-03   0.181 0.858329
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 0.007109 on 16 degrees of freedom
Multiple R-squared:  0.9831,  Adjusted R-squared:  0.9672
F-statistic: 61.96 on 15 and 16 DF,  p-value: 2.707e-11
```

7.2.3 Design Matrix and ANOVA for Mechanic Dataset with Accuracy At 1

A (Weight)	B (Pre-processing)	C (Structured Data)	D (Subset)	Treatment combination	Replicate I	Replicate II
+	+	+	+	1000 Terms, Term Frequency, Language-blind, With Structured Data	38,17%	37,29%
+	+	-	+	1000 Terms, Term Frequency, Language-blind, No Structured Data	30,92%	31,88%
+	-	+	+	1000 Terms, Term Frequency, Language-Oriented, With Structured Data	20,94%	20,17%
+	-	-	+	1000 Terms, Term Frequency, Language-Oriented, No Structured Data	12,39%	12,97%
-	+	+	+	1000 Terms, TF-IDF, Language-blind, With Structured Data	33,54%	33,97%
-	+	-	+	1000 Terms, TF-IDF, Language-blind, No Structured Data	27,51%	26,72%
-	-	+	+	1000 Terms, TF-IDF, Language-Oriented, With Structured Data	21,23%	17,68%
-	-	-	+	1000 Terms, TF-IDF, Language-Oriented, No Structured Data	13,06%	11,91%
+	+	+	-	All terms, Term Frequency, Language-blind, With Structured Data	39,13%	39,21%
+	+	-	-	All terms, Term Frequency, Language-blind, No Structured Data	33,45%	33,54%
+	-	+	-	All terms, Term Frequency, Language-Oriented, With Structured Data	21,33%	20,17%
+	-	-	-	All terms, Term Frequency, Language-Oriented, No Structured Data	14,70%	14,51%
-	+	+	-	All terms, TF-IDF, Language-blind, With Structured Data	38,60%	36,94%
-	+	-	-	All terms, TF-IDF, Language-blind, No Structured Data	33,54%	34,06%
-	-	+	-	All terms, TF-IDF, Language-Oriented, With Structured Data	21,42%	20,65%
-	-	-	-	All terms, TF-IDF, Language-Oriented, No Structured Data	15,27%	14,41%

Figure 65 2k design matrix with mechanic data for accuracy at 1

```
Call:
lm(formula = yield.mont.1 ~ A * B * C * D, data = fdacc.mont.1)

Residuals:
      Min        1Q    Median        3Q       Max
-0.017771 -0.003864  0.000000  0.003864  0.017771

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.2566519  0.0015028 170.783  < 2e-16 ***
A            0.0063209  0.0015028   4.206  0.00067 ***
B            0.0861428  0.0015028  57.322  < 2e-16 ***
C            0.0311284  0.0015028  20.714 5.57e-13 ***
D           -0.0126798  0.0015028  -8.437 2.76e-07 ***
A:B          0.0053603  0.0015028   3.567  0.00257 **
A:C          0.0014137  0.0015028   0.941  0.36083
B:C         -0.0028533  0.0015028  -1.899  0.07579 .
A:D          0.0056167  0.0015028   3.738  0.00179 **
B:D         -0.0051150  0.0015028  -3.404  0.00363 **
C:D          0.0036352  0.0015028   2.419  0.02785 *
A:B:C        0.0003330  0.0015028   0.222  0.82743
A:B:D        0.0033353  0.0015028   2.219  0.04126 *
A:C:D       -0.0006551  0.0015028  -0.436  0.66873
```

```
B:C:D        0.0005132  0.0015028   0.342  0.73715
A:B:C:D     -0.0018558  0.0015028  -1.235  0.23469
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.008501 on 16 degrees of freedom
Multiple R-squared:  0.9959,   Adjusted R-squared:  0.992
F-statistic: 257.3 on 15 and 16 DF,  p-value: 3.66e-16
```

7.2.4 Design Matrix and ANOVA for Mechanic Dataset with Accuracy At 25

A (Weight)	B (Pre-processing)	C (Structured Data)	D (Subset)	Treatment combination	Replicate I	Replicate II
+	+	+	+	1000 Terms, Term Frequency, Language-blind, With Structured Data	79,65%	79,74%
+	+	-	+	1000 Terms, Term Frequency, Language-blind, No Structured Data	81,75%	81,83%
+	-	+	+	1000 Terms, Term Frequency, Language-Oriented, With Structured Data	61,96%	62,73%
+	-	-	+	1000 Terms, Term Frequency, Language-Oriented, No Structured Data	54,37%	54,08%
-	+	+	+	1000 Terms, TF-IDF, Language-blind, With Structured Data	72,05%	73,10%
-	+	-	+	1000 Terms, TF-IDF, Language-blind, No Structured Data	68,73%	67,69%
-	-	+	+	1000 Terms, TF-IDF, Language-Oriented, With Structured Data	63,02%	62,92%
-	-	-	+	1000 Terms, TF-IDF, Language-Oriented, No Structured Data	55,04%	52,64%
+	+	+	-	All terms, Term Frequency, Language-blind, With Structured Data	80,09%	80,09%
+	+	-	-	All terms, Term Frequency, Language-blind, No Structured Data	82,97%	81,40%
+	-	+	-	All terms, Term Frequency, Language-Oriented, With Structured Data	63,78%	62,63%
+	-	-	-	All terms, Term Frequency, Language-Oriented, No Structured Data	54,85%	53,60%
-	+	+	-	All terms, TF-IDF, Language-blind, With Structured Data	79,39%	80,26%
-	+	-	-	All terms, TF-IDF, Language-blind, No Structured Data	81,75%	82,10%
-	-	+	-	All terms, TF-IDF, Language-Oriented, With Structured Data	60,61%	62,15%
-	-	-	-	All terms, TF-IDF, Language-Oriented, No Structured Data	53,22%	54,95%

Figure 66 2k design matrix with mechanic data for accuracy at 25

```
Call:
lm(formula = yield.mont.25 ~ A * B * C * D, data = fdacc.mont.25)

Residuals:
      Min        1Q    Median        3Q       Max
-0.012008 -0.004585  0.000000  0.004585  0.012008

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.682857   0.001407 485.481  < 2e-16 ***
A            0.014345   0.001407  10.199 2.09e-08 ***
B            0.100004   0.001407  71.098  < 2e-16 ***
C            0.019753   0.001407  14.043 2.04e-10 ***
D           -0.013292   0.001407  -9.450 6.00e-08 ***
A:B          0.012183   0.001407   8.662 1.95e-07 ***
A:C         -0.003619   0.001407  -2.573 0.020431 *
B:C         -0.022154   0.001407 -15.751 3.67e-11 ***
A:D          0.011228   0.001407   7.983 5.70e-07 ***
B:D         -0.013892   0.001407  -9.877 3.26e-08 ***
C:D          0.004640   0.001407   3.299 0.004532 **
A:B:C       -0.004460   0.001407  -3.171 0.005934 **
A:B:D        0.013990   0.001407   9.946 2.96e-08 ***
```

```
A:C:D       -0.005720   0.001407  -4.067 0.000897 ***
B:C:D        0.003439   0.001407   2.445 0.026440 *
A:B:C:D     -0.002358   0.001407  -1.677 0.113048
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.007957 on 16 degrees of freedom
Multiple R-squared:  0.9974,   Adjusted R-squared:  0.9949
F-statistic: 405.4 on 15 and 16 DF,  p-value: < 2.2e-16
```