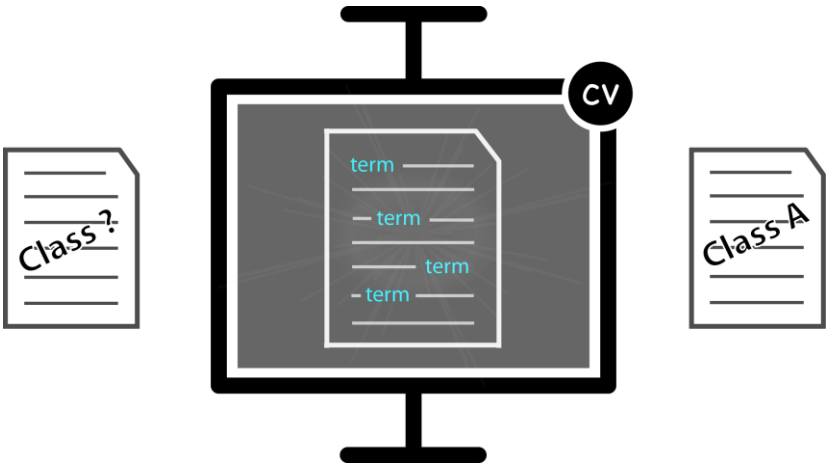
Classification of field service data using n-grams

M.P.E. Möllenbeck

2011

Master thesis

Classification of field service data using n-grams



Version 1.0.0

May 20th, 2011

Eindhoven University of Technology, Eindhoven
Philips Healthcare, Best

Mark P.E. Möllenbeck BSc – 0536390
Business Information Systems
m.p.e.mollenbeck@student.tue.nl

1st Supervisor TU/e:	dr. ir. A.J.M.M. Weijters
2nd Supervisor TU/e:	prof. dr. R.J. Kusters
Graduation Tutor RuG:	R.A. Ittoo MSc
Business Tutor Philips:	ing. C. Wolvekamp

Table of Contents

Preface

Performing a research project is the final requirement for gaining a Master of Science degree in Business Information Systems. Such a project can be completely scientific or a combination of theoretical and practical work. This project is one of the latter type, during which a practical solution has been developed using scientific research. Such type of project is what I personally favor, but what is also most appropriate in the field of Business Information Systems (BIS). BIS is a combination of Computer Science Engineering and Industrial Engineering, targeting the area where business and IT meet. Targeting that area was exactly the case during the performed project.

This project could not be done without the extensive help and support from all the people of the iXR Customer Services/Data Analyses department at Philips Healthcare, especially Cees Wolvekamp and Guillaume Stollman; the constructive help on all the technical and non-technical details from Ashwin Ittoo at Rijksuniversiteit Groningen; the nice and enduring support from my graduation supervisors Ton Weijters and Rob Kusters at Eindhoven University of Technology; and last but not least all the love and support from my family and friends, especially my parents and Nicole. Thanks for keeping faith in me.

Section 1: Introduction

1 Introduction to the subject

Many product development and customer service organizations are struggling with the rising number of customer complaints due to failures. To minimize failure rate, the product development process and delivered services need to be improved by tackling the causes of failures. Previous studies suggested that the information in product development and customer service data sources could provide insight on causes of failures (Petkove, 2003) (Heynen, 2002) (Franken, et al., 2002). A useful data source could be so called field service reports (FSRs). FSRs are reports containing the actions taken to repair a specific failure or during maintenance. Although FSRs are widely available, in most situations they are only used for declaring expenses. One might not recognize the valuable information contained by FSRs. Lack of knowledge of how to retrieve information out of FSRs might be another important factor.

Retrieving information out of some data can be as simple as reading the data. There might also be information hidden that is not directly retrievable. To get such information, data needs to be processed first. Take for example FSRs containing information about the repair of some system parts. The parts replaced are mentioned explicitly, but not the actual problem. In this situation the number of parts replaced is directly retrievable, by counting the number of replaced parts as mentioned. Information about which type of problems occur most frequently, is however not directly retrievable. For each FSR the type of problem has to be determined first, before counting can take place. Determining the type of problem for each FSR is just one way to go. One can group data into any predefined categories. This is called classification, which will be discussed in more detail in chapter two of the current section.

Classification of data can be done by hand. In case of FSRs this is doable for a small set of reports, but in case of thousands of reports this is not very efficient. Classification using computer algorithms (Manning, et al., 2008) (Sebastiani, 2002) will be more efficient, especially when the number of reports is large. The performance of the classification, in terms of quality and speed, depends on the used classification techniques. For each type of data different techniques might deliver optimal results.

This report describes the research into and application of some specific techniques to automatically classify FSRs. More specifically, the focus has been on automatic classification of FSRs containing natural language text only. Special natural language processing (NLP) techniques have been used to tackle this type of data. In the remainder of this chapter this will all be discussed in more detail.

1.1 Business context

This research project actually got initiated by an existing business problem at the interventional X-Ray business unit (iXR) of Philips Healthcare (PHC). PHC is one of the three main divisions of the Dutch founded company Philips Electronics. Besides healthcare, Philips Electronics is also involved in the area of lightning and consumer electronics. Philips Lightning is the oldest part of the company, which started as a production factory for light bulbs. Research into the field of X-Ray tubes initiated the founding of the medical division PHC. Nowadays all kinds of products and services are developed at PHC for the medical market, such as operating tables and patient monitoring systems. Not only the diversity of products being made has grown, also the number of applications of X-Ray has increased. One of the oldest applications of X-Ray is probably taking pictures of bone-structures to detect fractures. Today also more complex systems using X-Ray are built, such as cardiovascular X-Ray systems (Figure 1). Cardiovascular X-Ray systems provide live images of the interior of a patient's body during an intervention. This enables a surgeon for example, to perform an operation through just a minor incision, while following his actions by looking to the live images on a screen. For the patient such an operation is less invasive and leaves smaller scars than a traditional operation, where the part being operated has to be exposed.

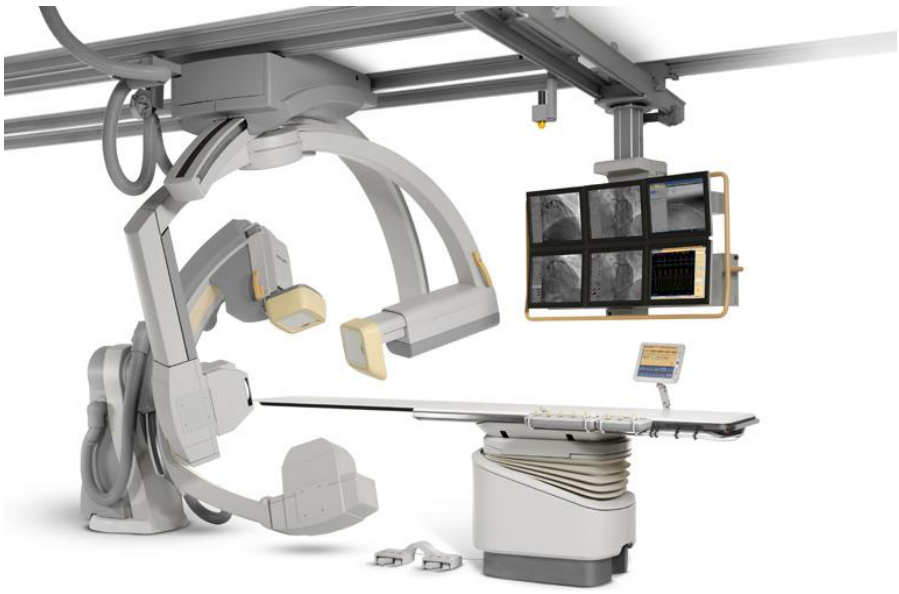


Figure 1: Example of a cardiovascular X-Ray system (Model: Allura Xper FD10/10 - Philips Healthcare)

X-Ray systems specifically designed for supporting interventions (i.e. operations), are called interventional X-Ray systems. This explains the name of the business unit iXR, which is responsible for the development and maintenance of interventional X-Ray systems. Each quarter, results need to be reported to management, such as details

about system failures and services delivered to repair them. The department Customer Services (CS) at iXR is responsible for delivering information about customer complaints and services delivered concerning interventional X-Ray systems. Not only this information is important to measure customer satisfaction, but it is also very useful to prevent future failures and complaints. Information about system failures for example, can be fed back to product development, so weak spots in the design or the production process can be improved.

The prevention of future failures and complaints is not only important for having satisfied customers and high sales, but is also important for the safety of patients. In case of a defect cardiovascular X-Ray system at a hospital, scheduled operations cannot be performed until the defect is corrected. This drives the costs for the hospital significantly. It is even more disastrous if a system breaks down during an operation, which brings the life of the patient at risk.

An important source of information about customer complaints and services delivered are field service reports (FSR). As mentioned earlier, an FSR describes the actions taken by a service engineer to correct a defect (including diagnosis) or to deliver a service. An FSR belongs to one specific call, whereas a call belongs to zero (e.g. no action taken) or more FSRs. A call is the registration of a customer complaint. It is important to notice, that a customer complaint in this context is a technical service request by the customer. Figure 2 shows the process of creation of FSRs. Once a customer complaint has been registered by a call, an FSR is created for that call each time a service engineer starts working on the problem.

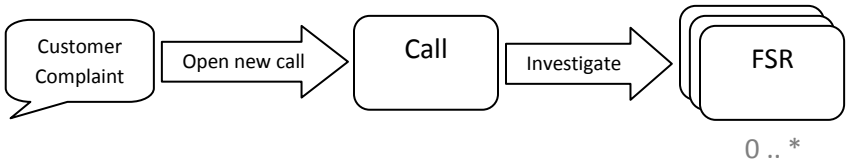


Figure 2: Process view of the relation between calls and FSRs

The FSRs for one call are bundled in a so called 'job-sheet'. A job-sheet also includes the customer complaint itself. In Figure 3 a part of a job-sheet is given as an example. Each job-sheet contains at least the following fields of information:

- Call-ID: Together with the country code, it forms a unique identification number for each call.
- Country Code: A unique number indicating the country the call is from (not shown in Figure 3**Error! Reference source not found.**).
- Call-Open Period: Period the call is created (year + month).
- Call-Close Period: Period the call is closed (year + month).
- Call Type: Indicates if it is a call about correcting a problem or something else.
- System code: A number indicating the type of system.

- Part ID: The identification number of the part replaced by a service engineer for the given call. In case there are multiple parts replaced for a call, a job-sheet is repeated for each part used.
- Part Description: Standard description of the part replaced.
- Customer Complaint: The message that initiated the call.
- Reptext: The main part of the FSRs belonging to the call. This is the textual description of the actions taken by field service engineers. The repair texts of all FSRs belonging to a call are concatenated in this field.
- Total CM cost: Indication of the total cost to correct the problem of the given call.

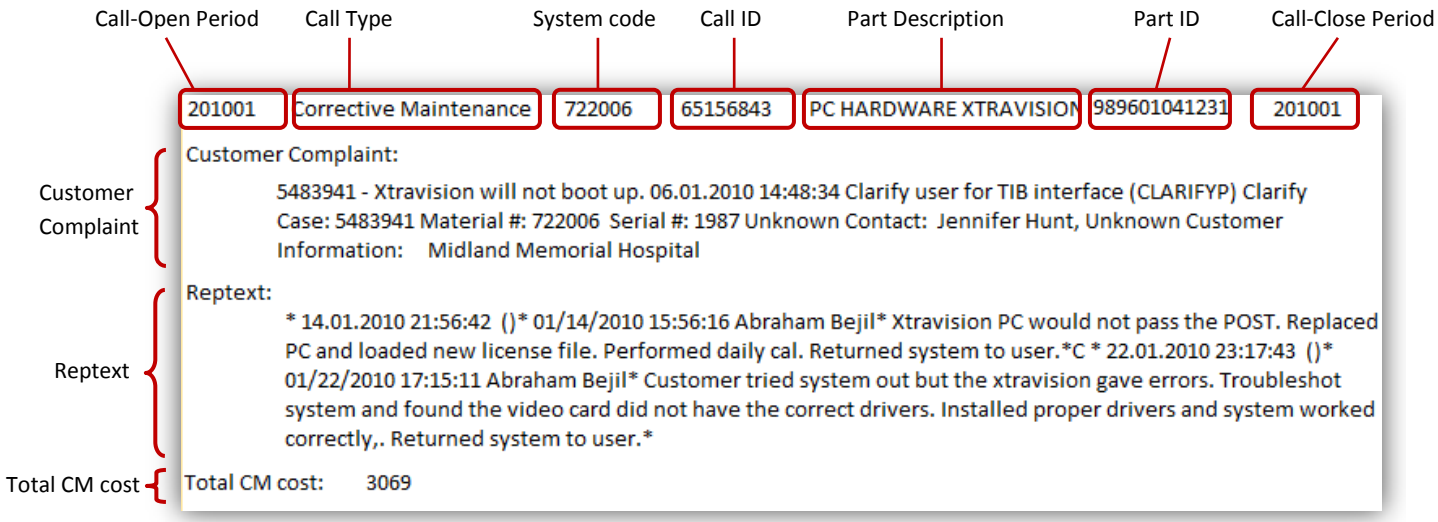


Figure 3: Example of a partial job-sheet

A job-sheet contains more fields of information, such as the number of hours worked on the problem by the FSE, but the example above should give an impression of how a job-sheet looks like. While a job-sheet belongs to just one call, we use both terms interchangeably in the remaining part of the report.

As said earlier, knowledge about problems that occur in the field, can be used to improve the design and production process of systems. An important overview that has to be delivered to management by the CS department each quarterly, is the number of calls per subsystem. A subsystem is a part of a system or its environment, possibly consisting of smaller parts like subassemblies and components.

If a subsystem is subject of a lot of calls, a closer look to the subsystem by development might be useful. Determining the subsystem(s) a call belongs to, is relatively easy for those calls containing part information. For the calls having no explicit part information, the subsystems involved have to be extracted from the textual descriptions of the complaint (customer complaint) and actions taken (reptext). One way to do this, is manually reading all those job-sheets. This is however very time consuming, because of the large number of job-sheets without parts. Manually reading

a job-sheet and assigning the subsystem(s) it belongs to, is also error-prone and subjective. One can not only misread a text, but also interpret a text differently than someone else. This leads to different job-sheet classifications among different readers. Automatizing the assignment of job-sheets to subsystems would therefore be a better way. It improves speed significantly, reduces the chance on misreading and ensures that each job-sheet is analyzed in the same way. In the current situation, a semi-automated approach is used, but the results are not sufficient enough. More details about the current situation and what would be a better approach will be discussed next, including an outline of the rest of this report.

1.2 Report outline

As said earlier, the assignment of job-sheets to subsystems would be better by doing automatically than manually. In the current situation a semi-automatic method is being used, consisting of two main steps. The first step consists of assigning subsystems to job-sheets by automatically searching the job-sheets on predefined words. For each subsystem several words are defined as indicators for that subsystem. The number of predefined words found in a job-sheet defines the subsystem it belongs to. However, a lot of job-sheets cannot be assigned to a certain subsystem, because they contain none of the predefined words. In these cases, a second step has to be performed, which is manually reading the job-sheets. For the job-sheets assigned to a subsystem in step 1, around 50% turns out to be assigned to the right sub-system. It may be obvious that the semi-automatic method currently used, is not substantially faster than the manual method. It delivers even worse results, looking at the output of the first step.

This report describes the research into an automatic method for assigning subsystems to job-sheets. Chapter two of the current section gives some background about automatic text classification and the relation to customer support information, using related work. Chapter three will provide more detail on the objectives of the research by defining research questions and the contributions made. After reading this chapter it will also be clear what is requested by the company and the limitations of the research. The first section is finished by chapter four, which describes the methodology used during the research.

Section two of this report describes the experiment performed to compare the classification techniques as proposed in chapter two of section one. The first chapter of section two describes the setup of the experiment, including details about the used data. Chapter two contains the results of the experiment performed.

Finally, the research will be summarized and conclusions will be drawn in the first chapter of the third section. The second chapter of the concerning section contains some recommendations for future work and open issues to be investigated further.

2 Background

2.1 Text classification

Classification is the task of grouping things or information into specified classes and has been done long before computers were invented. Until the late 80's, the most popular approach was one based on Knowledge Engineering (KE), consisting in manually defining a set of rules encoding expert knowledge on how to classify things or information into defined categories (Sebastiani, 2002 p. 2). The manually defined rules can be applied by hand to classify things or information, but they can also be applied in an automated way. An example of the latter method is the semi-automatic approach for classifying job-sheets, as mentioned in the first chapter.

Nowadays classification tasks are merely computer problems, like the classification of digital documents into predefined topics. A more automated approach of classification is preferable and, because of the evolved computer technology, possible. In the 90's a shift of focus took place from KE based classification to Machine Learning (ML) based classification, especially in the research community. The ML based approach consists of a general inductive process that automatically builds a classification model, by learning from a set of pre-classified information. The advantages of such ML based approach are an accuracy comparable to that achieved by a KE based approach, and a considerable savings in terms of expert labor power, since no expert knowledge is needed for the construction of the classifier. Note that for the ML approach, classification is defined as the task of building a classifier, a model by which information can be assigned to predefined categories (Manning, et al., 2008). From this point on, this definition of classification will be used. The process of assigning information to predefined categories using a classifier, will be called *categorization* from this point on.

The categorization of natural language text (Lewis, et al., 1994) is a specific type of categorization. As already mentioned in the previous chapter, natural language text is not directly understood by a computer like a numeric value. A computer only sees a sequence of characters, no more no less. To be able to automatically categorize text (Joachims, et al., 2002), machine-readable information has to be extracted from the text using Natural Language Processing (NLP) techniques (Roth, 1998) (Jurafsky, et al., 2008). Common used NLP techniques are based on linguistics, statistics (Manning, et al., 1999) or a combination of those two (hybrid). Linguistic approaches, like part-of-speech tagging, identify terms based on their formation patterns. Statistical approaches, like log-likelihood, identify terms based on their occurrence frequencies. The mentioned approaches extract terms as machine readable information, which means that a computer program can recognize them as a form of knowledge. Terms are information-bearing lexical units, which can be words or phrases expressing some domain-specific knowledge. Terms need not to be confused with general words, because general words do not express domain-specific knowledge. For example "X-Ray tube" is a term, while "the" is a general word.

NLP-techniques can be useful to extract features, but other (non-linguistic-based) techniques exist to extract other type of features than terms, like n-grams (Rahmoun, et al., 2006) (Cavnar, et al.). N-grams seem to be very useful for classifying field service data, because of the properties of this type of data. We will discuss the use of n-grams in chapter 4 of this section.

Figure 4 shows the general process of text categorization (TC). A seed file contains pre-categorized texts or documents, which will be used for training (B) a model (classifier). Using the model, uncategorized documents or texts can be categorized (C). Optionally information about wrongly categorized documents can be given as feedback to the model for improvement. Input to the process, such as a seed file and uncategorized documents, need to be pre-processed first (A) so the textual input is readable by a computer.

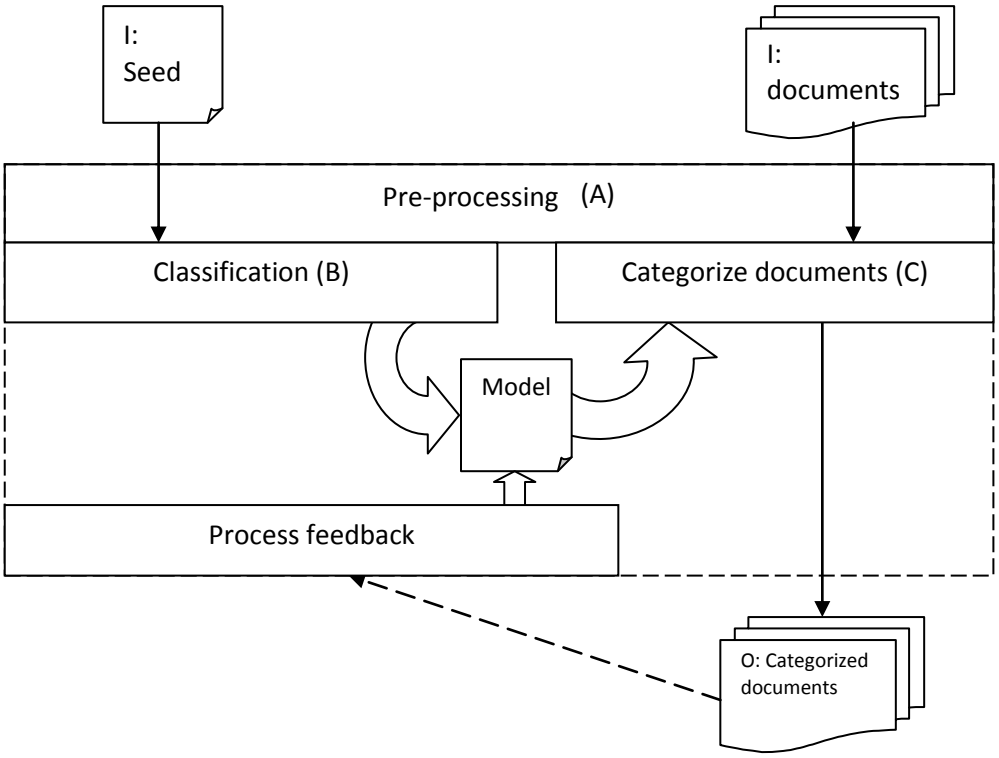


Figure 4: process of categorizing textual documents

2.2 Classification techniques

For the classification and categorization part, several methods exist. Some best known are Naïve Bayes (NB), Support Vector Machine (SVM) and Concept Vector-Based (CVB). More methods exist (Sebastiani, 2002) (Manning, et al., 2008), but because of the limited time, not all of them are discussed here. Besides, the three mentioned methods are also among the most used for text classification.

Naïve Bayes

Naïve Bayes belongs to the group of probabilistic classifiers (Manning, et al., 2008 pp. 219-235) (Kim, et al., 2000). This type of classifier computes the probability that a document represented by a vector $\vec{d}_i =< o_{1i}, \dots, o_{|T|i} >$, where o_{li} represents the number of occurrences of term l in document i, belongs to a certain class (category) c_j using Bayes' theorem. This can be expressed by:

$$P(c_j|\vec{d}_i) = \frac{P(c_j)P(\vec{d}_i|c_j)}{P(\vec{d}_i)}$$

$P(\vec{d}_i)$ is the probability that a randomly picked document matches the vector $\vec{d}_i$. This probability is the same for each document in a collection, so this value will not be used in actual computations. $P(c_j)$ is the probability that a randomly selected document belongs to class c_j. To make the computation fast and achievable for $P(c_j|\vec{d}_i)$, it assumed that any two terms in document $\vec{d}_i$ are statistically independent. This is called the independence assumption and, as expressed by the name of this type of classifier, is a bit of naïve assumption (Lewis, 1998). In practice the occurrence of one term is in some cases indeed related to the occurrence of another term in a document.

Naïve Bayes classifiers are very popular in practice for TC, while they are easy to understand and work relatively fast. Surprisingly the results of Naïve Bayes classifiers are also quite impressive (Yang, et al., 1999). However, more complex classifiers are available which deliver better results. Naïve Bayes is also very sensitive to sparse data, containing less frequent words.

Support Vector Machine

Support Vector Machine (SVM) (Joachims, 1998) (Burges, 1998) is a binary classification method that tries to find the best possible decision surface dividing the negatives (not-belonging) from the positives (belonging) for a specific category. The best possible decision surface is in this case defined as the one dividing the positives from the negatives by the widest possible margin. In figure 5 an example of such a best decision surface σ_i is graphically represented in a two dimensional and linearly separable space. The other lines represent examples of non-optimal decision surfaces. The SVM method is also applicable to the case in which the negatives and the positives are not linear separable. SVM is also very useful in case of multiple categories. In that case, the SVM method needs to be applied for each category, to find the best possible decision surface for each category.

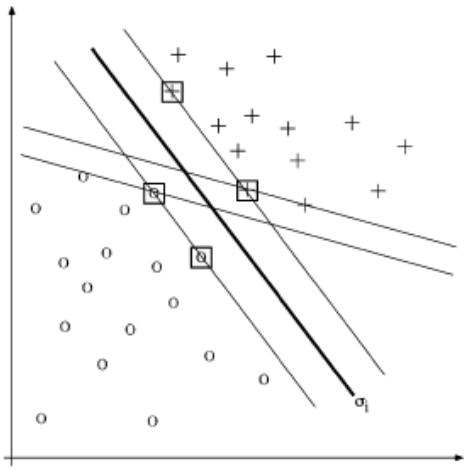


Figure 5: Schematic view of support vectors and decision surfaces in a term vector space

For the classification of well-formed textual data, SVM turns out to be one of the best methods currently available (Dumais, et al., 1998). Advantages of SVM regarding other classification methods, are robustness to overfitting and the capability of handling considerable dimensionalities (terms). It is true that some parameter values need to be estimated beforehand to get the best results. It can take up quite some time to find good values (manually or automatically). However, it also possible to use some standard values for the parameters. Using standard parameter values will not deliver the best results, but is much faster.

Disadvantages of SVM are its complexity and high processing time (Joachims, et al., 2002). It is also questionable how accurate this method will be in case of sparse textual data, while the best results are almost all achieved in case of well-formed textual data.

Concept Vector-Based

The Concept Vector-Based Classifier (CVB) is a very simple method, derived from the Information Retrieval paradigm (Jurafsky, et al., 2008 p. chap. 23). Each document is represented by a vector $\vec{d}_i =< w_{1i}, \dots, w_{|T|i} >$, where w_{ti} is the weight of term t in document i, which will be normalized first so that it is of unit length. Each category is represented by a concept vector $\vec{c}_j =< w_{1j}, \dots, w_{|T|j} >$, where w_{tj} is the weight of term t in category j , which is computed by summing up all normalized vectors of the documents belonging to category j. The computation of the concept vectors, which is the classification part, is not that complex but can take up some time in case of a large term space.

Classifying uncategorized documents (test documents) using CVB is also that simple and foremost, it is fast. For a given test document represented by the vector $\vec{d}_x =< w_{1t}, \dots, w_{|T|x} >$, which need to be normalized first, the similarity between $\vec{d}_x$ and each concept vector $\vec{c}_j$ will be computed. A

well-known method for determining the similarity between two vectors is computing the cosine similarity. The smaller the angle between two vectors, the smaller the value for the cosine similarity and the more similar the two vectors are. The test document will eventually be assigned to the category represented by the concept vector most similar to the test document vector.

Besides the low processing times during classifying, another advantage of CVB is the summarizing of characteristics of each class in the form of concept vectors. For example, prominent dimensions of a concept vector, which are terms having high weight values, are not necessarily terms occurring frequently in all documents belonging to a category. These terms might also occur frequently in just a subset of documents belonging to a category. This is important for high dimensional and sparse data sets for which coverage of any individual feature is quite low.

2.3 Challenges

The discussed classification methods all have their strengths and weaknesses. What specifically is important for this research, is the suitability of a method for field service data. According to earlier test results (Sebastiani, 2002 p. 38), SVM performs much better than NB and CVB. However, these results are achieved using well-formed general texts, containing few spelling and typographic errors and mostly general language words. Field service data, such as the repair text of a job-sheet, has some other characteristics which might influence the performance of the methods. These characteristics can be defined as challenges to tackle in classifying such type of data.

Fuzzy text

Field service data is fuzzy, which means it is not that nice and faultless written as general texts. Fuzzy texts contain relatively a lot of spelling errors and typo's, which makes it difficult to find similar terms and expands the set of features unnecessary. Grammatical errors like missing punctuation are also a form of fuzziness, which can lead to incorrect multi-word terms. Take for example the following phrase of a fuzzy text: "The CPU fan has been calibrated plugs have been placed back". In this example a closing dot misses after the word "calibrated", to mark the end of a sentence. In case the dot is in place, "calibrated" and "plugs" can be retrieved as separate terms. However, if multi-terms are allowed, "calibrated plugs" could be retrieved as a multi-term in the current example without a dot. This would be incorrect, because those two words are not directly related to each other in the given example.

Short texts

Opposed to standard textual documents, such as news-articles and books, field service data contains short texts. What can be defined as short is difficult to say, but in case of job-sheets a couple of lines text is the average length. For standard textual documents, one or more pages of text would be more the average length. The length of a text is an important factor for successfully

classifying it, while the more information being extracted, the better it can be identified. Note that not always long texts are better, because the number of unique terms in a text also contributes to the success factor. A text with only one word repeated 1000 times, is for example not better to classify than a text with only the same word just stated one time.

Domain specific

Typical for field service data is the domain specific information it contains. In more general texts mostly words are used that can be found in standard dictionaries. However, in case of field service data, a lot of words are used that are specific for the field, which don't occur in standard dictionaries. Without a knowledge resource like a dictionary, or having a lot of words not occurring in knowledge resources, it is difficult for classification programs to recognize these words as terms. NLP techniques like lemmatizing and POS-tagging (Jurafsky, et al., 2008) are difficult or even impossible to apply to such words.

Different languages

Products and services are sold all around the world these days. In case of problems, it is not always possible to retrieve a product back to the fabric. This is for example the case for Cardio Vascular X-Ray systems shown in chapter one of the current section, which are simply too large to send as a whole and are too complex to decompose in a fast and efficient way. To be able to deal with product problems, a field service engineer will visit the location where the product is installed. While these locations can be spread all over the world, field service engineers are confronted with different languages; customers who can only communicate in their native language and systems configured to read and write texts in a specific language. As a consequence, field service data contains texts written in different languages.

NLP techniques being used to retrieve terms out of texts, are very language dependent. For example the construction of stems of English terms is very different than the construction of stems of Greek terms. So a different method for stemming needs to be used for each of the two languages in the example. A classifier build for one language, is not automatically suitable for another language and hence, a lot of adaptations might to be made to the classifier for each language. This is however very time-consuming. Another possibility to tackle the problem of multi-language data, is to translate all the non-English texts to English. Translation algorithms are however not perfect, so translated texts might not correctly express the original texts.

3 Objectives

In the previous chapter we have introduced the subject of text classification and some well-known classification techniques. We also defined four challenges in case of using field service data with text classification. These four challenges give rise to two implications, which we formulate as our research questions (RQ).

RQ1: Contribute simple character-based feature selection methods to a better classification result than more advanced linguistics-based feature selection methods, in case of domain-specific texts like field service data

RQ2: In case simple character-based extraction methods are used, are simple classification techniques more suitable for classifying domain-specific texts like field service data than more advanced classification techniques

To be able to answer these research questions, a couple of contributions have been made. The most important contribution is the development of an industrial strength text categorization tool for field service data, specifically for the categorization of job-sheets. The tool has been given the name *Clavis Verbum* (CV) tool, which is Latin for "keyword" and stands for an important concept of classification: a term which enables to classify a text into a predefined category. The tool's name contains also a little joke, while its abbreviation is equal to the abbreviation of cardio vascular, the domain for which the tool is intended initially. The tool has been built as part of the Data Fusion project, which is a project performed by Rijksuniversiteit Groningen (RuG), Eindhoven University of Technology (TU/e) and some major industrial partners. The purpose of this project is to develop techniques for combining and extracting product data to improve the development process.

The CV tool plays a central role in the other contributions, which directly address the aforementioned research questions. These contributions are:

- The measurement of the effect on classification performance for job-sheet data, in case of using n-grams, with respect to using more sophisticated retrieved features using NLP techniques. (To find an answer to RQ1)
- The comparison of three classification methods, each of a different complexity level, by measuring the performance of each of these methods applied to job-sheet data and using n-grams. (To find an answer to RQ2)

These contributions will be explained in more detail in the remainder of this document, especially in section 2, which deals about the experiment performed. The research in total and the contributions specifically, have been kept manageable by defining some constraints, which are:

- Only dealing English text
- Each job-sheet has exactly one category (no multi-category)

4 Methodology

In this chapter it will be made clear which steps and tasks have been performed to meet the contributions as stated before. All these steps and tasks together form the used methodology and are in fact the ingredients for the experiments, as described in section two.

As mentioned in chapter two of this section, we are interested in ML approaches for classifying textual field service data. Figure 4 contains a high-level overview of an ML based TC methodology. Each of the components in this overview can be divided into smaller steps, which can be shown in Figure 6 for components A, B and C.

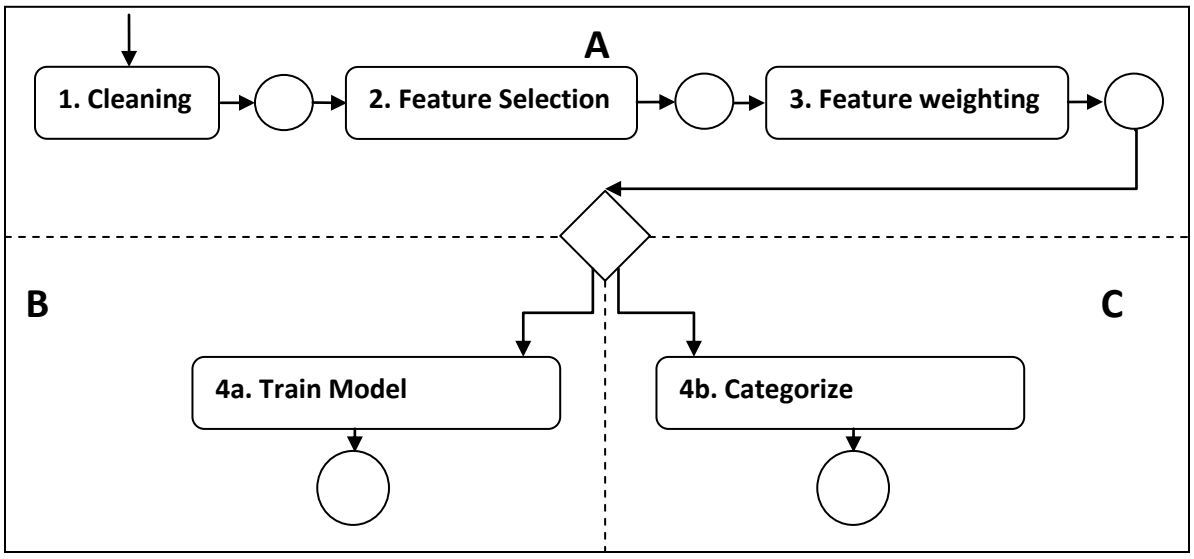


Figure 6: Steps TC methodology

Each step is composed out of several tasks. The number and type of tasks affect the performance of the TC methodology and need to be chosen well. The tasks defined for each step in our TC methodology will be explained below.

4.1 Cleaning

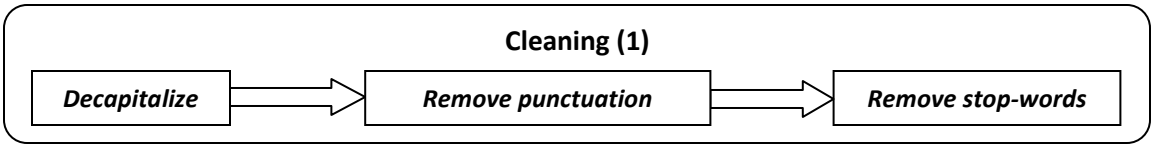


Figure 7: An overview of the tasks at the cleaning step

One of the first tasks to perform, is filtering out as much irrelevant information as possible, to limit the feature dimension space. Information is irrelevant for classification purposes, in case it occurs in almost every text, like the word "to" for example.

Fortunately two types of irrelevant information can be easily recognized and removed from texts: punctuation characters and so called stop-words.

Before the removal of punctuation characters and stop-words, we first decapitalize all characters. A regular expression is used to replace each capital with its lowercase. This is done to prevent that a certain word written with a capital, is recognized as a different word against the same word with no capital. Figure 8 gives an example of applying the regular expression to a whole sentence.

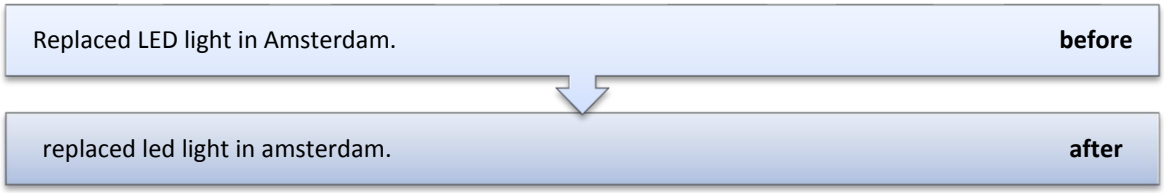


Figure 8: Example of replacing capitals by lowercases.

The next task of the cleaning step is about removing punctuation characters. We have defined the following non-alphanumeric and numerical characters as punctuation characters:

(punct_1) `!@#$%^&*()-_€+`~\|][}{=;:'",.><?/1234567890`

The non-alphanumeric characters defined as punctuation characters, are all those non-alphanumeric characters that are shown on a standard keyboard, having US-international layout. While most computers have a keyboard with a US-international layout and we only deal with English texts, we may assume that no other non-alphanumeric characters are typed. In case of texts written in another language, the set of punctuation characters may be defined differently. Note that the definition of the punctuation characters given by punct_1, depends on the second step in A (Figure 6), feature selection. As we will see later on, it may be useful to keep some punctuation characters till the second step.

The actual removal of punctuation characters is done using regular expressions. First a regular expression is used to replace all defined punctuation characters by a single whitespace. The punctuation characters are not simply removed, because we want to keep two or more words divided by a punctuation character, also separated after removal of the punctuation character (e.g. "test-machine" -> "test machine"). Next a regular expression is used to replace all multiple whitespaces by a single whitespace, while multiple whitespaces after each other might occur after replacing punctuation characters by whitespaces. Figure 9 gives an example of a sentence having punctuation characters (before) and the resulting sentence after replacing punctuation characters by a single whitespace (interm) and replacing multiple whitespaces by a single whitespace (after) using definition punct_1.

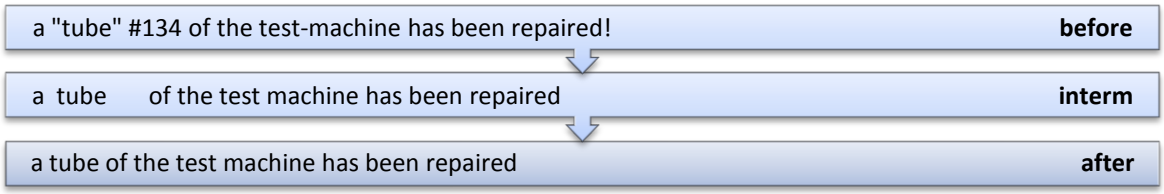


Figure 9: Example of punctuation removal using the definition of punct_1. As we can see, all the characters *"#134-!* are removed and only single whitespaces remain.

After the punctuation characters have been removed, the so-called stop-words will be removed. The order in which these tasks do take place, is irrelevant for the outcome. The search for stop-words is however a bit more intensive than the search for punctuation characters and therefore it is preferable to remove the punctuation characters first to reduce the text.

Stop-words are general language words which are used very often and in almost all texts. One could think of words like "the", "or" and "he". There is not any definite list of stop-words available. It is interpretable which words are stop-words and which are not. Besides, stop-words are language dependent, so for each language, a different list of stop-words needs to be used. Fortunately we only have to deal with English text and examples of lists of English stop-words can easily be found on the internet. The list we have used, can be found in appendix A. This list does not only contains general English stop-words, but is also extended with some domain-specific irrelevant words, delivered by domain experts.

To find and remove stop-words, a text is first tokenized. This is done by splitting the text on white-spaces, delivering white-space-free chunks called tokens. Then, each token is compared with each stop-word in the stop-wordlist. In case a match is found, the token will be removed and the next token will be compared to each stop-word. When eventually all tokens have been checked, the tokens that still remain will be concatenated by white-spaces to form one text again.

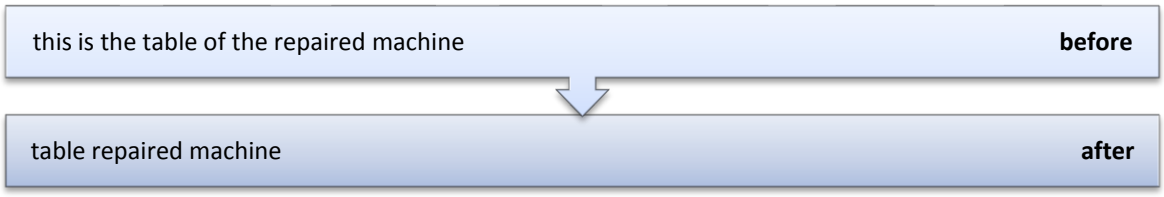


Figure 10: Example of stop-word removal using the stop-wordlist of appendix A. As we can see, the stop-words "this", "is", "the" (2x) and "of" are removed from the sentence.

4.2 Feature Selection

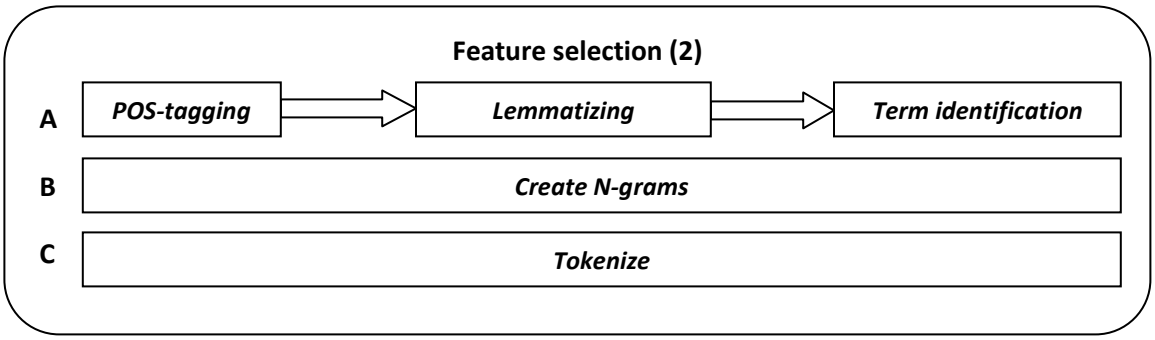


Figure 11: An overview of the tasks at the feature selection step. A, B and C indicate the level of sophistication of the feature selection, where each level has a different set of tasks.

The feature selection step contains tasks to filter out features from the cleaned texts. The way this step is performed, depends on the sophistication of the feature selection. For this research we have defined three levels of sophistication. Figure 11 shows the tasks for each level of sophistication (level A, B and C), where A is the most sophisticated level and B and C are the least sophisticated levels. Level B creates n-grams (Cavnar, et al.) (Rahmoun, et al., 2006) and we think this is the most suitable feature selection method to use for field service data. The other two levels are useful to compare the results of using n-grams against more or less sophisticated methods. Level A uses linguistics, which deliver good results for well-formed texts normally. Level C just tokenizes text, so no further preprocessing is done for this level. The last level is useful to see if n-grams perform better doing nothing at all. We will describe the tasks for each level below.

Level A

As already said, level A uses linguistics to filter out features from text. The first task at this level tries to label each word of a text with the part-of-speech it belongs to (POS), called POS-tagging. A part-of-speech is a linguistic category of words, such as *noun* and *verb*. The POS-tagging is done using a digital dictionary. For each specific language we need to use a dictionary for that language, because of the different set of words and possibly a different set of linguistic categories (e.g. not every language has a distinction between *adjective*s and *verbs*). In case of English, we can use the Wordnet dictionary (Miller, et al.), which is very popular in the field of NLP and is freely available. Using the WordNet dictionary, each word of a text is being looked up to retrieve its POS-tag. In case a word can belong to more than one linguistic category, it is the job of the POS-tagger to choose the right POS-tag using context information. The word "fast" for example, can be an *adjective* as in "a fast car" or an adverb as in "he droves fast".

Besides words, also punctuation characters are tagged by a POS-tagger, like a dot being tagged as "sentence-final punctuation". This can be useful for the term identification task, which we will see later on. Punctuation characters occurring after or before any words, are separated by a white space first. This is done to be able to POS-tag a word, separately from a surrounding punctuation character.

Several algorithms exist for POS-tagging. The one we have used, is the "English Maximum Entropy POS-Tagger", included by the freely available SharpNLP package (Northedge), a C#.NET implementation of the popular OpenNLP package. The algorithm has been used in combination with WordNet. An example of how POS-tags are assigned during the POS-tagging task of our methodology, can be found in Figure 12. Table 1 gives an overview of all the POS-tags that can be assigned by the used algorithm.

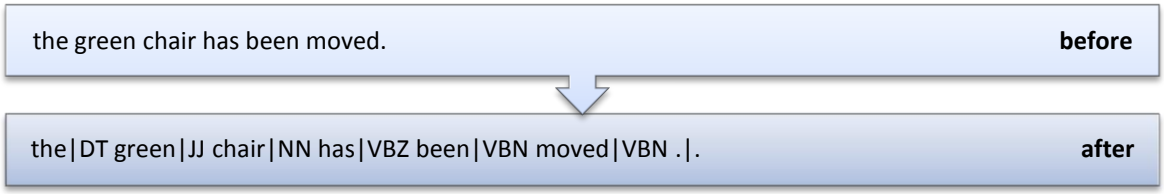


Figure 12: Example of POS-tagging using the English Maximum Entropy POS-Tagger of the OpenNLP package. For each word or punctuation character, the belonging POS-tag is being placed after it, separated by "|".

CC	Coordinating conjunction	RP	Particle
CD	Cardinal number	SYM	Symbol
DT	Determiner	TO	to
EX	Existential there	UH	Interjection
FW	Foreign word	VB	Verb, base form
IN	Preposition/subordinate conjunction	VBD	Verb, past tense
JJ	Adjective	VBG	Verb, gerund/present participle
JJR	Adjective, comparative	VBN	Verb, past participle
JJS	Adjective, superlative	VBP	Verb, non-3rd ps. sing. present
LS	List item marker	VBZ	Verb, 3rd ps. sing. present
MD	Modal	WDT	wh-determiner
NN	Noun, singular or mass	WP	wh-pronoun
NNP	Proper noun, singular	WP$	Possessive wh-pronoun
NNPS	Proper noun, plural	WRB	wh-adverb
NNS	Noun, plural	``	Left open double quote
PDT	Predeterminer	,	Comma
POS	Possessive ending	''	Right close double quote
PRP	Personal pronoun	.	Sentence-final punctuation
PRP$	Possessive pronoun	:	Colon, semi-colon
RB	Adverb	$	Dollar sign
RBR	Adverb, comparative	#	Pound sign
RBS	Adverb, superlative	-LRB-	Left parenthesis
		-RRB-	Right parenthesis

Table 1: Overview of all the possible POS-tags that can be assigned by the POS-tagger. After each POS-tag, a short description is given of the abbreviation.

The second task of level A tries to minimize the set of features, by grouping the different inflected forms of a word to a single root form. In the area of NLP there are two well-known techniques to achieve this (Manning, et al., 2008 pp. 32-34) (Hull, 1996). The simplest and fastest one is stemming. Stemming finds the root form of a word by just reducing it syntactically to a base form. For example, the words "fish", "fisher", "fishing" and "fished" are all reduced to the root form "fish". In case of the words "am" and "was", stemming will not find the root form "be", because simply reducing the two words will not deliver something near "be". A better but more complex technique than stemming, is lemmatization, which can find the root form "be" for the words "am" and "was". Lemmatization uses normalization rules and a dictionary to look up root forms. This makes it possible to group semantically-equal words into one root form. For example, "car" and "automobile" can be replaced by the root form "car". For lemmatization, it is important to know the POS-tag of a word, because the root form of a word might be different for each part-of-speech. For example, the word "meeting" can be a noun or a form of a verb, depending on the context. The root form of the noun "meeting", like in the sentence "The meeting has started", is "meeting". However, the root form of "meeting" as a form of the verb "to meet", like in the sentence "we are meeting each other tomorrow", is "meet".

In our approach we made use of the lemmatization algorithm of the SharpNLP package. The mentioned lemmatization algorithm takes as input a word with its corresponding POS-tag and outputs all possible root-forms. If no root form is found, the original word is kept; else the original word is replaced by its root form (Figure 13). In case multiple root forms are found for a given word, the shortest root form is taken. This is useful in case we have for example the words "automobile" and "car", which both have root forms "car" and "automobile". By choosing the shortest one, both words are replaced by the same root form, which is "car".

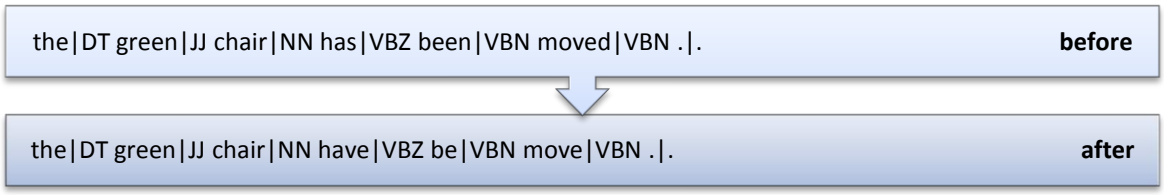


Figure 13: Example of lemmatizing using the lemmatizing algorithm of the SharpNLP package. As can be seen, the word "has" is replaced by its root form "have", "been" by its root form "be" and "moved" by its root form "move".

An important remark to make, is the fact that the used lemmatization algorithm only accepts three different POS-tags: "adjective", "verb" and "noun". Therefore the POS-tags as generated by the first task, have to mapped to one of these three tags first, before being able to apply the lemmatization algorithm. The used mapping can be found in Table 2. Words having POS-tags, not defined by this mapping, are not lemmatized. Punctuation characters are therefore never lemmatized.

POS-tags generated by the first task		POS-tag used by lemmatization
JJ, JJR, JJS	⇒	Adjective
VB, VBD, VBG, VBN, VBP, VBZ	⇒	Verb
NN, NNS, NNP, NNPS	⇒	noun

Table 2: Mapping of POS-tags generated by the first task, to the corresponding POS-tag as used by the lemmatization algorithm.

The third and final task of level A of the feature selection step, is about term identification, where we select the actual features. For level A, these features are noun-based terms, which are single nouns or multi-word phrases around one or more nouns. One can identify very complex multi-word terms, not only around nouns. We have kept this step very simple however, because of the fuzziness of the texts, which does not lean to identify complex multi-word terms. The noun-based terms are identified using a very simple regular expression:

(regexpr_terms) (VBG?)(Adjective*)(Noun+)

The POS-tags "Adjective" and "Noun" used in the expression above, are the same as used in the mapping for lemmatizing (Table 2). The POS-tag "VBG" is the same is assigned by the POS-tagger of the first task (Table 1). Now only terms which comply to the regular expression (regexpr_terms) are filtered out of a text. Figure 14 shows the result of this task for a given POS-tagged and lemmatized sentence. What it does, is reading the sentence from left to right and for each word or punctuation character, it looks at its POS-tag. Note that first all noun- and adjective-related POS-tags need to be mapped to their corresponding POS-tags as defined in Table 2. In case the current word being looked at is a noun, it will be kept without its POS-tag. In case it is an adjective or a verb of the form "VBG", it needs to be directly followed by zero or more adjectives and eventually one or more nouns, to be kept. The other way around this means, that a noun directly preceded by zero or more nouns, directly preceded by zero or more adjectives, directly preceded by zero or one "VBG" is identified as one term. A sequence of words forming a valid term according to the specified regular expression, is concatenated with a "_" between each word. Also all POS-tags of words in a multi-word term are removed. All other combinations of words based on their POS-tags are discarded and removed.

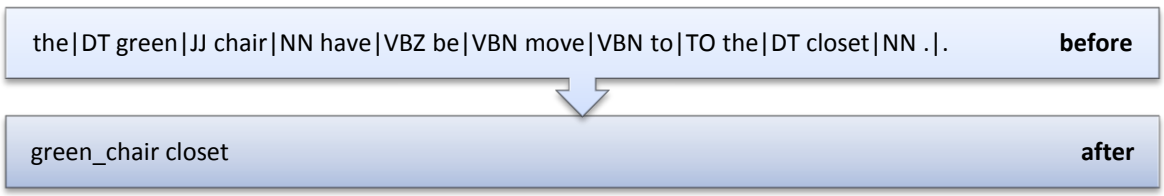


Figure 14: Example of term identification using the regular expression *regexpr_terms*. The identified terms are eventually "green_chair" and "closet".

The performance of the term identification task might be influenced by the settings of the punctuation removal at the cleaning step, as mentioned before. For example,

sentence closing punctuation characters like dots, not only mark the end of a sentence, but also indicate the separation of two words. If the last word of a sentence is a noun and the first word of the next sentence is also a noun, the punctuation character can help to identify these two nouns as two separate terms. In case the punctuation character is removed at the cleaning step, these two nouns are mistakenly identified as one multi-word term.

Level B

A great difference between the three levels is that level A uses mainly NLP techniques and levels B and C do not use NLP techniques at all. While NLP techniques can improve feature selection, they are most useful in case of well-formed (grammatically correct) and clean (almost no spelling errors and typo's) texts. This is, as defined by the challenges in chapter 3 of this section, not the case for field service data like the job-sheets we use in our experiments, described in section 2. Instead of selecting linguistic-based features like terms, where only existing words as defined by a vocabulary are used, we can also select character-based features like raw tokens and n-grams. The selection of raw tokens is part of level C and will be discussed later on. Now we will focus on the selection of n-grams, which is the main task of level B.

The word *n-gram* is used for several different definitions in literature. An n-gram can for example be defined as a sequence of n words. Here we define an n-gram as a sequence of n characters. That is why we called level B a character-based feature selection. Let n=3, then an n-gram (3-gram) of the word "computer" is "com". Another 3-gram of the same word is "mut", because it is also a sequence of three characters of the given word ("co**m**p**ut**er"). An n-gram of the latter form has however little value, because the relation with the word it is taken from, is almost gone. The n-gram can relate to any word having the three characters "mut" in it, so the information that is given by this n-gram is lesser than the information given by the n-gram "com", which retains some of the structure of the original word. Therefor we have defined the following constraint:

(constr_1) Only sequences of directly consecutive characters are allowed

Having defined the constrain "constr_1", all 3-grams of the word "computer" are: "com", "omp", "mpu", "put", "ute", "ter". Following this, an n-gram can now be seen as a character-window of size n, which makes only a sequence of n consecutive characters visible (Figure 15). To get all n-grams of a phrase, like the one in Figure 15 (where n=3),

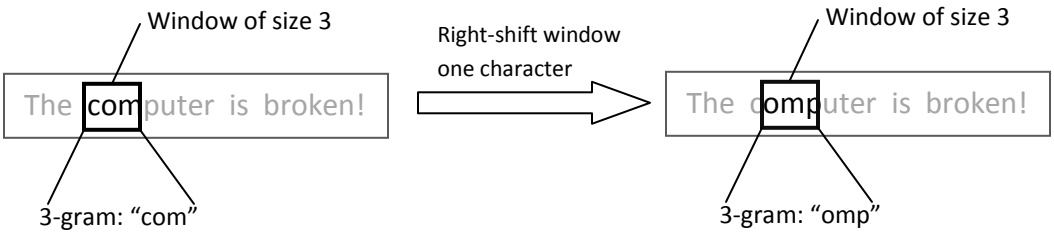


Figure 15: N-gram visualized as a character-window of size n (n=3). To retrieve the next n-gram, a shift to the right of the size of one character needs to be made by the window.

the window is placed on the left-most three characters ("The"), which is the first n-gram. Now the window is shift to the right by one character each time, to get the next n-grams, until the window has reached the last three characters of the phrase, which is the last n-gram. The total number of n-grams that can be retrieved from a text with a size of m characters, can be expressed by the following formula:

(form_ngram_1) #n-grams for m characters = m-(n-1)

The formula can easily be explained using the window example again. When the window is placed on the first n characters, the window can be right-shifted until the window covers the last n characters of the text. The number of possible right-shifts is then the number of characters m minus the last n characters, for which no right-shifts are possible anymore. Finally we have to add one to this number, because the starting position of the window (the first n-gram) has to be counted also. Summarized we get m-n+1, which is mathematically equal to m-(n-1), and that is exactly form_ngram_1.

An important remark to make is that white-space characters are also treated like normal characters. This means that n-grams can contain white-spaces. For visibility and to ease processing of n-grams later on, all white-space characters in n-grams are replaced by an underscore character "_". This makes it possible to place all n-grams after each other, divided by new white-spaces. Note that an underscore character indicates the beginning or ending of a word, which gives an n-gram a higher information-value. By replacing only white-space characters by an underscore character, we forget valuable information about the starting of the first word and the ending of the last word of a text. Therefor a white-space character is added at the beginning and ending of a text, before all n-grams are retrieved. As a consequence, the number of n-grams is now raised by two. So we reformulate form_ngram_1 as follow:

(form_ngram_2) #n-grams for m characters = m-(n-1)+2 = m-n+3

The value for n, which defines the size of the n-grams, needs to be chosen well. If n-grams are too small, they might not exposure enough distinctive information, because then they occur in a lot of texts. On the other hand, if n-grams are too large, they might be too distinctive, by including multiple words in one n-gram. According to results in earlier research projects (Rahmoun, et al., 2006) and from experiments (see Section 2), n-grams of size five (5-grams) seem to deliver the best results for the classification part, so that is what we have used in all our experiments. To summarize the whole task of creating n-grams, Figure 16 gives an example of retrieving 5-grams (after) from a given sentence (before).

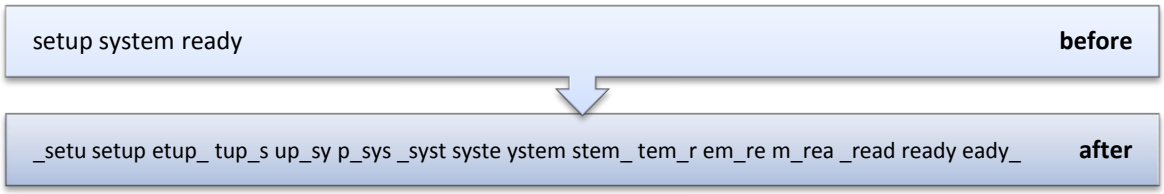


Figure 16: Example of creating 5-grams. Whitespaces are also treated like a single character and are replaced by "_" in the 5-grams for visibility.

Level C

Level C is the least sophisticated level of the three defined. This level is used as a baseline, to see if more sophisticated approaches are really effective on this type of data. The main task of level C is very simple. It splits a given text on white-space characters, delivering just raw tokens (mostly single words). An example of the output of this task for a given input is given in Figure 17.

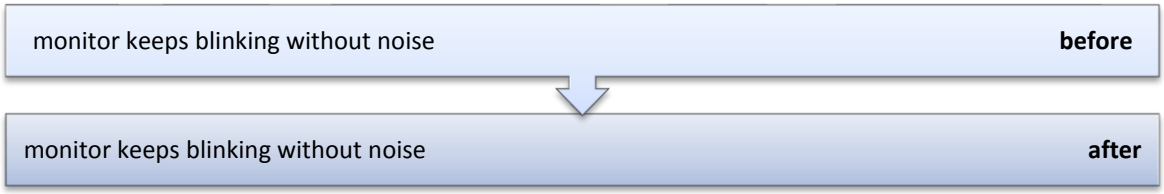


Figure 17: Example of creating raw tokens, by splitting the given text (before) on white-space characters. While no other processing is done, the output is the same as the input.

4.3 Feature Weighting

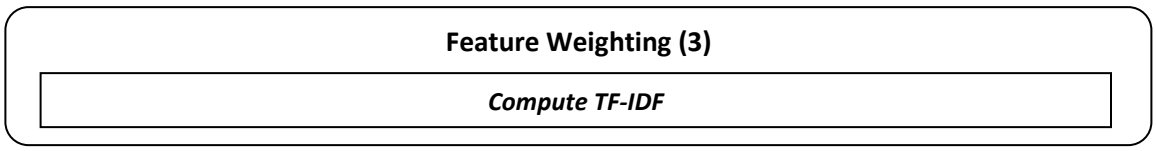


Figure 18: An overview of the tasks at the feature weighting step.

Not all features are equally important, which means a feature has some value of importance, called a weight, to distinct a document or category from one another. Computing the weight of a feature is called weighting and several methods exist. The simplest method is computing the Term Frequency (TF): the number of occurrences of a feature in a document or category. TF does however not take into account the other documents or categories. In case a given feature x has a high TF-value, but x occurs in almost all documents or categories, the distinctive power of x is still very low.

A better method is computing the Term Frequency – Inverted Document Frequency (TF-IDF), taking into account the other documents or categories:

$$TF - IDF(f_k, c_m) \;=\; TF(f_k, c_m) * Log\left(\frac{|C|}{df(f_{k)}}\right)$$

Where

$$TF(f_k, c_m) = number\ of\ occurences\ of\ feature\ f_k\ in\ category\ or\ document\ c_m$$
$$|C| = \ total\ number\ of\ categories\ or\ documents$$
$$df(f_{k)} = The\ number\ of\ categories\ or\ documents\ in\ which\ feature\ f_k\ occurs$$

Table 3 shows the TF, DF and TF-IDF for the following two example texts belonging to the same category:

- What car was driving in river red or blue
- One person was red and one person was blue or green

Having weighted all features, a simple dimensionality/feature reduction can be made by removing all features having a low weight. This improves speed and reduces the storage size in memory. However, we have chosen to keep all features, because speed and memory usage is not important for this research.

Feature	TF	DF	TF-IDF
what	1	1	1*Log(2/1) = 0.30
car	1	1	1*Log(2/1) = 0.30
was	3	2	3*Log(2/2) = 0
driving	1	1	1*Log(2/1) = 0.30
in	1	1	1*Log(2/1) = 0.30
river	1	1	1*Log(2/1) = 0.30
red	2	2	2*Log(2/2) = 0
or	2	2	2*Log(2/2) = 0
blue	2	2	2*Log(2/2) = 0
one	2	1	2*Log(2/1) = 0.60
person	2	2	2*Log(2/2) = 0
and	1	1	1*Log(2/1) = 0.30
green	1	1	1*Log(2/1) = 0.30

Table 3: Intermediate results of the computation of TF-IDF values for three given example texts belonging to the same category, where the total number of documents |C| is three in the TF-IDF formula.

4.4 Train model & categorize

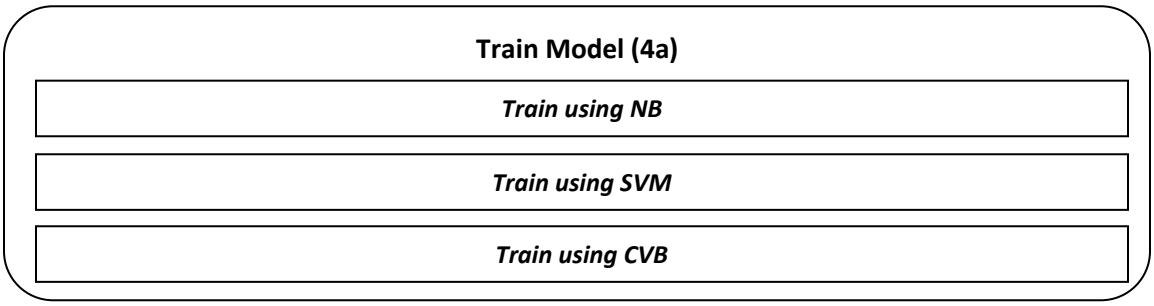


Figure 19: An overview of the tasks at the model training step. For each used classifier, a different training task is defined.

The task performed to train a model, is different for each classifier being used (Figure 19). Once a model is trained, using pre-processed labeled texts, it is temporarily kept in memory, until a new model is trained. More details about the models trained, can be found in paragraph 2 of chapter 2.

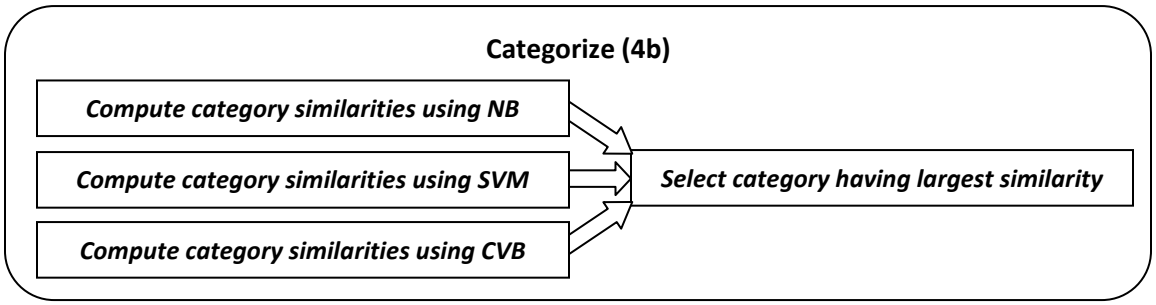


Figure 20: An overview of the tasks at the categorizing step. For each used classifier, a different similarity computation task is defined. The category selection task is the same for all classifiers.

The categorization step assigns a category to an unclassified (preprocessed) text using the trained model currently residing in memory. The first task computes the similarity between the document and each category. How this similarity is computed, depends on the used classifier. NB for example computes the probability a given document belongs to a category. SVM and CVB compute similarity between a given document and a category by computing vector distances. When the similarities between a document and each category are computed, a second task selects the category having the highest similarity value. This task is independent of the used classifier, while for all similarity values holds, that the more similar a document is to a category, the higher the value.

Except the CVB algorithm, which we have implemented ourselves, we have used open-source and freely available algorithms for SVM (Johnson, 2008) and NB (Guenther, 2006). The SVM algorithm needs to be given values for some input parameters, which can be estimated using specific

methods. However, we have chosen to use the default parameter values of the SVM algorithm, to keep it as simple as possible. Finding optimal values for the parameters is a complex and time-consuming job and lies outside the scope of this research. The default values are:

SVM type:	C-SVC
SVM kernel:	RBF
Gamma parameter:	0
C parameter:	1

Section 2 – Experiment and results

To be able to answer the research questions (RQ1 + RQ2) as defined in chapter 3, section 1, an experiment has been performed using a prototype of the Clavis Verbum tool (CV tool) and pre-selected job-sheet data. The prototype of the CV tool implements the methodology as defined in chapter 4, section 1. First the setup of the experiment is given, followed by some characteristics of the used seed file. Then something is told about the measures used to analyze performance during the experiment. Finally the results will be given and analyzed.

1 Setup of the experiment

The prototype we have used, has been implemented using C#.NET technology. One can however implement the steps and tasks of the methodology, as described in chapter 4 of section 1, using any type of programming language. The choice for C#.NET was merely because of familiarity with this technology and good integration with Microsoft Access. The latter was important, because the input seed files are Microsoft Access files. More details of the tool can be found in the user manual of the prototype, which can be found in Appendix C.

For the setup of the experiment, it is important to know that the prototype of the CV tool contains two main parts; one part for handling the pre-processing tasks (A in Figure 6) and one part for handling the classification and classifying (B and C in Figure 6). For the experiment we only used one standard seed file containing job-sheet data, as input for the part handling the pre-processing. The part handling the classification and classifying, takes as input an output file of the pre-processing part. We did not use any input files containing unclassified data, as will be clear later on. Details about the format of input and output files can be found in the manual of the CV tool (Appendix C). Details about the standard seed file will be given below.

1.1 Characteristics seed file

The seed file used for the experiment is created by a domain-expert and is manually checked on correctness (does each job-sheet contain the correct category). Manually creating and checking a seed file is a time-consuming task. Therefore the seed file is not that big, only 800 job-sheets categorized into four different categories, but useful enough for our experiment. Table 4 contains the number of job-sheets per category in the seed file. As one can see, the number of job-sheets is not equally divided among the four categories. This is not a problem, because job-sheets belonging to a smaller category might contain more (distinctive) words. In case results for a category are biased extremely, Table 4 might help in explaining it.

# of job-sheets	Category
197	Interoperability
265	IQ-settings
208	Tafel
132	Xtravision

Table 4: # of job-sheets per category in the standard seed file

All the job-sheets in the seed file are in English and each job-sheet has exactly one category. This is according to the earlier defined constraints in chapter 3 of section 1. Due to privacy, no parts of the seed file can be included in this document unfortunately. An example of a seed file can be found in the manual of the CV tool (Appendix C).

1.2 Setup details

For answering RQ1, we compared the use of a simple character-based feature selection method with the use of a more sophisticated feature selection method. In our methodology we defined three levels of sophistication for feature selection, as can be seen in Figure 11. Level A in this figure is the most sophisticated level, level B is the level of sophistication we are most interested in (n-grams) and level C is the least sophisticated level which is used as a base line. Using the pre-processing part of the CV tool (which performs all the steps of A in Figure 6), we could create at least three different pre-processed files out of the standard seed file: one for each feature selection level. For feature selection level B we did not know however which value to take for n. In chapter 4 of section 1 we already referenced to literature indicating a value of 5 for n is optimal. We needed to check that, so multiple pre-processed files where created with n ranging from 2 to 8. A value of 1 seemed not to be logical, because you then only retrieve single characters. A value higher than 8 seemed also not to be logical, because than you cannot retrieve a lot of small words correctly and according to the referenced literature the value should be around 5. Only in case results where best or almost best for n having a value of 8, we should have created more pre-processed files with n having larger values. The results will later on show that this was not needed.

We kept settings for the cleaning tasks the same for all three levels of feature selection, so for all levels the standard defined punctuation characters (punc_1) and stop-words (Appendix A) were removed. The term weighting was done using TF-IDF for all levels.

Using the pre-processed files for classification and categorizing, we could see which level of feature selection delivers the best results. To get results for answering RQ2, we needed to perform different levels of classification and categorizing. This could be nicely done using the pre-processed seed files. While we have three levels of classification and categorizing (Figure 19 and Figure 20), we got three times the number of pre-processed seed files as output. This gave us enough results to analyze, which can be found in chapter 3 of this section.

In the next chapter the used method for measuring the performance of classification and categorizing is explained.

2 Measurement

A well-known method to validate the performance of a classification technique is called “k-fold validation”, which is also the validation method we have used. For k-fold validation only a pre-processed seed file is needed, which is a pre-processed file where the belonging category for each text is included. By defining a value for k (whole positive integer greater than 2), the seed file will be divided in k parts. Using those k parts, an equal number of k runs will be performed. For each run, one of the k parts will be used as a test set and all the other k-1 parts will be used as a training set. Each one of the k parts needs to be test set once, which is why there are k runs. A training set is used for classification, so for training a model. Based on this model, items in a test set can be categorized. We have used a value of 10 for k, which leads to test sets of 80 job-sheets and 10 runs per pre-processed file, which seems reasonable.

Each run outputs a categorized test set. By comparing the categorized category with the actual category for each item in the test set, a performance measure can be computed. A simple measure could be the percentage of the correctly classified items in the test set for a given category. We have used three measures which are well-known in the area of Machine Learning (ML): Precision, Recall and F1. These three measures are automatically computed by the CV tool and placed in the output file of the classification and categorizing part. Below the definitions are given for these three measures, where relevant items are those items actual belonging to category c and retrieved items are those items categorized as category c.

$$Precision(c) = \frac{|\{relevant\ items\} \cap \{retrieved\ items\}|}{|\{retrieved\ items\}|}$$

$$Recall(c) = \frac{|\{relevant\ items\} \cap \{retrieved\ items\}|}{|\{relevant\ items\}|}$$

$$F1(c) = 2 * \frac{precision(c) * recall(c)}{precision(c) + recall(c)}$$

To totalize the values for a specific measure and category for all k runs, the average of all k measurement values could be computed. The computation of this average can be done in two different ways: micro-averaging and macro-averaging. Micro-averaging means, that all the classified items of all the runs are put in one giant set and measurement values are computed for this whole set. Macro-averaging means, that for a certain measure, all k set-values are added first and then divided by the number of sets k. We have computed both total values, which are actually also computed by the CV tool and placed into the output file.

The totalized precision, recall and F1 measure for all categories together is computed by adding all values of a given measure and divide it by the number of categories. This can be expressed as follow, where C is the collection of predefined categories:

$$Precision = \frac{\sum_{c \in C} Precision(c)}{|C|}$$

$$Recall = \frac{\sum_{c \in C} Recall(c)}{|C|}$$

$$F1 = \frac{\sum_{c \in C} F1(c)}{|C|}$$

3 Results and analysis

Before we are going to analyze the performance of the used techniques, we first show that a value of 5 for n can be accepted as a good value for the n-gram feature selection. We have created seven pre-processed seed files using the n-gram feature selection, each with a different value for n, ranging from 2 to 8. These pre-processed files have been validated using the k-fold validation (k=10) and two different classifiers, which are Naïve Bayes (NB) and Concept Vector Based (CVB). Figure 21 shows the macro-averaged totalized F1 measures for given n in a separate graph for each of the two classifiers. Table 5 contains the exact values of Figure 21.

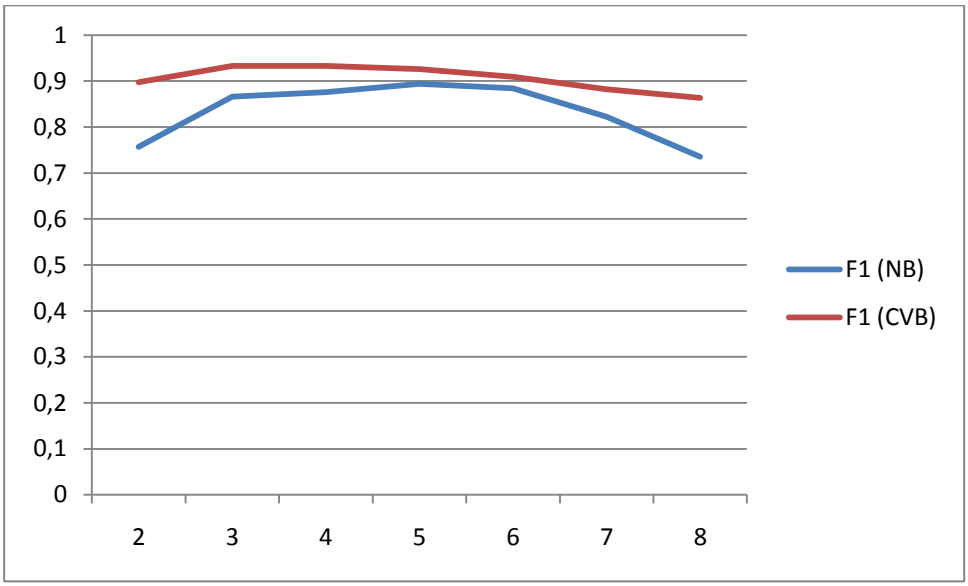


Figure 21: Macro-averaged totalized F1 measures for different values of n (n-gram feature selection) using NB and CVB classifiers.

n	F1 (NB)	F1 (CVB)
2	0,756558	0,897321
3	0,865777	0,932663
4	0,875546	0,932955
5	0,893701	0,925562
6	0,8844	0,909244
7	0,822012	0,882006
8	0,735428	0,863054

Table 5: Macro-averaged totalized F1 measures for NB and CVB classifiers, given a specific value for n.

The macro-averaged totalized F1 measures in the previous table and figure indicate indeed that a value of 5 for n is very good. For NB it even turns out to be the best value. For CVB a value of 4 for n seems to be best, but because the F1 measure for n=5 is very close to the F1 measure for n=4, 5 is also acceptable as best value. Together with the results in referenced literature, as stated in chapter 4 of section 1, we also assume this value (close) to be best for other classification techniques like SVM. In the remainder of the experiment, only the pre-processed seed file containing 5-grams is used, out of the pre-processed files using n-gram feature selection.

The following figures give an overview of the macro-averaged performances of the different classification techniques in combination with the used feature selection options during pre-processing. In these figures, NLP is level A, 5-grams is level B and Raw tokens is level C, as defined in Figure 11. All results are retrieved using k-fold validation for k=10.

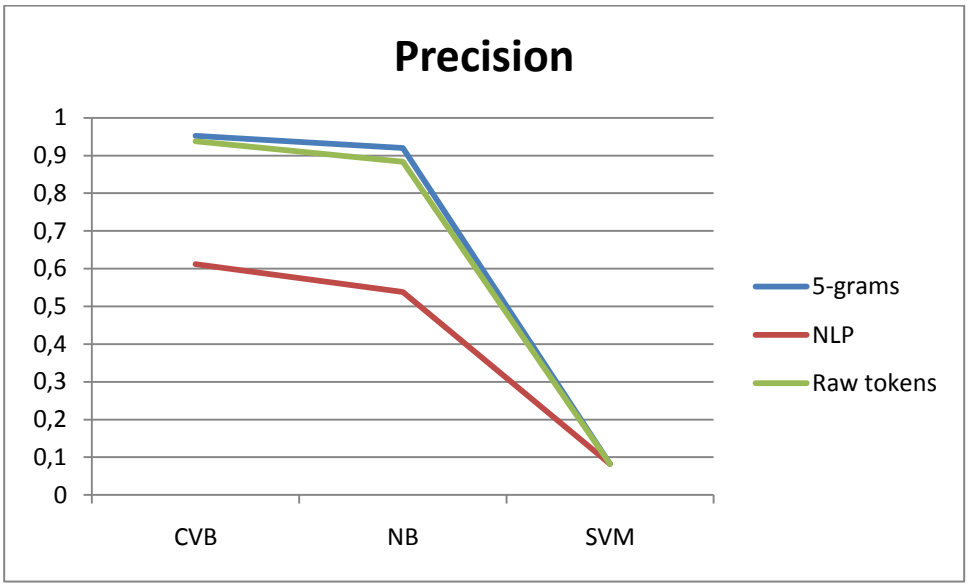


Figure 22: Macro-averaged totalized precisions of all combinations of feature selection type and classification technique.

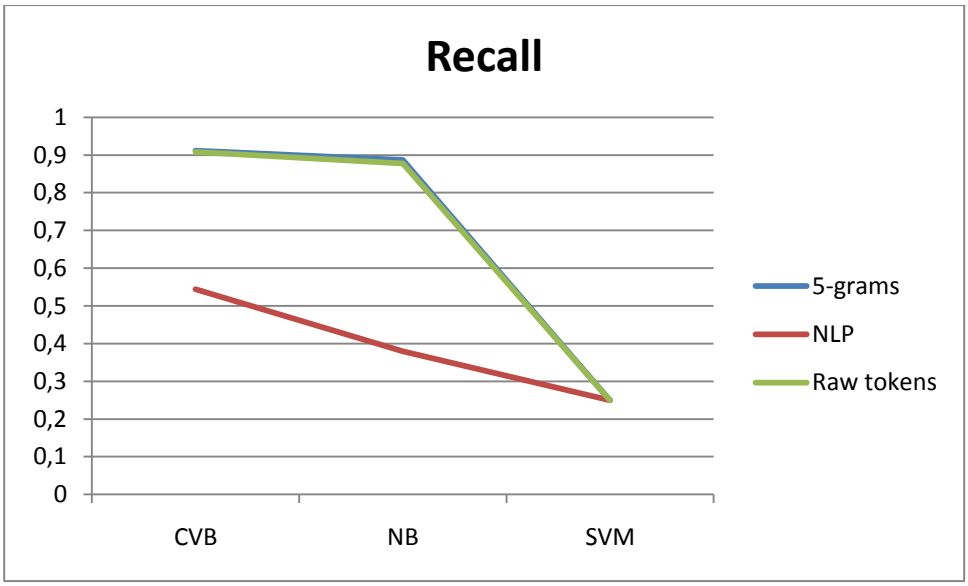


Figure 23: Macro-averaged totalized recalls of all combinations of feature selection type and classification technique.

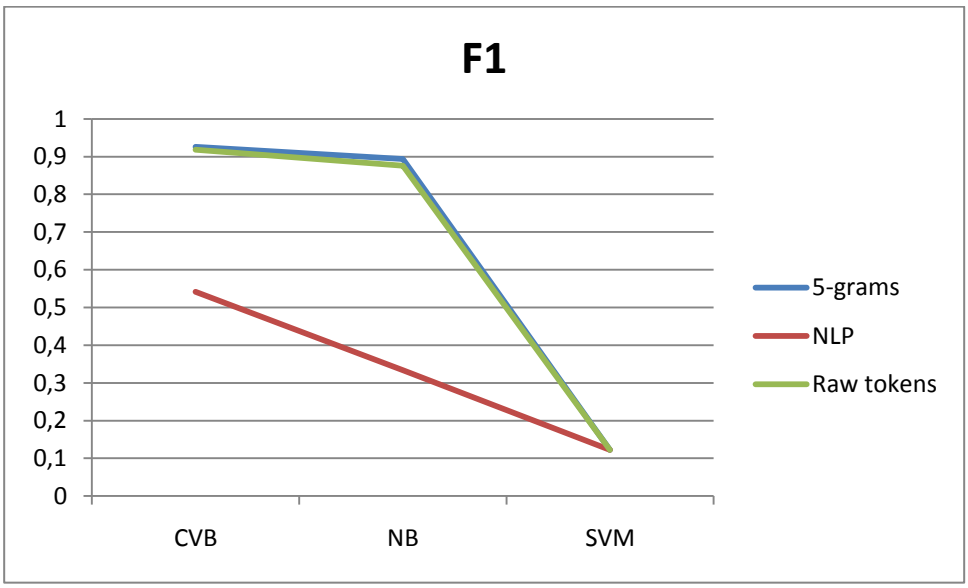


Figure 24: Macro-averaged totalized F1 measures of all combinations of feature selection type and classification technique.

If we look to all three figures above, a couple of things stand out. Most remarkable to see is the very bad performance of the SVM classifier. However, something strange has happened here. If we look closer to the results of the SVM classifier, we found that for all type of feature selections used, the same performance values occur (Table 6). If we look to the figures in the table below, we can see that only one category has been

classified. This is very strange, the more because it occurs independently from the used feature selection. Two causes can be given for this to happen. The first and most simple cause is that the algorithm does not work correctly. All input data is given correctly to the algorithm, so it might be a problem internal. A second cause can be that the standard parameters used for the classifier are not suitable or good enough for the type of data (all weighted text). The working of the SVM classifier is however outside the scope of this research, so this needs to be investigated in a follow up. For the remainder of this chapter we omit the results of the SVM classifier and only look at the results of the NB and CVB classifiers, which seem to be more assumable.

#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0	0	0
10	Interoperability	0	0	0
10	IQ-settings	0,33125	1	0,491313045
10	Xtravision	0	0	0
TOTAL		0,0828125	0,25	0,122828261

Table 6: Results SVM classifier. These figures hold for all type of feature selections.

When we have another look at the performance figures (Figure 22, Figure 23, Figure 24), it stands out that 5-grams and raw tokens contribute to a much better performance than using NLP for feature selection. If we compare 5-grams and raw tokens, 5-grams seem to be slightly better than raw tokens. A third finding is the better performance of the CVB classifier against the NB classifier, among all feature selection types.

If we take a closer look to the classifiers NB and CVB, in combination with the feature selection types 5-grams and raw tokens (Table 7 to 10) we can see that the combination of classifier CVB with 5-grams has the best performance in total. It is also remarkable to see that the measurement values for almost every category are above 85% (Table 7), which is very good for such a simple feature selection technique and classifier. Besides, it is much better than the precision of the semi-automatic classification method used at the company, which was around 50% (section 1, chapter 1).

#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0,976304461	0,942258936	0,95819395
10	Interoperability	0,976714286	0,890287829	0,929469585
10	IQ-settings	0,856305762	0,9815781	0,913383405
10	Xtravision	1	0,831742424	0,901201489
TOTAL		0,952331127	0,911466822	0,925562107

Table 7: Results CVB classifier in combination with 5-grams.

#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0,943078359	0,923703186	0,931175844
10	Interoperability	0,927487632	0,982413943	0,953061363
10	IQ-settings	0,859486038	0,921084282	0,887027553
10	Xtravision	0,950899471	0,71962482	0,80353774
TOTAL		0,920237875	0,886706558	0,893700625

Table 8: Results NB classifier in combination with 5-grams.

#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0,978792735	0,933867327	0,95521928
10	Interoperability	1	0,900623385	0,946432349
10	IQ-settings	0,842763315	0,985925926	0,907349847
10	Xtravision	0,93	0,81215368	0,864395725
TOTAL		0,937889012	0,908142579	0,9183493

Table 9: Results CVB classifier in combination with raw tokens.

#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0,91759509	0,944332556	0,928748494
10	Interoperability	0,901330209	0,948799308	0,924044167
10	IQ-settings	0,894388791	0,878983167	0,884483169
10	Xtravision	0,821190476	0,736594517	0,767337189
TOTAL		0,883626141	0,877177387	0,876153255

Table 10: Results NB classifier in combination with raw tokens.

The full performance measures of all combinations of classifier and feature selection type, using 10-fold validation, can be found in Appendix B. In this appendix, also all micro-averaged results are included. These micro-averaged results do not differ a lot from the macro-averaged results.

In Table 11 are some processing times given for the NB and CVB classifier in combination with different feature selection types. Note while the used tool is just a prototype, it has not been optimized yet for optimal processing times. Some combinations of classifier and feature selection type might even be faster than another combination being faster in the figure below. However, it gives an indication of the speed differences between all the combinations. 5-gram turns out to be the slowest option for feature selection, but this seems to be logical because 5-gram generates far more features than NLP and raw tokens. Maybe remarkable to see, is the longer processing time for CVB+5-gram against NB+5-gram, because the CVB algorithm is faster than NB in combination with the other two feature selection types.

Combination	Processing times (seconds)
CVB + NLP	16
CVB + Token	20
NB + NLP	40
NB + Token	45
NB + 5-gram	115
CVB + 5-gram	147

Table 11: Processing times performing 10-fold validation on pre-processed standard seed file, for given combinations of classifier and feature selection type.

Section 3 - Conclusions and future work

In this chapter conclusions will be formulated using the retrieved results as given in the previous section. Also some recommendations for future work will be given.

1 Conclusions

We have arrived at the final part of this report and perhaps also the most important one. After all the research, experimenting and gained results, it is time to see if we can answer each of the two research questions. The research questions will be recapped and discussed one by one below.

RQ1: Contribute simple character-based feature selection methods to a better classification result than more advanced linguistics-based feature selection methods, in case of domain-specific texts like field service data

To answer this question, we have introduced three types of feature selection methods, one more sophisticated than another. The most simple type is the selection of raw tokens, which means no more than cutting a text into features on white-spaces. A little more sophisticated, but still character-based, is the selection of n-grams. The n-gram method takes sequences of characters from a text, having length n. The most sophisticated method (we called it shortly "NLP") uses linguistics to extract terms out of a text. Terms can be seen as words or phrases of words, which contains some meaningful or human-readable information.

During the experiment, we have seen that the character-based methods do their work quite well for field service data. The n-gram method, which we were most interested in, delivers the best results; much better than the more sophisticated NLP method and also better than the raw tokens method. That NLP would not be optimal to use, was a bit predicted, because of the challenges we introduced in chapter 2 of section 1. However, it is surprisingly to see how much better the n-gram method works, while it is less sophisticated. The n-gram method seems to be less sensitive for the challenges, like fuzzy text and domain specific information. N-gram just looks to characters, therefor typing errors have no influence. A typing error influences the recognition of a word by linguistic methods, but for n-grams the recognition of words does not play a role. The same holds for domain specific information, which is difficult to be recognized by linguistic techniques using standard dictionaries.

It was also important to see, that n-grams worked better than just raw tokens. This showed us, that n-grams is more than just splitting a text randomly into parts. The power of the n-grams against raw tokens, is the way n-grams are extracted from text. In chapter 4 this has been explained. Although n-grams are just characters and mostly not forming a meaningful word of phrase, due to the window-based extraction they contain some hidden information like the order of overlapping n-grams. Such information is not contained by raw tokens.

In case of the challenge about different languages, the n-grams also benefit from the property that only characters are dealt with. Linguistic techniques only are able to recognize words or phrases in a certain language, in case they use a dictionary and grammatical rules in that language. N-grams are language independent, because a letter "A" stays a letter "A". An interesting question arises for future work. It is about the applicability of n-grams to other character sets, like Arabic or Chinese. Chinese characters for example, can represent phrase or whole sentences. The question is if all the beneficial properties of using n-grams mentioned, still apply to these type of characters.

So, based on our results, the answer to this research question can be answered positively for field service data like job-sheets. To be sure that this technique also works best for other types of textual field service data (having the same properties as defined by the challenges), more tests using larger data sets need to be performed.

RQ2: In case simple character-based extraction methods are used, are simple classification techniques more suitable for classifying domain-specific texts like field service data than more advanced classification techniques

We had planned to use three different classification techniques during our experiment, each of a different level of sophistication. The most sophisticated one, Support Vector Machines (SVM), delivered not the results we were hoping for unfortunately. Analyzing these results, indicate that something is wrong in the used implementation of SVM or that parameters need to be set more specific.

The Concept-Vector Based (CVB) classifier, based on the information retrieval paradigm, is the most simple one and seemed to do best during our experiment. However, the second least sophisticated approach, Naïve Bayes (NB) is a close second. The close performance values of both classifiers give rise to another experiment, using more and larger datasets. Then, the processing times can be investigated a bit further also. Indications are given in the previous chapter, that CVB is slower than NB in case of using 5-grams. The question is, if this holds for all sizes of data and how much performance is lost in terms of processing time against performance in classification results.

The second research question is a bit more difficult to answer using our results. Firstly because the results of the most sophisticated classification technique are not strong enough, or even totally unusable. Secondly, because the other two techniques are very close in terms of classification performance. We still think that the most basic one, CVB, will do the job best in case of using n-grams. It does work better than NB, though it is a little bit. It can do better than other more sophisticated stuff in combination of n-grams, because n-grams can produce an enormous set of different features. A high value of different features can cause trouble to the more sophisticated techniques, at least in terms of processing time. This is just hypothetical speaking. More research needs to be done using sophisticated classification techniques in combination with n-grams.

References

Burges, C.J.C. 1998. *A tutorial on support vector machines for pattern recognition.* s.l. : Data Mining and Knowledge Discovery 2, 1998. pp. 121-167.

Cavnar, W.B. and Trenkle, J.M. *N-Gram-Based Text Categorization.* s.l. : Environmental Research Institute of Michigan, Cambridge, MA.

Dumais, S.T., et al. 1998. *Inductive learning algorithms and representations for text categorization.* s.l. : Proceedings of CIKM-98, 7th ACM International Conference on Information Knowledge Management, 1998. pp. 148-155.

Fahmi, I. 2009. *Automatic term and relation extraction for medical question answering system.* Groningen : s.n., 2009.

Franken, B.F. and Hendriks, M.M. 2002. *Fast Field Feedback: a new feedback procedure.* Eindhoven : Eindhoven University of Technology, 2002.

Guenther, E. 2006. *Naive Bayes Algorithm written in C#.NET.* s.l. : erich_guenther@hotmail.com, 2006.

Heynen, E.W.H.P. 2002. *Fast Field Feedback: a field study.* Eindhoven : Eindhoven University of Technology, 2002.

Hull, D. 1996. *Stemming algorithms - A case study for detailed evaluation.* s.l. : JASIS 47 (1), 1996. pp. 70-84.

Jain, A.K., Murty, M.N. and Flynn, P.J. 1999. *Data Clustering: A Review.* s.l. : ACM Computing Surveys, Vol. 31 (3), 1999. pp. 264-323.

Joachims, T. 1998. *Text categorization with support vector machines: learning with many relevant features.* s.l. : Proceedings of ECML-98, 10th European Conference on Machine Learning, 1998. pp. 137-142.

Joachims, T. and Sebastiani, F. 2002. *Guest editors' introduction to the special issue on automated text categorization.* s.l. : J. intell, Inform Syst Vol. 18, 2002. pp. 103-105.

Johnson, M. 2008. *SVM.NET: A C#.NET library of SVM classifiers.* s.l. : http://www.matthewajohnson.org/software/svm.html, 2008.

Jurafsky, D. and Martin, J.H. 2008. *Speech and Language Processing: International Version, 2nd Edition.* s.l. : Pearson Higher Education, 2008. 0135041961.

Kim, Y.H., Hahn, S.Y. and Zhang, B.T. 2000. *Text filtering by boosting naive Bayes classifiers.* s.l. : Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval, 2000. pp. 168-175.

Lewis, D.D. and Hayes, P.J. 1994. *Guest editorial for the special issue on text categorization.* s.l. : ACM, Trans. Inform. Syst. Vol. 12 (3), 1994. p. 231.

Lewis, D.D. 1998. *Naive (Bayes) at forty, The independence assumption in Information Retrieval.* s.l. : Proceedings of ECML-98, 10th European Conference on Machine Learning, 1998. pp. 4-15.

Manning, C D, Raghavan, P and Schütze, H. 2008. *Introduction to Information Retrieval.* Cambridge : Cambridge University Press., 2008.

Manning, C. and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing.* s.l. : MIT Press, Cambridge, MA, 1999.

Miller, G.A. and al., et. *WordNet: a lexical database for English.* s.l. : Princeton University. p. http://wordnet.princeton.edu.

Northedge, R.J. *SharpNLP: C#.NET implementation of NLP tools using the WordNet database.* s.l. : http://sharpnlp.codeplex.com.

Petkove, P.T. 2003. *An Analysis of Field Feedback in Consumer Electronic Industry.* Eindhoven : Eindhoven University of Technology, 2003. ISBN 90-386-1758-5.

Rahmoun, A and Elberrichi, Z. 2006. *Experimenting N-Grams in Text Categorization.* s.l. : ACM, Vol. 3, 2006. pp. 50-62.

Roth, D. 1998. *Learning to resolve natural language ambiguities: a unified approach.* s.l. : Proceedings of AAAI-98, 15th Conference of the American Association for Artificial IntelligenceTrans, 1998. pp. 806-813.

Sebastiani, F. 2002. *Machine Learning in Automated Text Categorization.* s.l. : ACM Computing Surveys, Vol. 34 (1), 2002. pp. 1-47.

Yang, Y and Liu, X. 1999. *a re-examination of text categorization methods.* s.l. : Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, 1999. pp. 42-49.

A. Stop-word List

a	all	anyways	because	briefly	d	eighty
able	almost	anywhere	become	but	date	either
about	alone	apparently	becomes	by	did	else
above	along	approximately	becoming	c	didnt	elsewhere
abst	already	are	been	ca	different	end
accordance	also	aren	before	came	do	ending
according	although	arent	beforehand	can	does	enough
accordingly	always	arise	begin	cannot	doesnt	especially
across	am	around	beginning	cant	doing	et
act	among	as	beginnings	cause	done	et-al
actually	amongst	aside	begins	causes	dont	etc
added	an	ask	behind	certain	down	even
adj	and	asking	being	certainly	downwards	ever
adopted	announce	at	believe	co	due	every
affected	another	auth	below	com	during	everybody
affecting	any	available	beside	come	e	everyone
affects	anybody	away	besides	comes	each	everything
after	anyhow	awfully	between	contain	ed	everywhere
afterwards	anymore	b	beyond	containing	edu	ex
again	anyone	back	biol	contains	effect	except
against	anything	be	both	could	eg	f
ah	anyway	became	brief	couldnt	eight	far

few	getting	her	id	it'll	latterly	may
ff	give	here	ie	its	least	maybe
fifth	given	hereafter	if	itself	less	me
first	gives	hereby	i'll	i've	lest	mean
five	giving	herein	im	j	let	means
fix	go	heres	immediate	just	lets	meantime
followed	goes	hereupon	immediately	k	like	meanwhile
following	gone	hers	importance	keep	liked	merely
follows	got	herself	important	keeps	likely	mg
for	gotten	hes	in	kept	line	might
former	h	hi	inc	keys	little	million
formerly	had	hid	indeed	kg	'll	miss
forth	happens	him	index	km	look	ml
found	hardly	himself	information	know	looking	more
four	has	his	instead	known	looks	moreover
from	hasnt	hither	into	knows	ltd	most
further	have	home	invention	l	m	mostly
furthermore	havent	how	inward	largely	made	mr
g	having	howbeit	is	last	mainly	mrs
gave	he	however	isnt	lately	make	much
get	hed	hundred	it	later	makes	mug
gets	hence	i	itd	latter	many	must

my	nobody	okay	owing	primarily	regarding	seeing
myself	non	old	own	probably	regardless	seem
n	none	omitted	p	promptly	regards	seemed
na	nonetheless	on	page	proud	related	seeming
name	noone	once	pages	provides	relatively	seems
namely	nor	one	part	put	research	seen
nay	normally	ones	particular	q	respectively	self
nd	nos	only	particularly	que	resulted	selves
near	not	onto	past	quickly	resulting	sent
nearly	noted	or	per	quite	results	seven
necessarily	nothing	ord	perhaps	qv	right	several
necessary	now	other	placed	r	run	shall
need	nowhere	others	please	ran	s	she
needs	o	otherwise	plus	rather	said	shed
neither	obtain	ought	poorly	rd	same	she'll
never	obtained	our	possible	re	saw	shes
nevertheless	obviously	ours	possibly	readily	say	should
new	of	ourselves	potentially	really	saying	shouldnt
next	off	out	pp	recent	says	show
nine	often	outside	predominantly	recently	sec	showed
ninety	oh	over	present	ref	section	shown
no	ok	overall	previously	refs	see	showns

shows	specified	th	there'll	throughout	unfortunately	very
significant	specify	than	thereof	thru	unless	via
significantly	specifying	thank	therere	thus	unlike	viz
similar	state	thanks	theres	til	unlikely	vol
similarly	states	thanx	thereto	tip	until	vols
since	still	that	thereupon	to	unto	vs
six	stop	that'll	there've	together	up	w
slightly	strongly	thats	these	too	upon	want
so	sub	that've	they	took	ups	wants
some	substantially	the	theyd	toward	us	was
somebody	successfully	their	they'll	towards	use	wasnt
somehow	such	theirs	theyre	tried	used	way
someone	sufficiently	them	they've	tries	useful	we
somethan	suggest	themselves	think	truly	usefully	wed
something	sup	then	this	try	usefulness	welcome
sometime	sure	thence	those	trying	uses	we'll
sometimes	t	there	thou	ts	using	went
somewhat	take	thereafter	though	twice	usually	were
somewhere	taken	thereby	thoughh	two	v	werent
soon	taking	thered	thousand	u	value	we've
sorry	tell	therefore	throug	un	various	what
specifically	tends	therein	through	under	've	whatever

what'll	who'll	you
whats	whom	youd
when	whomever	you'll
whence	whos	your
whenever	whose	youre
where	why	yours
whereafter	widely	yourself
whereas	willing	yourselves
whereby	wish	you've
wherein	with	z
wheres	within	zero
whereupon	without	polaris
wherever	wont	paperless
whether	words	clarify
which	world	onestar
while	would	
whim	wouldnt	
whither	www	
who	x	
whod	y	
whoever	yes	
whole	yet	

B. 10-fold validation results

5-gram + CVB				
#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0,976304461	0,942258936	0,95819395
10	Interoperability	0,976714286	0,890287829	0,929469585
10	IQ-settings	0,856305762	0,9815781	0,913383405
10	Xtravision	1	0,831742424	0,901201489
TOTAL		0,952331127	0,911466822	0,925562107
#runs	Cat	Precision_Micro	Recall_Micro	F1_Micro
10	Tafel	0,97029703	0,942307692	0,956097561
10	Interoperability	0,977653631	0,892857143	0,933333333
10	IQ-settings	0,855263158	0,981132075	0,913884007
10	Xtravision	1	0,877862595	0,93495935
TOTAL		0,950803455	0,923539877	0,934568563

5-gram + NB				
#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0,943078359	0,923703186	0,931175844
10	Interoperability	0,927487632	0,982413943	0,953061363
10	IQ-settings	0,859486038	0,921084282	0,887027553
10	Xtravision	0,950899471	0,71962482	0,80353774
TOTAL		0,920237875	0,886706558	0,893700625
#runs	Cat	Precision_Micro	Recall_Micro	F1_Micro
10	Tafel	0,941747573	0,932692308	0,937198068
10	Interoperability	0,927536232	0,979591837	0,952853598
10	IQ-settings	0,871428571	0,920754717	0,895412844
10	Xtravision	0,962616822	0,786259542	0,865546218
TOTAL		0,9258323	0,904824601	0,912752682

5-gram + SVM				
#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0	0	0
10	Interoperability	0	0	0
10	IQ-settings	0,33125	1	0,491313045

10	Xtravision	0	0	0
TOTAL		0,0828125	0,25	0,122828261
#runs	Cat	Precision_Micro	Recall_Micro	F1_Micro
10	Tafel	0	0	0
10	Interoperability	0	0	0
10	IQ-settings	0,33125	1	0,497652582
10	Xtravision	0	0	0
TOTAL		0,0828125	0,25	0,124413146

NLP + CVB				
#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0,696040626	0,581056527	0,629027903
10	Interoperability	0,634874154	0,57833306	0,594680678
10	IQ-settings	0,550528405	0,776115201	0,639521441
10	Xtravision	0,565833333	0,241507937	0,303984962
TOTAL		0,611819129	0,544253181	0,541803746
#runs	Cat	Precision_Micro	Recall_Micro	F1_Micro
10	Tafel	0,697142857	0,586538462	0,637075718
10	Interoperability	0,638888889	0,586734694	0,611702128
10	IQ-settings	0,550135501	0,766037736	0,640378549
10	Xtravision	0,607843137	0,236641221	0,340659341
TOTAL		0,623502596	0,543988028	0,557453934

NLP + NB				
#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0,700571096	0,33550136	0,445518496
10	Interoperability	0,608616522	0,241287132	0,333053767
10	IQ-settings	0,662175325	0,183636807	0,284218373
10	Xtravision	0,181809953	0,756525974	0,273389176
TOTAL		0,538293224	0,379237818	0,334044953
#runs	Cat	Precision_Micro	Recall_Micro	F1_Micro
10	Tafel	0,708333333	0,326923077	0,447368421
10	Interoperability	0,590361446	0,25	0,35125448
10	IQ-settings	0,675675676	0,188679245	0,294985251
10	Xtravision	0,18464351	0,770992366	0,297935103
TOTAL		0,539753491	0,384148672	0,347885814

NLP + SVM				
#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0	0	0
10	Interoperability	0	0	0
10	IQ-settings	0,33125	1	0,491313045
10	Xtravision	0	0	0
TOTAL		0,0828125	0,25	0,122828261
#runs	Cat	Precision_Micro	Recall_Micro	F1_Micro
10	Tafel	0	0	0
10	Interoperability	0	0	0
10	IQ-settings	0,33125	1	0,497652582
10	Xtravision	0	0	0
TOTAL		0,0828125	0,25	0,124413146

Raw tokens + CVB				
#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0,978792735	0,933867327	0,95521928
10	Interoperability	1	0,900623385	0,946432349
10	IQ-settings	0,842763315	0,985925926	0,907349847
10	Xtravision	0,93	0,81215368	0,864395725
TOTAL		0,937889012	0,908142579	0,9183493
#runs	Cat	Precision_Micro	Recall_Micro	F1_Micro
10	Tafel	0,97979798	0,932692308	0,955665025
10	Interoperability	1	0,903061224	0,949061662
10	IQ-settings	0,844660194	0,98490566	0,909407666
10	Xtravision	0,974137931	0,86259542	0,914979757
TOTAL		0,949649026	0,920813653	0,932278527

Raw tokens + NB				
#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0,91759509	0,944332556	0,928748494
10	Interoperability	0,901330209	0,948799308	0,924044167
10	IQ-settings	0,894388791	0,878983167	0,884483169
10	Xtravision	0,821190476	0,736594517	0,767337189
TOTAL		0,883626141	0,877177387	0,876153255
#runs	Cat	Precision_Micro	Recall_Micro	F1_Micro

10	Tafel	0,911627907	0,942307692	0,926713948
10	Interoperability	0,902912621	0,948979592	0,925373134
10	IQ-settings	0,8996139	0,879245283	0,889312977
10	Xtravision	0,875	0,801526718	0,836653386
TOTAL		0,897288607	0,893014821	0,894513361

Raw tokens + SVM				
#runs	Cat	Precision_Macro	Recall_Macro	F1_Macro
10	Tafel	0	0	0
10	Interoperability	0	0	0
10	IQ-settings	0,33125	1	0,491313045
10	Xtravision	0	0	0
TOTAL		0,0828125	0,25	0,122828261
#runs	Cat	Precision_Micro	Recall_Micro	F1_Micro
10	Tafel	0	0	0
10	Interoperability	0	0	0
10	IQ-settings	0,33125	1	0,497652582
10	Xtravision	0	0	0
TOTAL		0,0828125	0,25	0,124413146

C. User Manual prototype CV tool

1. Introduction

This document describes shortly how to use beta-1.0 of the Clavis Verbum (CV) tool. The CV tool has been developed for the Customer Service Data Analysis department of the iXR division at Philips Healthcare Best and automates the process of classifying job-sheets as contained by the Masterlist. In this context classification needs to be seen as assigning each job-sheet a category out of a set of pre-defined categories. This can be done manually, by reading each job-sheet line by line, but this takes a lot of time. The CV tool automates the assignment of pre-defined categories to job-sheets (Figure 25), by learning from a so called seed file: a file containing manually classified job-sheets. Once the CV tool has learned from a seed file (1), it can classify unclassified job-sheets (2) automatically. The CV tool contains different techniques to learn from a seed file and to classify unclassified job-sheets.

Beta-1.0 of the CV tool is just a prototype, which means that the user interface and the performance are not optimal yet. The use of this version of the tool is also limited to English textual data and it can classify each job-sheet only to one predefined category. Except for those constraints, for a couple of techniques the classification of job-sheets functions fully. More information about classification in general and the used classification techniques in beta-1.0 of the CV tool, can be found in the thesis “Classification of field service data using n-grams” by M.P.E Möllenbeck (TU/e, 2009).

* *Appendix C contains a quick manual to be up and running in just a few minutes, without needing advanced settings.*

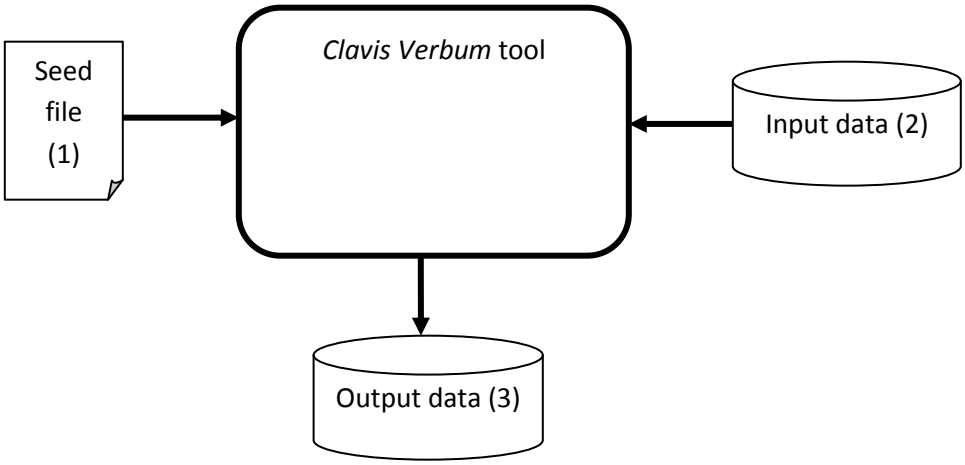


Figure 25

2. Installation

Prerequisites

The CV tool can only be installed on a computer with Microsoft Windows XP or higher installed on it. Any other operating system is not supported yet. It is preferable to have Microsoft Access 2007 or higher installed before installing the CV tool, because all the input files used and all the output file the CV tool generates are Microsoft Access 2007 files.

Installing

The installation directory[1] of the CV tool contains the following items:

- File “ClavisVerbum_Installer.msi”
- File “cvtool_setup.exe”
- Directory “Office2007PIARedist” containing:
    - File “ o2007PIA.msi”

The installation of the CV tool can be started by double-clicking the setup file “cvtool_setup.exe”. Follow the instructions on screen to install the tool on the computer. Once installed a shortcut on the desktop and an entry in the Start Menu are created to start the CV-tool. Double-click the CV-tool icon on the desktop or in the CV-tool entry of the Start Menu to start the tool.

Uninstalling

To uninstall the CV tool, double-click the uninstaller in the CV-tool entry of the Start Menu and follow the instructions on the screen.

[1] *In case the installation directory is compressed into a zip-file, the zip-file needs to be extracted first to a temporary location to make the installation directory visible. Opening the setup file from within the zip-file leads to a failing installation.*

3. Main Screen

The first screen that shows up after starting the CV tool is the main screen (Figure 26). The main screen shows the logo of the CV tool, the current version of the tool left-below and a navigation panel on the right side. The navigation panel contains two blue text links: “Pre-process Input” and “Classification”. By clicking on one of them, the corresponding screen will be shown. These two screens will be discussed in the following chapters of this manual. For now it is important to know that classification can take place only using pre-processed data. This holds for all input data: seed files and unclassified job-sheets.

The CV tool can be closed at any time by clicking the red button at the upper-right corner.

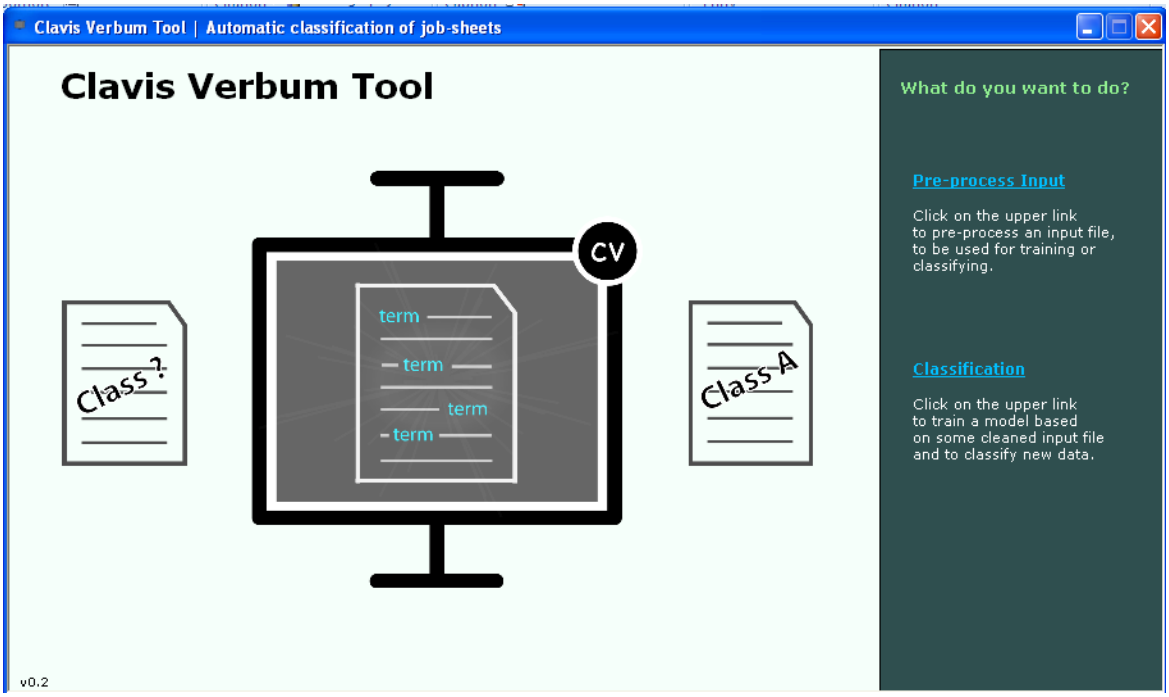


Figure 26

Before we discuss the other sections of the tool, it is important to know how the process of classification looks like using the CV tool. Figure 27 shows a simple process diagram of the classification process using the CV tool. It can be seen that first a seed file needs to be preprocessed in the preprocess section. After that a file containing unclassified job-sheets need to be preprocessed. It is important that exactly the same settings are used for preprocessing the “unknown jobsheets file” as used for preprocessing the seed file, so the type of preprocessed information will be the same. Finally a preprocessed “unknown jobsheets file’ can be classified using a preprocessed seed file. Later on in this document we will see that It is also possible to use only a seed file during classification.

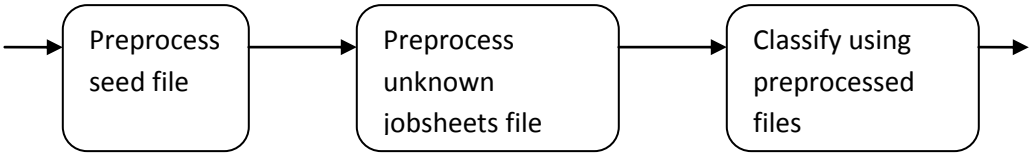

Preprocess seed file
Preprocess unknown jobsheets file
Classify using preprocessed files

Figure 27

4. Pre-process Input section

The pre-process input section of the CV tool is meant to pre-process all input data to be used at the classification section. The initial screen of this section can be seen in

Figure 28. The left side of the screen is the working area, which contains all the buttons and settings. The right side of the screen contains all the steps of the preprocessing process, where the current step is marked by a grey bar. When a step has been completed, its checkbox will be checked.

First the type of input file needs to be selected, which is by default 'CV Seed file'. In case a file with unclassified job-sheets needs to be preprocessed, 'Unknown Jobsheets file' is selected. Each type of input file needs to be in a specific format, which can be found in Appendix A. Note that for classifying an 'Unknown Jobsheets file' using a certain ' CV Seed file' , both files need to be preprocessed using the same settings! Otherwise the classification will not be optimal.

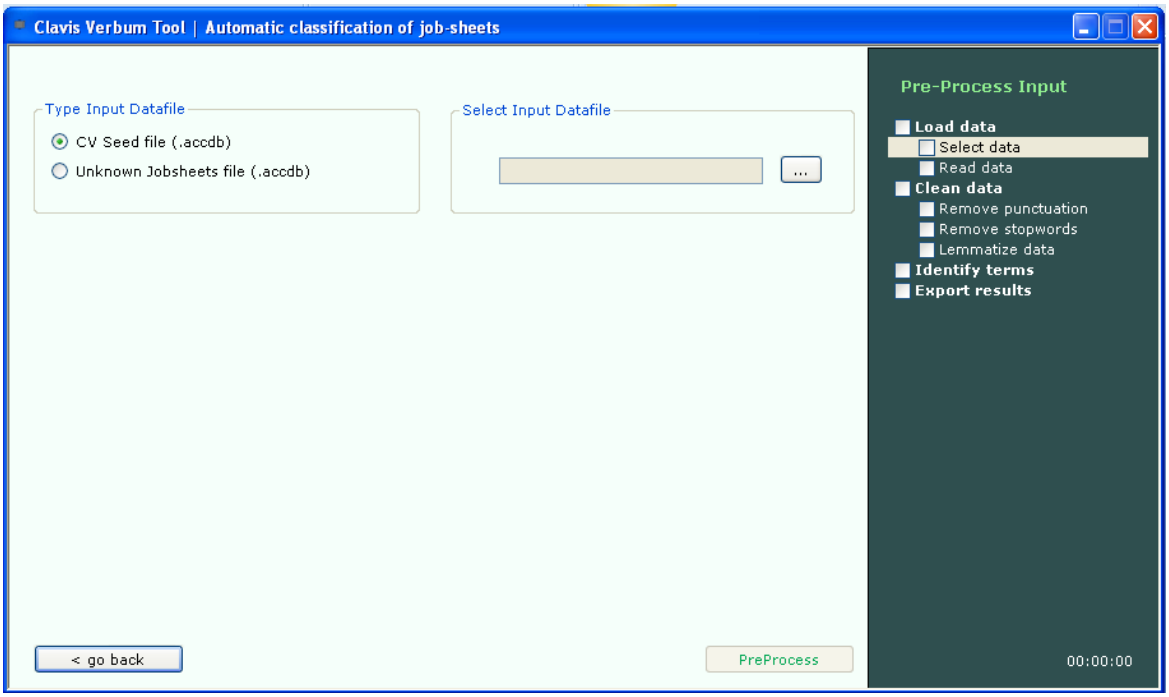


Figure 28

The second step in this section is selecting the actual input file to classify (remember that the input file needs to be in a specific format as given in Appendix A! No checkup will take place). Once a file is selected, new configuration options will show up (Figure 29).

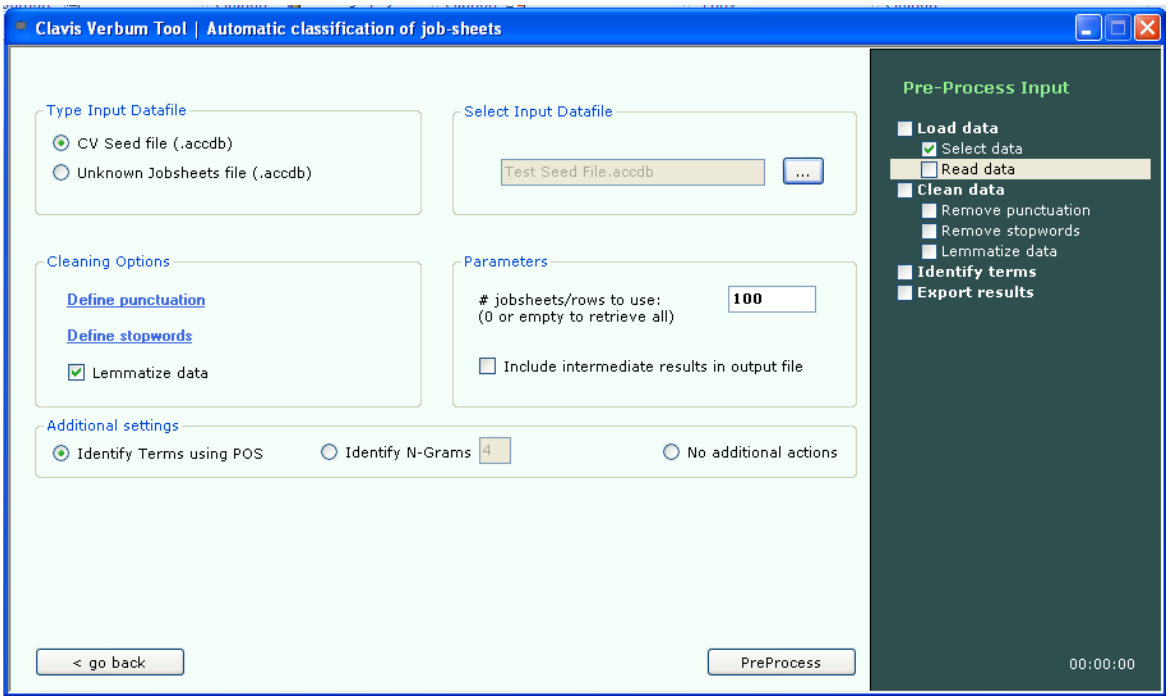


Figure 29

Cleaning options

The cleaning options specify which irrelevant data needs to be cleaned out of the input data, such as punctuation characters and stopwords (common general words not needed for classification). By clicking on the textual link “Define punctuation”, a popup window will show up with the currently defined punctuation characters to be removed from the input data (Figure 30). Defining more characters to be removed, can be done by adding them to the end of line in the input box, without any white space characters between them. Removing characters can be done by just deleting them from the input box. Default settings can be restored by clicking on “Reset to default”. Once clicked on “Save settings”, the currently typed characters will be stored to be removed from the input data. By clicking on the red button in the upper-right corner, all modifications are cancelled and are not stored.

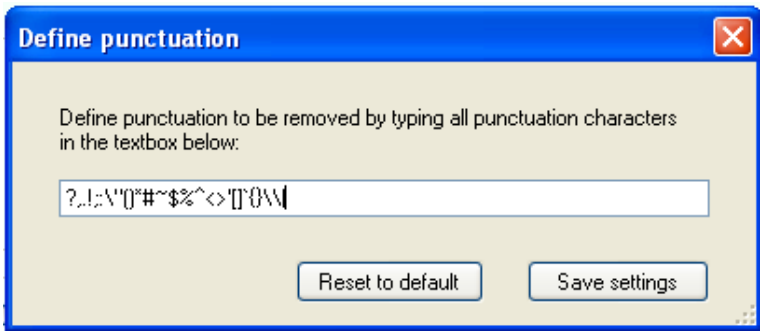


Figure 30

The stopword list can be viewed and changed by clicking on the textual link “Define stopwords”, which will raise a popup window (Figure 31). More stopwords to be removed can be added by typing each new stopword on a separate line without any white-spaces. A stopword can simply be removed by deleting the complete line in the list, so don’t leave any empty lines! Default settings can be restored by clicking on “Reset to default”. Once clicked on “Save settings”, the current list of stopwords as shown will be stored to be removed from the input data. By clicking on the red button in the upper-right corner, all modifications are cancelled and are not stored.

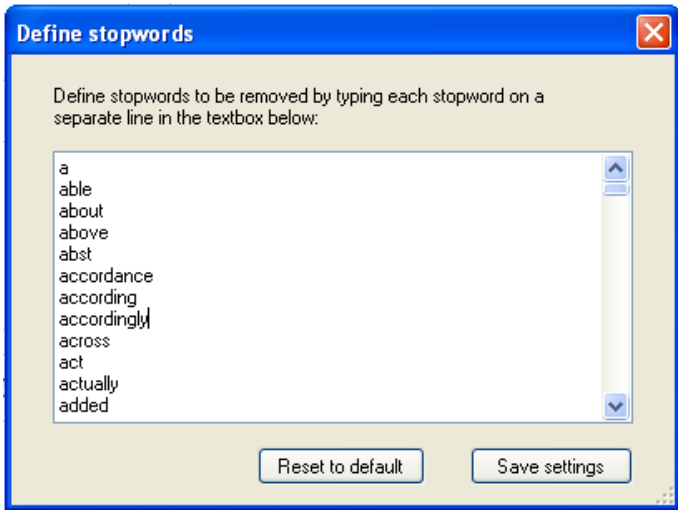


Figure 31

One more option is available at the cleaning options, which is the checkbox “Lemmatize data”. By default this option is disabled[2], but once activated, the tool tries to replace all verbs, adverbs and nouns by its root form using a dictionary. This option is most effective when “Identify Terms using POS” is selected at the additional settings.

Additional settings

The additional settings define the type of terms to be retrieved from the input data, which is the actual data to be used during classification. The retrieval of terms will take place after all the defined cleaning has taken place. The default setting is “Identify N-Grams” with value 5, which delivers best results in case of the job-sheets. The N-Gram value defines the character-size of each term to be retrieved. So a value of 5 means that character-sequences of precisely 5 characters are retrieved from the input data. More technical information can be found in the documentation mentioned in the introduction.

[2] *In case this option is disabled, the tool still performs some basic tasks which are part of the lemmatizer. This will be visible during execution of the preprocessing process by showing some progress during lemmatizing.*

The other two options of the additional settings are “Identify Terms using POS” and “No additional actions”. Using the option “Identify Terms using POS”, phrases of nouns and verbs are retrieved as multi-word terms using a dictionary. In case the option “No additional actions” is selected, just all words of the input are retrieved, which are all words separated by a white space in the given input.

Parameters

The parameters are settings which do not impact the result of the pre-processing directly. The first parameter to be set is the number of job-sheets to be used from the input file. By default this value is 0, which means that all job-sheets in the input file are preprocessed. However, if you want to use just a subset of the input file, than you can define here the number n of job-sheets to use, where n are the first n job-sheets in the input file.

The checkbox “Include intermediate results in output file” specifies if intermediate results need to be placed in the output file. By default this option is disabled, because it slows down the preprocessing.

Starting the preprocessing

By clicking the button “PreProcess”, the preprocessing will start and two progress bars will show the actual progress of the process. Below-right a clock will keep track of the time the process runs. Once the process is finished, a message will show up indicating if the process has successfully completed or an error has occurred. In case the process has successfully completed, an output file is placed into the same folder as the selected input file. The name of the output file will be the same as the name of the input file, except the extension, which is “.cvppi” (clavis verbum preprocessed input) in case of a seed file and “.cvppu” (clavis verbum preprocessed unkown) in case of a unkown jobsheets file. Examples of the output files can be found in Appendix B. In case an output file with exact the same name already exists in the folder of the input file, a popup window will show up at completion of the process, with the question to overwrite the existing output file or not. In case the existing output file does not to be overwritten, a timestamp is added to the name of the output file to be created. Note that the output file is also a Microsoft Access 2007 file, and can as such being opened by Microsoft Access 2007 or higher.

As long as the process has not being started, one can return to the main screen of the CV tool by clicking on the “go back” button left-below. Once the process has being started, it can be stopped by clicking the “Cancel” button right-below.

5. Classification section

The classification section of the CV tool is meant to classify unclassified and preprocessed data using a preprocessed seed file. The initial screen of this section can be seen in Figure 32. The left side of the screen is the working area, which contains all the buttons and settings. The right side of the screen contains all the steps of the classification process, where the current step is marked by a grey bar. When a step has been completed, its checkbox will be checked.

First a preprocessed seed file needs to be selected. Once a file is selected, new configuration options will show up (Figure 33).

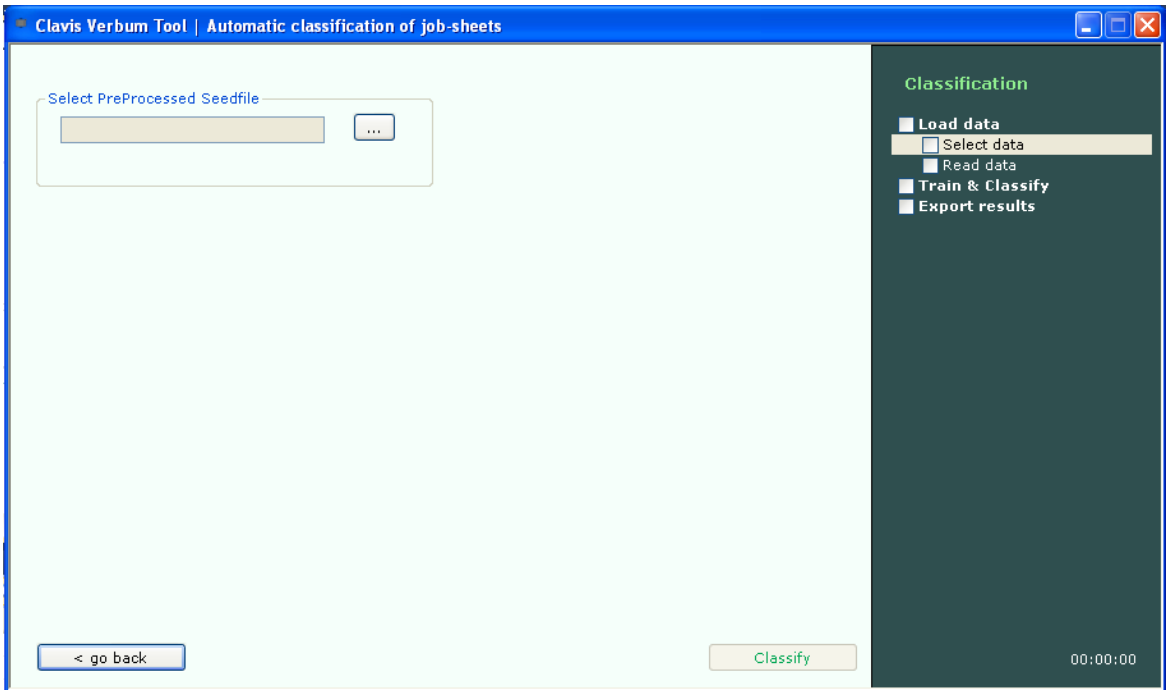


Figure 32

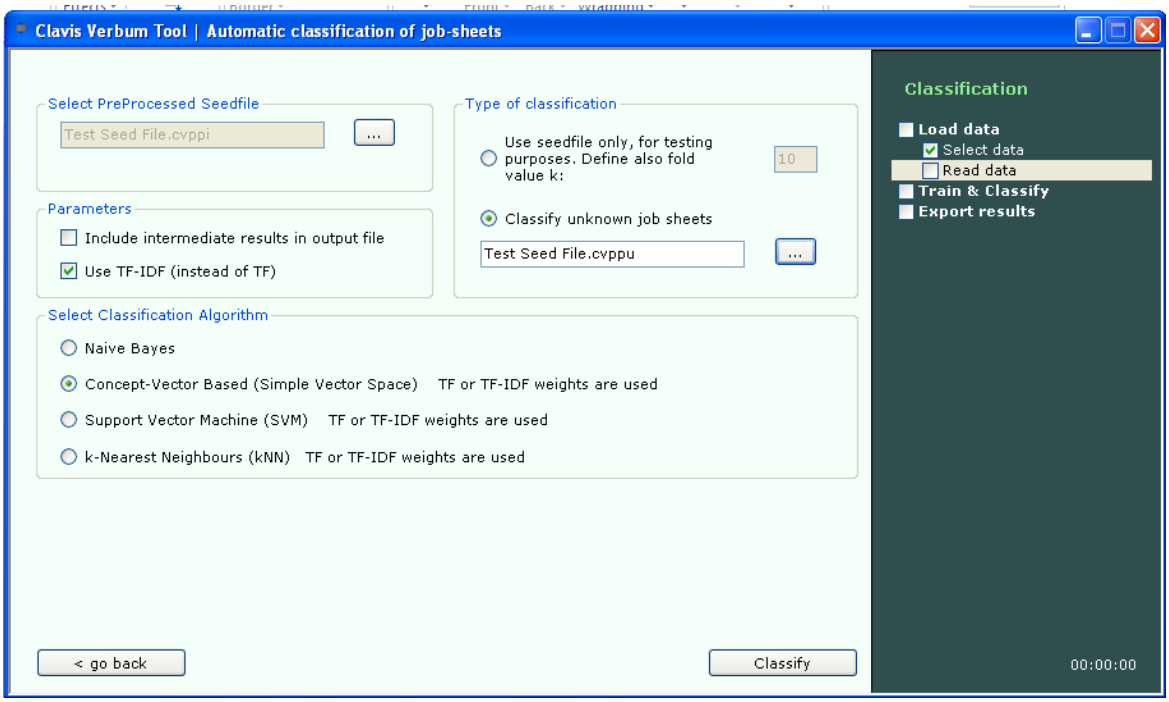


Figure 33

Type of classification

Two types of classification can be selected. The default type is “Classify unknown job sheets”, for classifying unclassified job-sheets in a preprocessed input file, which needs to be selected. The other classification type “Use seedfile only, for testing purposes”. For this second type no additional input file has to be selected, but a k-value needs to be given (default is 10). This type of classification will use only the selected seed file and divides the seed file in k equal parts. During classification, each part is used as a test set. A test set is a set of job-sheets with known categories that will be classified, based on a model trained by all other job-sheets not occurring in the test set. By doing so, the performance of the classification and the accuracy of the seed-file can be measured. This type of classification is only needed to test which classification settings work best for certain data and to validate the seedfile.

Parameters

The checkbox “Include intermediate results in output file” specifies if intermediate results need to be placed in the output file. By default this option is disabled, because it slows down the classification.

The checkbox “Use TF-IDF (instead of TF)” specifies if a more advanced computation needs to be used during classification. This parameter is checked by default. Note that this setting only influences a part of the classification algorithms, as can be seen after the names of the available algorithms.

Select Classification Algorithm

In this area, the used classification algorithm (technique) needs to be selected. Each algorithm has its strengths and weaknesses and it depends on the type of data to be classified, which algorithm is most optimal. The classification type using only the seed file might be useful in determining the most optimal algorithm. By default the “Concept-Vector Based (Simple Vector Space)” algorithm is selected. This algorithm performs best in case of job-sheets, being preprocessed using the n-grams setting.

Starting the classification

By clicking the button “Classify”, the classification will start and two progress bars will show the actual progress of the process. Below-right a clock will keep track of the time the process runs. Once the process is finished, a message will show up indicating if the process has successfully completed or an error has occurred. In case the process has successfully completed, an output file is placed into the same folder as the selected seed file. The name of the output file will be the same as the name of the seed file or the preprocessed input file containing unclassified job-sheets. Only the extension will be different, which is “.cvo” (clavis verbum output). In case an output file with exact the same name already exists in the folder of the input file, a popup window will show up at completion of the process, with the question to overwrite the existing output file or not. In case the existing output file does not to be overwritten, a timestamp is added to the name of the output file to be created. Note that the output file is also an Microsoft Access 2007 file, and can as such being opened by Microsoft Access 2007 or higher. In appendix B two example output files can be found, one for each type of classification. The way the output file looks like, depends on the type of classification.

As long as the process has not being started, one can return to the main screen of the CV tool by clicking on the “go back” button left-below. Once the process has being started, it can be stopped by clicking the “Cancel” button right-below.

Appendix A – Input Files

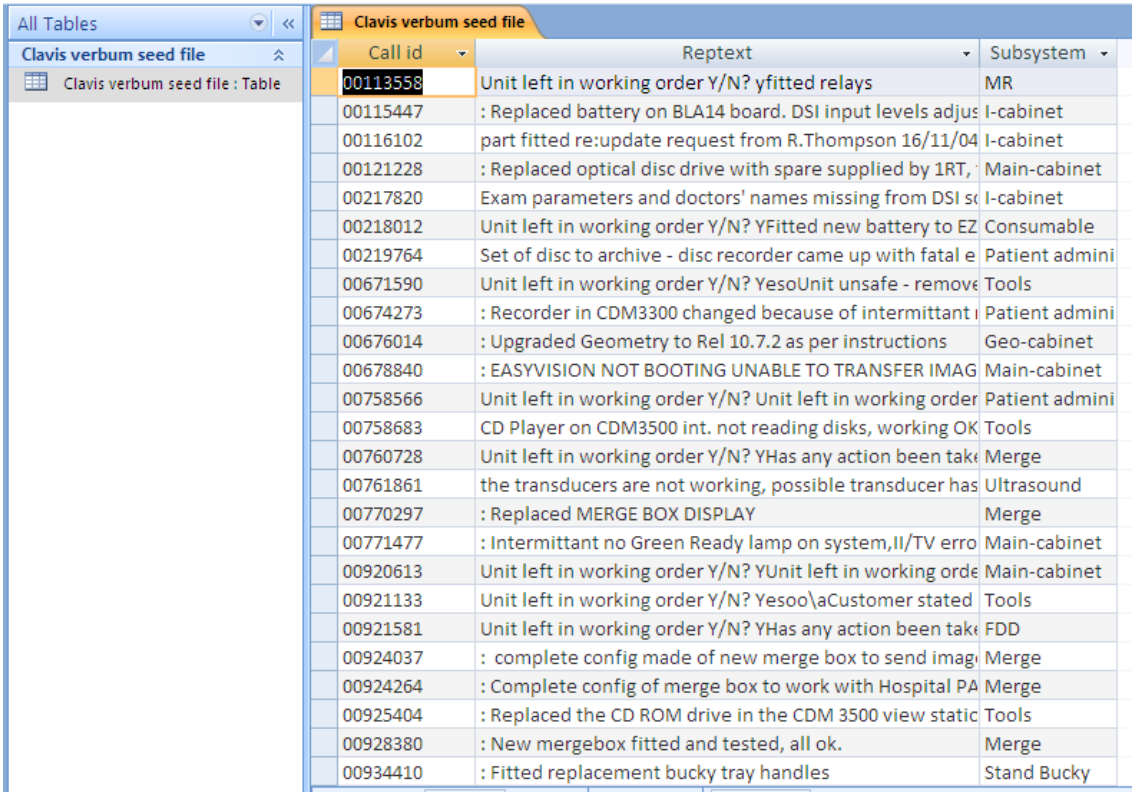
In this appendix short descriptions are given on how to create a seed file and an unknown job-sheets file.

Seed file

1. Create a new access 2007 file (.accdb) in Microsoft Access 2007 or higher.
2. Create a table in the newly created access file and name it “Clavis verbum seed file”. Note the capital letters!
3. In the newly created table, create three columns (Note the capital letters!):

Field Name	Data Type	Explanation
Call id	Text	Contains unique identifier
Reptext	Memo	Contains the actual text
Subsystem	Text	Contains the predefined category of the text

Example of seed file:



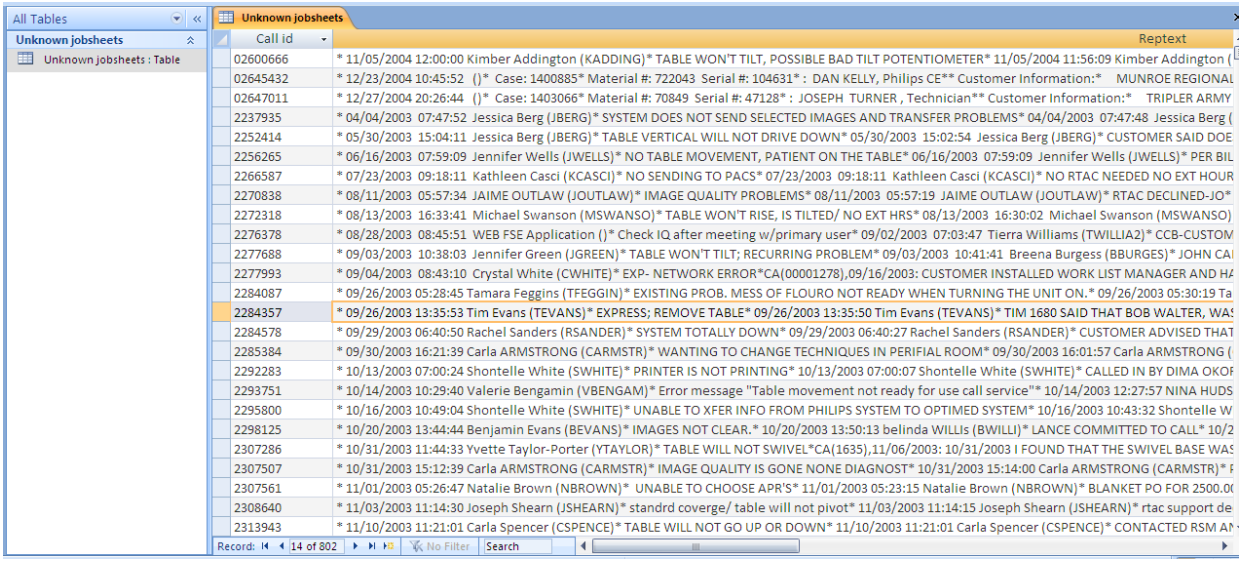
Call id	Reptext	Subsystem
00113558	Unit left in working order Y/N? yfitted relays	MR
00115447	: Replaced battery on BLA14 board. DSI input levels adjus	I-cabinet
00116102	part fitted re:update request from R.Thompson 16/11/04	I-cabinet
00121228	: Replaced optical disc drive with spare supplied by 1RT,	Main-cabinet
00217820	Exam parameters and doctors' names missing from DSI sc	I-cabinet
00218012	Unit left in working order Y/N? YFitted new battery to EZ	Consumable
00219764	Set of disc to archive - disc recorder came up with fatal e	Patient admini
00671590	Unit left in working order Y/N? YesoUnit unsafe - remove	Tools
00674273	: Recorder in CDM3300 changed because of intermittant i	Patient admini
00676014	: Upgraded Geometry to Rel 10.7.2 as per instructions	Geo-cabinet
00678840	: EASYVISION NOT BOOTING UNABLE TO TRANSFER IMAG	Main-cabinet
00758566	Unit left in working order Y/N? Unit left in working order	Patient admini
00758683	CD Player on CDM3500 int. not reading disks, working OK	Tools
00760728	Unit left in working order Y/N? YHas any action been take	Merge
00761861	the transducers are not working, possible transducer has	Ultrasound
00770297	: Replaced MERGE BOX DISPLAY	Merge
00771477	: Intermittant no Green Ready lamp on system,II/TV erro	Main-cabinet
00920613	Unit left in working order Y/N? YUnit left in working orde	Main-cabinet
00921133	Unit left in working order Y/N? Yesoo\aCustomer stated	Tools
00921581	Unit left in working order Y/N? YHas any action been take	FDD
00924037	: complete config made of new merge box to send imag	Merge
00924264	: Complete config of merge box to work with Hospital PA	Merge
00925404	: Replaced the CD ROM drive in the CDM 3500 view static	Tools
00928380	: New mergebox fitted and tested, all ok.	Merge
00934410	: Fitted replacement bucky tray handles	Stand Bucky

Unknown job-sheets file

1. Create a new access 2007 file (.accdb) in Microsoft Access 2007 or higher.
2. Create a table in the newly created access file and name it “Unknown jobsheets”. Note the capital letters!
3. In the newly created table, create two columns (Note the capital letters!):

Field Name	Data Type	Explanation
Call id	Text	Contains unique identifier
Reptext	Memo	Contains the actual text

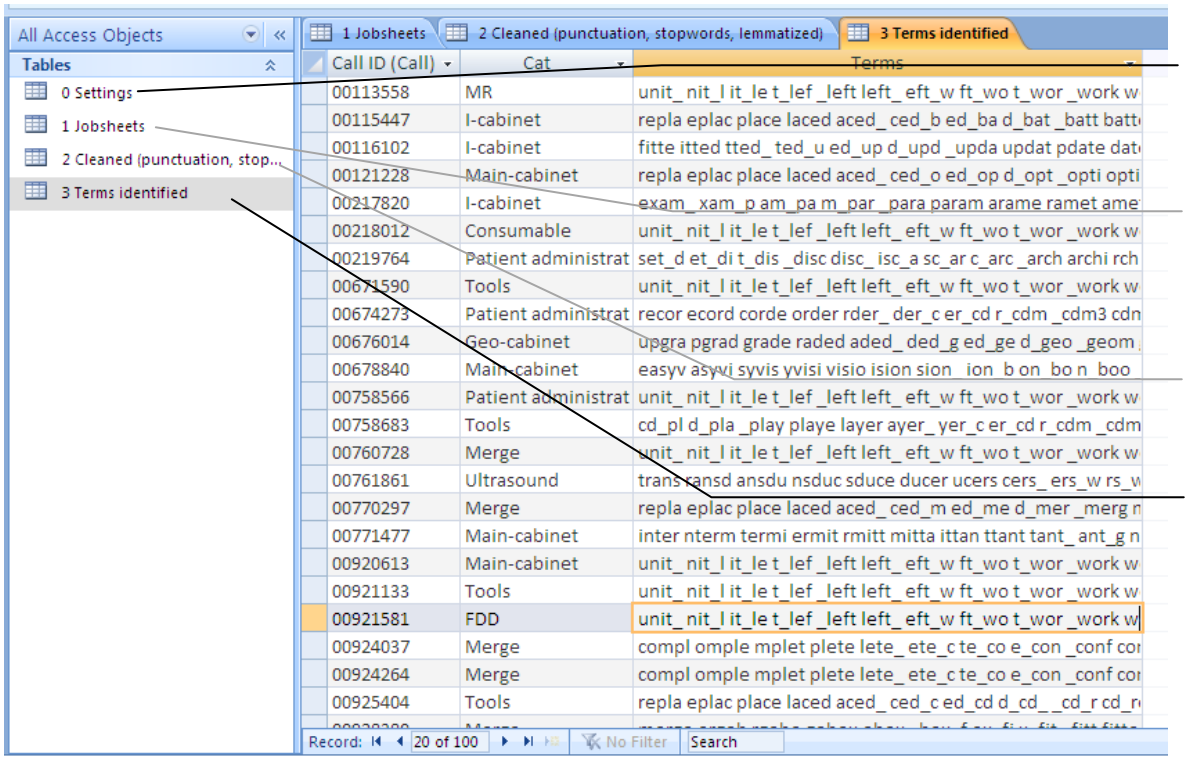
Example of unknown job-sheets file:


All Tables
Unknown jobsheets
Unknown jobsheets : Table
Unknown jobsheets
Call id
Reptext
02600666 * 11/05/2004 12:00:00 Kimber Addington (KADDING)* TABLE WON'T TILT, POSSIBLE BAD TILT POTENTIOMETER* 11/05/2004 11:56:09 Kimber Addington (
02645432 * 12/23/2004 10:45:52 ()* Case: 1400885* Material #: 722043 Serial #: 104631* : DAN KELLY, Philips CE** Customer Information:* MUNROE REGIONAL
02647011 * 12/27/2004 20:26:44 ()* Case: 1403066* Material #: 70849 Serial #: 47128* : JOSEPH TURNER , Technician** Customer Information:* TRIPLER ARMY
2237935 * 04/04/2003 07:47:52 Jessica Berg (JBERG)* SYSTEM DOES NOT SEND SELECTED IMAGES AND TRANSFER PROBLEMS* 04/04/2003 07:47:48 Jessica Berg (
2252414 * 05/30/2003 15:04:11 Jessica Berg (JBERG)* TABLE VERTICAL WILL NOT DRIVE DOWN* 05/30/2003 15:02:54 Jessica Berg (JBERG)* CUSTOMER SAID DOE
2256265 * 06/16/2003 07:59:09 Jennifer Wells (JWELLS)* NO TABLE MOVEMENT, PATIENT ON THE TABLE* 06/16/2003 07:59:09 Jennifer Wells (JWELLS)* PER BIL
2266587 * 07/23/2003 09:18:11 Kathleen Casci (KCASCI)* NO SENDING TO PACS* 07/23/2003 09:18:11 Kathleen Casci (KCASCI)* NO RTAC NEEDED NO EXT HOUR
2270838 * 08/11/2003 05:57:34 JAIME OUTLAW (JOUTLAW)* IMAGE QUALITY PROBLEMS* 08/11/2003 05:57:19 JAIME OUTLAW (JOUTLAW)* RTAC DECLINED-JO*
2272318 * 08/13/2003 16:33:41 Michael Swanson (MSWANSO)* TABLE WON'T RISE, IS TILTED/ NO EXT HRS* 08/13/2003 16:30:02 Michael Swanson (MSWANSO)
2276378 * 08/28/2003 08:45:51 WEB FSE Application ()* Check IQ after meeting w/primary user* 09/02/2003 07:03:47 Tierra Williams (TWILLIA2)* CCB-CUSTOM
2277688 * 09/03/2003 10:38:03 Jennifer Green (JGREEN)* TABLE WON'T TILT; RECURRING PROBLEM* 09/03/2003 10:41:41 Breena Burgess (BBURGES)* JOHN CA
2277993 * 09/04/2003 08:43:10 Crystal White (CWHITE)* EXP- NETWORK ERROR*CA(00001278),09/16/2003: CUSTOMER INSTALLED WORK LIST MANAGER AND HA
2284087 * 09/26/2003 05:28:45 Tamara Feggins (TFEGGIN)* EXISTING PROB. MESS OF FLOURO NOT READY WHEN TURNING THE UNIT ON.* 09/26/2003 05:30:19 Ta
2284357 * 09/26/2003 13:35:53 Tim Evans (TEVANS)* EXPRESS; REMOVE TABLE* 09/26/2003 13:35:50 Tim Evans (TEVANS)* TIM 1680 SAID THAT BOB WALTER, WAS
2284578 * 09/29/2003 06:40:50 Rachel Sanders (RSANDER)* SYSTEM TOTALLY DOWN* 09/29/2003 06:40:27 Rachel Sanders (RSANDER)* CUSTOMER ADVISED THAT
2285384 * 09/30/2003 16:21:39 Carla ARMSTRONG (CARMSTR)* WANTING TO CHANGE TECHNIQUES IN PERIFIAL ROOM* 09/30/2003 16:01:57 Carla ARMSTRONG (
2292283 * 10/13/2003 07:00:24 Shontelle White (SWHITE)* PRINTER IS NOT PRINTING* 10/13/2003 07:00:07 Shontelle White (SWHITE)* CALLED IN BY DIMA OKOF
2293751 * 10/14/2003 10:29:40 Valerie Bengamin (VBENGAM)* Error message "Table movement not ready for use call service"* 10/14/2003 12:27:57 NINA HUDS
2295800 * 10/16/2003 10:49:04 Shontelle White (SWHITE)* UNABLE TO XFER INFO FROM PHILIPS SYSTEM TO OPTIMED SYSTEM* 10/16/2003 10:43:32 Shontelle W
2298125 * 10/20/2003 13:44:44 Benjamin Evans (BEVANS)* IMAGES NOT CLEAR.* 10/20/2003 13:50:13 belinda WILLIs (BWILLI)* LANCE COMMITTED TO CALL* 10/2
2307286 * 10/31/2003 11:44:33 Yvette Taylor-Porter (YTAYLOR)* TABLE WILL NOT SWIVEL*CA(1635),11/06/2003: 10/31/2003 I FOUND THAT THE SWIVEL BASE WAS
2307507 * 10/31/2003 15:12:39 Carla ARMSTRONG (CARMSTR)* IMAGE QUALITY IS GONE NONE DIAGNOST* 10/31/2003 15:14:00 Carla ARMSTRONG (CARMSTR)* F
2307561 * 11/01/2003 05:26:47 Natalie Brown (NBROWN)* UNABLE TO CHOOSE APR'S* 11/01/2003 05:23:15 Natalie Brown (NBROWN)* BLANKET PO FOR 2500.0(
2308640 * 11/03/2003 11:14:30 Joseph Shearn (JSHEARN)* standrd coverge/ table will not pivot* 11/03/2003 11:14:15 Joseph Shearn (JSHEARN)* rtac support de
2313943 * 11/10/2003 11:21:01 Carla Spencer (CSPENCE)* TABLE WILL NOT GO UP OR DOWN* 11/10/2003 11:21:01 Carla Spencer (CSPENCE)* CONTACTED RSM AN
Record: 14 of 802
No Filter
Search

Appendix B – Output Files

In this appendix examples of all the type of output files are given, including short explanations.

Preprocessed Seed File (.cvppi)



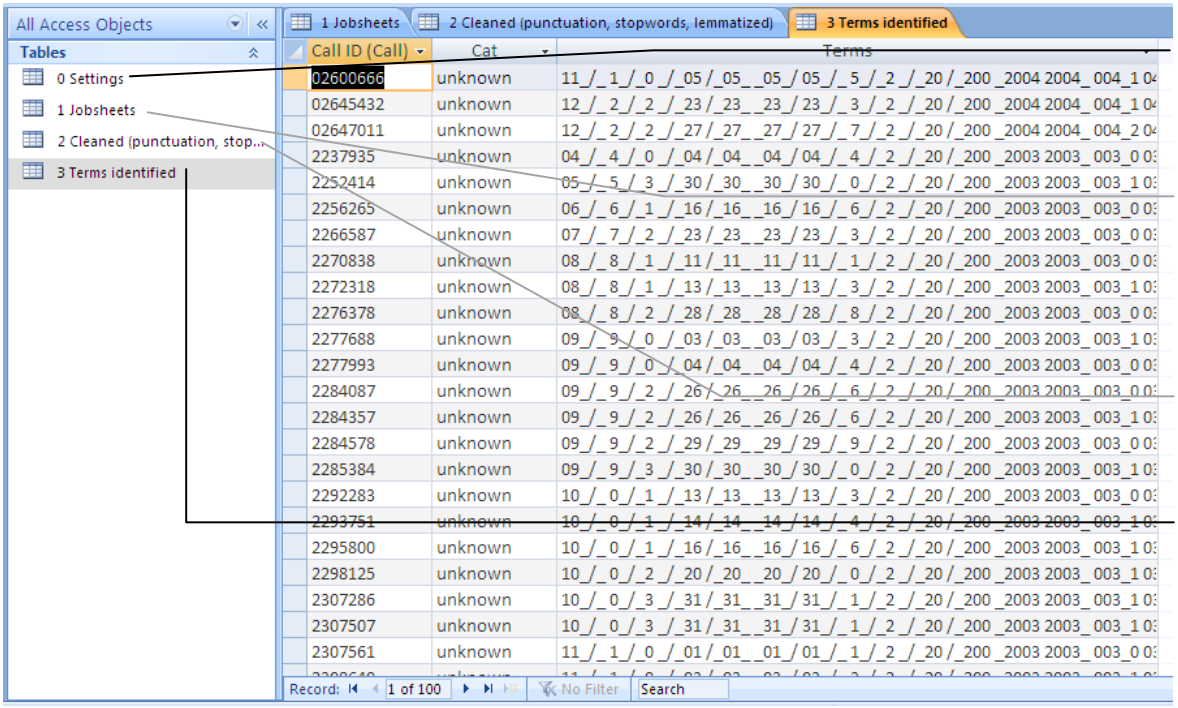
Overview of preprocessing options selected

Original seed file. Only if intermediate results selected

Intermediate results

Preprocessed seed file

Preprocessed unknown job-sheets file (.cvppu)



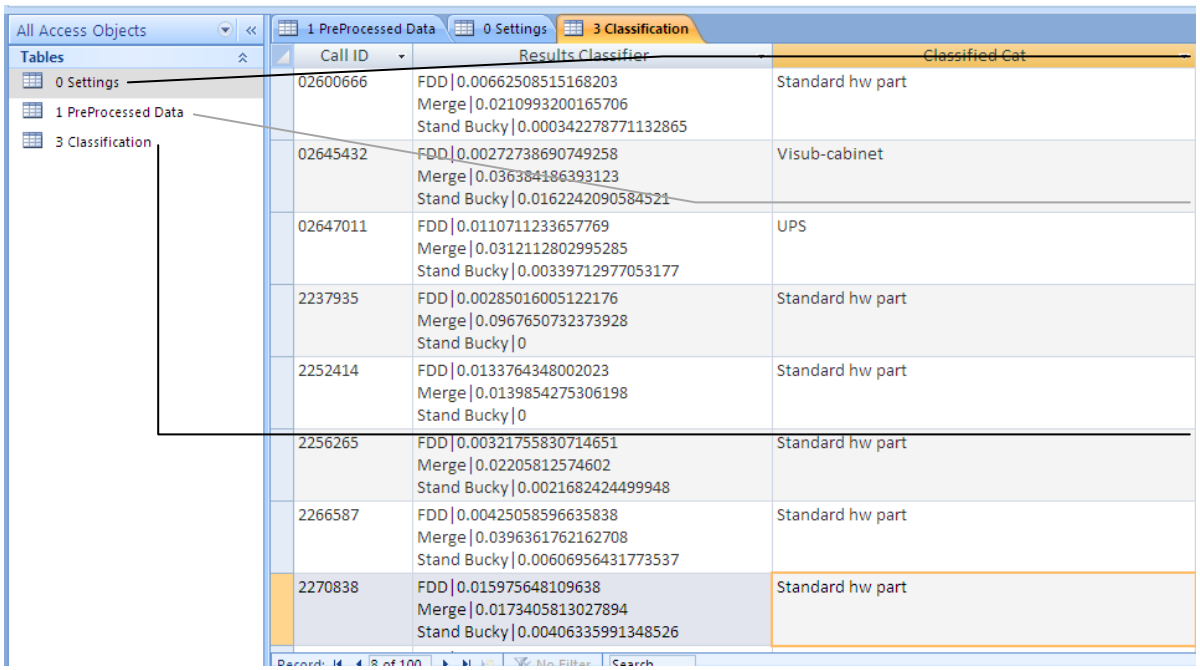
Overview of preprocessing options selected

Original unkown job-sheets file. Only if intermediate results selected

Intermediate results

Preprocessed unkown job-sheets file

Output file for classification of unknown job-sheets file (.cvo)



Overview of classification options selected

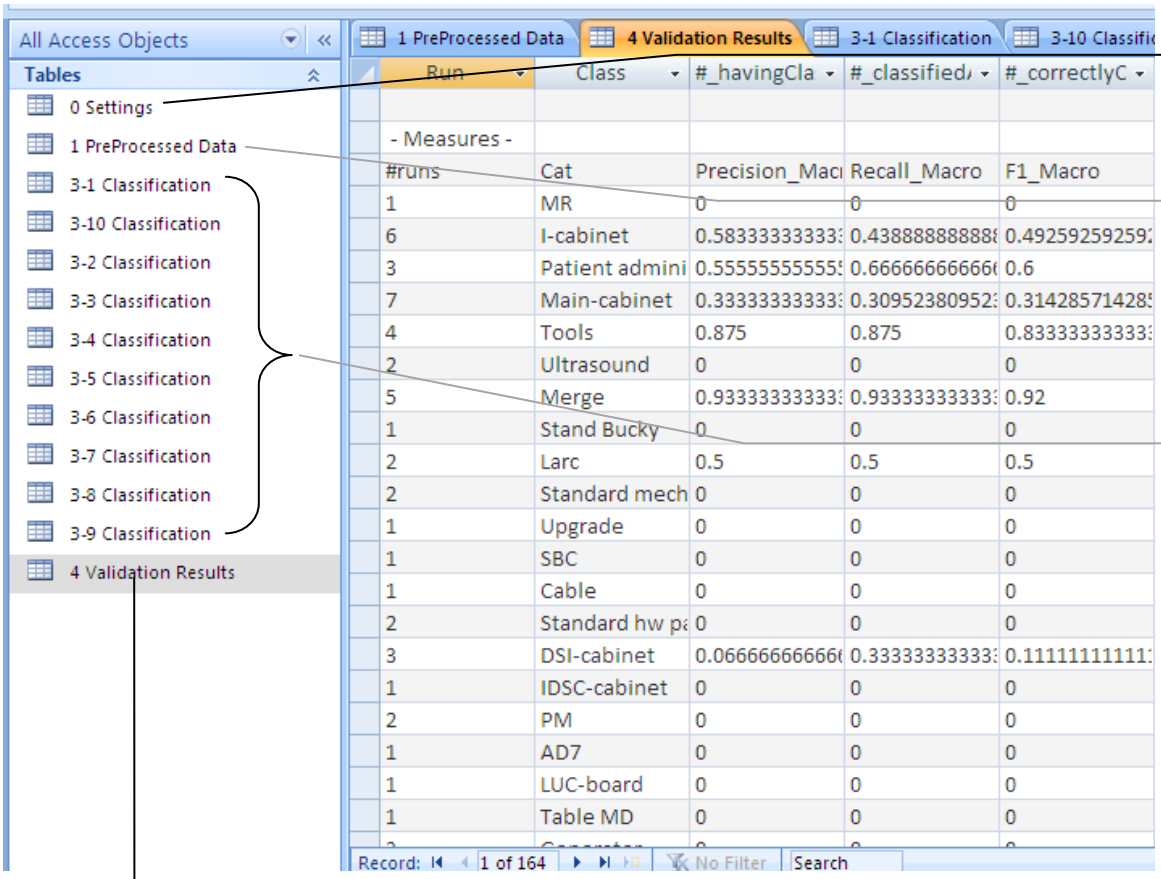
Original preprocessed data (unkown job-sheets file + seed file). Only if intermediate results selected

Classified data:

2nd column contains the chances a job-sheet belongs to a category.

3rd column contains the classified category (highest chance)

Output file for classification of seed file (k-fold validation) (.cvo)



Overview of classification options selected

Original preprocessed seed file. Only if intermediate results selected

Intermediate results:

For each run one of k test sets is used to classify and corresponding results are placed in a separate table

Performance results of the classification:

For each predefined category x is measured:

- Recall: Percentage of the job-sheets belonging to category x, that are classified as category x.
- Precision: Percentage of the job-sheets classified as category x, actually belonging to category x.
- F1: 2 * ((precision*recall) / (precision+recall))

Those measures for each category are also totalized. The higher the percentages are, the better the result. More information about the used measurements see http://en.wikipedia.org/wiki/Precision_(information_retrieval)

Appendix C – Quick Start Manual

This appendix describes the steps to perform a quick classification of unclassified job-sheets using standard settings.

1. Create a seed file using Appendix A (file containing manually classified job-sheets, which the tool uses to learn on how to classify unclassified job-sheets)
2. Create an unknown job-sheets file using Appendix A (file containing the job-sheets to be classified)
3. Start the Clavis Verbum (CV) tool and click on “Pre-process input” at the right of the main screen
4. Select “CV Seed File” as Type Input Datafile
5. Load the seed file at “Select Input Datafile”
6. Optionally change the number of job-sheets to use at the Parameters
7. Click on the “PreProcess” button and wait until it finishes
8. Once the preprocessing of the seed file has succeeded, select “Unknown Jobsheets file” as Type Input Datafile
9. Load the unknown job-sheets file at “Select Input Datafile”
10. Optionally change the number of job-sheets to use at the Parameters
11. Click on the “PreProcess” button and wait until it finishes.
12. Click on the “go back” button left-below to return to the main screen
13. Click on “Classification” at the right of the main screen
14. Load the preprocessed seed file at “Select PreProcessed Seedfile”
15. Load the preprocessed unknown job-sheets file below the selection “Classify unknown job sheets”
16. Click on the “Classify” button and wait until it finishes
17. Go to the directory where you have stored the seed file and unknown job-sheets file as created at step 1 and 2

18. Look for the *.cvo output file and open it in Microsoft Access 2007 or higher to see the results