DEGREE PROJECT IN COMPUTER ENGINEERING,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2016

Automated invoice handling with machine learning and OCR

Automatiserad fakturahantering med maskininlärning och OCR

ANDREAS LARSSON

TONY SEGERÅS

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF TECHNOLOGY AND HEALTH

Automated invoice handling with machine learning and OCR

Automatiserad fakturahantering med maskininlärning och OCR

Andreas Larsson
Tony Segerås

Degree Project in Computer Engineering
First cycle 15hp
Supervisor at KTH: Anders Lindström
Examiner: Ibrahim Orhan
TRITA-STH 2016:53

KTH
The School of Technology and Health
136 40 Handen, Sweden

Abstract

Companies often process invoices manually, therefore automation could reduce manual labor. The aim of this thesis is to evaluate which OCR-engine, Tesseract or OCRopus, performs best at interpreting invoices. This thesis also evaluates if it is possible to use machine learning to automatically process invoices based on previously stored data.

By interpreting invoices with the OCR-engines, it results in the output text having few spelling errors. However, the invoice structure is lost, making it impossible to interpret the corresponding fields. If Naïve Bayes is chosen as the algorithm for machine learning, the prototype can correctly classify recurring invoice lines after a set of data has been processed.

The conclusion is, neither of the two OCR-engines can interpret the invoices to plain text making it understandable. Machine learning with Naïve Bayes works on invoices if there is enough previously processed data. The findings in this thesis concludes that machine learning and OCR can be utilized to automatize manual labor.

Keywords
Machine learning, Naïve Bayes, OCR, OCRopus, Tesseract, Invoice handling

Sammanfattning

Företag behandlar oftast fakturor manuellt och en automatisering skulle kunna minska fysiskt arbete. Målet med examensarbetet var att undersöka vilken av OCR-läsarna, Tesseract och OCRopus som fungerar bäst på att tolka en inskannad faktura. Även undersöka om det är möjligt med maskininlärning att automatiskt behandla fakturor utifrån tidigare sparad data.

Genom att tolka text med hjälp av OCR-läsarna visade resultaten att den producerade texten blev språkligt korrekt, men att strukturen i fakturan inte behölls vilket gjorde det svårt att tolka vilka fält som hör ihop. Naïve Bayes valdes som algoritm till maskininlärningen och resultatet blev en prototyp som korrekt kunde klassificera återkommande fakturarader, efter att en mängd träningsdata var behandlad.

Slutsatsen är att ingen av OCR-läsarna kunde tolka fakturor så att resultatet kunde användas vidare, och att maskininlärning med Naïve Bayes fungerar på fakturor om tillräckligt med tidigare behandlad data finns. Utfallet av examensarbetet är att maskininlärning och OCR kan användas för att automatisera fysiskt arbete.

Nyckelord
Maskininlärning, Naïve Bayes, OCR, OCRopus, Tesseract, Fakturahantering

Preface

We would like to thank our supervisor at KTH, Anders Lindström, for the help and support he has given us during our three years at KTH. His contribution has been of much value to us and helped us to succeed with this thesis. We would also like to thank DGC and our company supervisor, Fredrik Lönnegren, for the time and help we received.

Glossary

OCR	Optical Character Recognition, a technique used to interpret scanned documents to plain text
Structure extraction template	A template made to extract the correct fields from an invoice
Invoice structure	The design and placements of tables, invoice lines and other important invoice data
Machine learning	Technique to teach a computer complex rules using different algorithms
Naïve Bayes	Machine learning algorithm using probability theory
Accord.NET	Open-source machine learning framework

Table of contents

1 Introduction

This chapter addresses why there was a need for a new system, how the client defined correct invoices, and the goals and limitations of this thesis.

1.1 Background

The company and network operator DGC was in need of a system for interpretation of invoices. The main purpose of the system was to determine if the data present in the invoice lines were correct, without human supervision. A system already existed but it had some limitations. The current system was able to interpret structured data from PDF and Excel documents with the help of regular expressions, based on templates depending on which company sent the invoice. The system was not able to interpret invoices of scanned image formats and when validating data, nor was it able to handle periodical costs. This was a problem due to the importance of invoice periods when figuring out what service the invoice pays for.

To solve the problems with the existing system, techniques for the following cases were evaluated.

- The interpretation of invoices, the performance of Optical Character Recognition (OCR) when extracting data from invoices in plain text, regardless who sent the invoice and format, i.e. PDF, Excel or image files.
- Applying text matching on the raw text to extract structured data from plain text and correct errors made in the OCR-process.
- Validation of invoice data: using machine learning to teach the system to make decisions about the correctness of an invoice.

Due to how the existing system worked, the emphasis of this thesis was to be on the validation of invoice data and applying of machine learning to it. Because the existing system already knew how to extract structured data from most types of invoices the company receive.

1.2 Invoice specification

To measure the correctness of the system, there had to be a way to evaluate what a correct invoice was. An invoice could contain one or more invoice lines, which all had their own specifics. The invoices contained recurring information, which was used for evaluating whether or not an invoice was correct. This was done by checking certain fields in the invoice line. The fields below were generic to all invoice lines included in this thesis. At the time of this project, the fields below were used to measure the correctness of an invoice line, in the future, fields might be added or removed.

- Amount: The total amount of the invoice line.
- Units: The units ordered.
- Unit Price: The price of each unit.
- Currency: The currency to pay the amount and unit price.
- Product Information: The invoice line information.
- Period: The number of days paid for.

A problem encountered with invoices was the inconsistency of positioning of the fields specified above. Depending on the company creating the invoice, the fields may be structured unlike other invoices from other companies. Another problem considered with the fields above was their values, when it was not needed to supervise data that was correct. For example, amount could have a minor difference in such a way that would not result in a notable loss if it were to be automatically processed. Similar rules had to be applied to some of the fields.

1.3 Goals

The main goal was to create a prototype with the ability to evaluate data from invoices and determine whether the invoice was correct or not. To achieve the goal, the work was divided into the following steps.

The survey should:

- evaluate existing OCR-engines on how they perform in terms of number of correctly recognized words on invoices,
- evaluate how text matching can be applied to extract structured data from plain text and correct OCR-generated errors through a survey of suitable theories and related works, and
- determine how the process of interpreting invoices would be automated using machine learning on the invoice specific data to conclude whether the invoice is correct or not.

The OCR-engine implementation should:

- make few spelling errors,
- produce readable plain text, and
- retain the invoice structure.

The machine learning implementation should:

- demonstrate whether the data in invoices is correct or should be supervised and corrected manually,
- be modular to make it easier to adapt the solution to future fields, and
- determine if an invoice is correct or not with a certain percentage.

The analysis should:

- evaluate how existing OCR-engines performs on different invoices,
- conclude if the machine learning prototype in fact becomes better as more data has previously been processed,
- calculate with how much certainty the machine learning prototype will deliver the answer, and
- evaluate if the technique is a working alternative to human supervision.

1.4 Delimitations

The delimitations in this project are:

- There will be no evaluation of how well OCR interprets handwritten text, only computer written documents will be used.
- Only English will be considered as the language the invoices are written in.
- Only the open-source OCR-engines OCRopus and Tesseract will be evaluated.

2 Background and Theory

This chapter focuses on the existing solutions in this field and also other fields that can be of interest to this thesis. First a discussion about the several different OCR-solutions, then text-matching will be discussed. Lastly a deeper discussion about machine learning and how it can be applied to the problem with invoice correctness.

2.1 Relevance of this thesis

The main reason for the study was to ease the workload of the economic team at DGC and to save time and money for the company. If processes like invoice handling can be automated, humans can put their attention on other more important problems. While the solution presented in this thesis may need some manual interaction, the machine learning part of this thesis might help decrease the amount of supervision needed as more data is processed.

2.2 Current system

The current system DGC uses is reached via an internal website within the company network, where users can upload invoices, which the system then interprets. The invoice is sent to a server where it is processed. The system can interpret the invoice if a structure extraction template has been made for the specific invoice format. If the system can interpret the invoice with a template, a result is presented to the user where the invoice head and invoice lines are presented in tables. The user has options to save the interpreted invoice to a database. What the current system does not include is the functions of telling the user if the content in the invoice is correct or not, and it does not remind the user if an invoice payment period is expiring and needs renewal.

2.3 OCR

Optical Character Recognition (OCR) is a technique used to interpret scanned documents into computer readable text. This thesis focus on using OCR to interpret invoices and their content. The OCR-process had to result in the invoice structure to be relatively alike how it was structured before the process. It was of uttermost importance that the structure of the invoice did not differ after the OCR process.

2.3.1 OCRopus and Tesseract

OCRopus and Tesseract were considered for the implementation, and the reason these engines were chosen was because of the delimitations set at the start of the thesis, to test invoice interpretation with only OCRopus and Tesseract. They were both open-source and DGC wanted to evaluate if they could prove to be suitable for their solution. Another reason the engines were chosen was because earlier studies existed, comparing the engines on historical texts, but no study were found on digital invoices.

2.3.2 Training the engines

OCRopus could be trained on data before it was used for image processing. Tesseract, however, already knows about the common fonts used and training is not necessary for digital fonts [1]. Training the engines is not part of the regular workflow, it is something the user has to do explicitly. Training engines entails a reduced error rate in terms of word spelling errors. There are two methods that can be used to train both Tesseract and OCRopus. The first one is the easiest and consists of having a human input a document and then give the correct form of the document, as seen in the article by Springmann et al. [2]. This gives the engine a sense of how to interpret the given document.

The second method is more advanced and include generating artificial images from available texts with the correct format. It might be possible to automate this method which, if successful, would make the learning faster.

2.3.3 Plain-text comparison OCRopus and Tesseract

Springmann et al. [2] used a historical context in their evaluation of these two engines that was based upon font-recognition of historical texts. While historical content was not something that was present in modern day invoices, it still showed how well these engines improved as they learned a given pattern. Invoices also have patterns, and when these engines interpret invoices instead of historical text, the success could correlate to the work of Springmann et al.

The evaluation done by Springmann et al. showed a small difference in average correctness between Tesseract and OCRopus when they were not trained. Tesseract showed an average of 78.77%, and OCRopus showed an average of 81.66%. This was based upon the average correctness per page in the total sample. This percentage corresponded to plain text scanned from a book.

2.3.4 Conclusion on OCR-scanning invoices

In this thesis the input was invoices. To further evaluate how accurate these engines behaved when handling such documents, an evaluation was needed. Due to the fact that no previous work was found using these engines to scan invoices, an evaluation will be conducted in later chapters. This was done to gain knowledge about how well the engines performed scanning data from invoices.

2.4 Text matching

Text matching or string pattern matching is described by Aoe [3] , and is well used in computer science due to several needs, such as interpreting real handwritten documents and digital invoices, matching a string to data patterns in a data source. In other words, text matching is used in almost every case of text processing. There are several different algorithms developed that can be used in the purpose of matching text, two of them are presented below.

2.4.1 Longest common substring

Longest Common Substring is a text-matching algorithm, which in two or more strings finds the longest common string.

The algorithm works as follows: a two-dimensional array is created with the length of the dimensions based on the lengths of the strings, and max length is set to zero. The two-dimensional array is then iterated through, checking whether the characters at the corresponding dimension index to the strings are equal. If they are, the value of the current element is set to the value from the element with one less index on both dimensions, plus one. And if the value in this element is greater than the max length, the max length is set to that value.

		A	B	A	B
	0	0	0	0	0
B	0	0	1	0	1
A	0	1	0	2	0
B	0	0	2	0	3
A	0	1	0	3	0

Figure 2.1 Example of a matrix result using the longest common substring algorithm on the strings “ABAB” and “BABA”

Figure 2.1 illustrates a comparison between the strings "ABAB" and "BABA”. The matrix shows that the first character in the common string was “B” and was therefore marked as 1 in the matrix, then the next common character in the two strings is “A” and since a common character has been found before the value in the matrix will be 2. This will continue until all characters in the two strings have been handled and the result will be as shown in Figure 2.1. The length of the found common substring is three and the string can be extracted.

2.4.2 Levenshtein distance

Another text matching algorithm is Levenshtein Distance. Levenshtein Distance is used to compute the distance between two strings similarity as stated by Haldar et al. [4]. The distance is computed by comparing two strings and determines the number of differences in characters between the strings. Differences in this case means substitution, insertion or removal of characters on a string to make it equal to the other. For example, the words “kitten” and “sitting” will return the distance three, since the value has been computed like this:

1. **k**itten → **s**itten (substitution of ”s” for “k”)
2. sitt**e**n → sitt**i**n (substitution of ”i” for “e”)
3. sittin → sittin**g** (insertion of “g” at the end)

By completing these steps, the distance of three can be determined

2.4.3 Case study: Invoice structure extraction

When it comes to extracting important fields from the data, Hamza et al. [5] have developed and tested a case-based reasoning approach for this problem.

The method developed by Hamza et al. is called CBR-DIA, which means Case-Based Reasoning for Document Invoice Analysis. Hamza et al. states that the method does not need to be previously trained and is adaptive. The method works as follows. First, the scanned words are organized in a more logical way to further work with. Then each word is given a set of attributes that states the *position* of the word, the *key-word* and the *nature* of the word. The *nature* of a word in this case is represented by an alphabetical character that defines whether it is a numerical, alphabetical or an alphanumerical word. If the word belongs to one of the words in a specified *key-word* list, it is tagged as a *key-word*. A line can be extracted from the scanned document, and the line itself has attributes of position and pattern. The pattern of the line is describing the nature of the fields in it, for example “ABCAA”, and will be used in table extraction.

Depending on what the pattern is related to, it can be a pattern structure (PS) or a key-word structure (KWS). A PS is when the actual pattern is related to a table, and a KWS is when the pattern is related key-words that are locally arranged in the document. The words in the document are divided into these two structures and are thereafter accessible to the first step of the CBR cycle, described by Aamodt et al. [6]. The first step is to look for similar cases in a document database. If there are no similar cases, a second CBR cycle is gone through to resolve the extracted KWS. The system compares the structure to stored structures in a structure database by graph edit distance, to find the nearest graph. If a similar structure is found, the system uses its solution of key-words and applies it on the structures. If not, rules associated with specific key-words will be applied and thereafter resolved.

After some experimental testing the solution achieved a recognition rate of 85.29% on documents with similar cases and 76.33% for documents without similar cases. The testing was applied on 950 invoices.

2.4.4 Case study: Automatic detection and correction of OCR errors

In the case of correcting the extracted data from scanned invoices, Sorio et al. [7] presented a solution supposed to automatically detect and correct OCR errors in an information extraction system. This corresponds to the problem with correcting errors made by OCR in this thesis. The extracted fields, here called elements, runs through an algorithm to receive the correct value to work with. The candidate string of the element run through a set of stages matching the element specifications. The stages clean the candidate strings to make it satisfy the syntax of the element.

The next stage is the substitution stage where, based on a set of substitution rules, the string is expanded on a set of strings that are possible candidates for the element. Thereafter the strings are put through a syntactic and semantic stage to sort out the candidates, which does satisfy these checks. The last stage is the element selection stage where the string candidate's minimal distance (Levensthein distance), with weight on character removal is calculated against the extracted string. The one with lowest score is chosen. After every element has a determined string, they are put through a correlated element fixing stage, where a correlated group of elements are semantically checked.

The result of the performance evaluation of the system made Sorio et al. state that the system overall nearly doubled the percentage of correctly identified values, from 49.1% to 86.8% on a data set with good quality invoices, and from 31.7% to 61.23% on a data set with invoices of poor quality. All in all, their tests were executed on a data set of 800 values that were extracted from 100 commercial invoices.

2.4.5 Conclusion on text matching on raw data from invoices

What can be concluded from the studies found and evaluated section 2.4.3 and 2.4.4 was that there exists solutions for the purposes for using of text matching relating to the problems of this thesis. Based on the results from the studies made in chapter 2.3, an implementation of extraction templates will not occur. This was due to no previous work found in using OCR-engines with invoices without extraction templates. An implementation of OCRopus and Tesseract to test this was completed.

2.5 Machine learning

The theory behind machine learning is applying a previous solution, to solve a future problem as stated by Kulkarni [8]. The solution needs to be calculated based on previous solutions and other relevant information to make a qualified guess that may or may not be correct.

There are several different algorithms and techniques used in machine learning. Not all of them are applicable on invoice data validation, and as such only those relevant to invoice data validation were evaluated.

2.5.1 Theoretical review

According to Kulkarni [8], the use of machine learning is based on knowledge and the way to augment knowledge in a specific context. In other words, it is teaching the machine to have basic knowledge of a context and to improve over time so that the augmented knowledge can be used for decision making.

The way of teaching the system is by applying the way knowledge life cycle works, as knowledge building and learning is closely associated with each other. The life cycle has five phases. The first is the understanding of the context. In order to acquire knowledge about a context a base knowledge of the context is needed to begin with, to understand the context on an overview level. The next step involves gathering of information and acquiring of knowledge. This is the step where information is collected from various sources on the context. Then comes the analysis of information phase, where the collected data from the previous step is analyzed. By applying various basic analysis methods, to store the data in various groups and map it to priorities and decision scenarios. These will then help to generate behavioral patterns. Furthermore, is the learning phase, where the system learns to use knowledge, based on empirical data while it explores new scenarios. Last is the enhancement phase. Here the system enlarges its current knowledge base by updating the priorities and scenarios.

2.5.2 Theoretical case studies

Kulkarni presented a couple of case studies that were helpful for understanding how machine learning systems were built in different scenarios. What their scenarios showed were based on what was happening over time. Their systems aggregate the knowledge from generated data to make efficient decisions by augmented knowledge. Because of the way machine learning improves over time and how it discovers patterns in data sets, it had the potential to be applied on validating invoice data.

2.5.3 Ways to solve machine learning problems

Kulkarni states that problems to be solved with machine learning can be divided into two different paradigms depending on if the predicted value is in the training data or not. Training data implies data that has previously been processed by the system. If the predicted value is to be found in the training data, the problem belongs to the paradigm supervised learning, which also means that the data is labeled. When the predicted value is not in the training data, the problem belongs to the paradigm unsupervised learning. In unsupervised learning the data is unlabeled.

The problems are thereafter sorted into categories. Three of them were studied: **regression**, **classification** and **clustering**. Regression is suitable for predicting values like "*how many units of a product will be sold next month?*". Classification is suited for classifying data into two or more classes, for example "*is this a correct invoice or not?*". Clustering is suited for segmentation of data, like "*what are our customer segments?*". Both regression and classification are used in supervised learning, while clustering is used in unsupervised learning. The algorithm selected to solve the problem is discussed later in this thesis.

2.5.4 Spam filtering

According to Panigrahi et al. [9], a common application of machine learning is spam filters. These filters work by matching and learning how a spam email differs from a regular email. These filters need an extremely high rate of correctness, because if they are wrong and classify a non-spam email as spam, the consequences could be severe. Several different algorithms could be used to create a spam filter. This relates to the problem with invoice data classification as spam is the relation between text data. This text has several features and relations with each other and some words may be spam in one context but not in another. By using spam filters as a solution to text data classification, a solution which derives from spam filters was concluded.

As concluded by Yang et al. [10], one possible algorithm for spam filtering is Naïve Bayes. As spam filtering had lots of similarities with the invoice classification problem, Naïve Bayes had the potential to work with invoices as well.

2.5.5 Feature classification

In the case of invoice handling, an invoice has different features that needed to be taken into consideration when determining the correctness. The features considered could be constructed as the onto function presented in formula (1).

$$f = A \rightarrow B \tag{1}$$

Where A = Unrelated features and *B = Affected features*. In this case the *relation A = {Amount, Currency, Period (days), Units, Unit Price, Product specifics}* and *B = {Correct}*. With this classification an assumption was made that every entry in set A affect the outcome of B.

2.5.6 Naïve Bayes

The Naïve Bayes algorithm has been widely used and derives from Bayesian decision theory. The algorithm is based upon features which the algorithm uses to classify the specified text. Rennie et al. [11] states that the algorithm does not take into consideration the relations between the features, which may be seen as a simple model. Due to features not having any relation between each other, it may result in patterns with more complex relations will not be found by the algorithm. According to Rish [12], Naïve Bayes is best utilized when the problem can be constructed as a *one-to-one* function.

Naïve Bayes algorithm determines the probability between different cases that something will happen, based on previously encountered facts. It determines the probability of each feature independently and they are multiplied with each other for all possible outcomes, and the outcome with the highest probability is chosen. Therefore, if the probability of one or more of the independent features for a specific outcome are zero, then the probability of the outcome is zero.

To illustrate how Naïve Bayes works, consider the following simple example. Suppose having two fruits, apple and banana, with the features *Long, Not Long, Yellow and Not Yellow*. Using a set of training data similar to Table 2.1.

Table 2.1 Example of a training set for classification of the fruits

Fruits	Long	Not Long	Yellow	Not Yellow
Apple	0	600	200	400
Banana	350	50	350	50
Total	350	650	550	450

All the probability combinations possible is calculated from the training data, P(Long|Banana), P(Yellow|Apple) and P(Not Long|Apple) and so forth.

An unknown fruit has the features Not Long and Yellow, and should be classified into a given fruit class. To classify the unknown fruit, formula (2) and formula (3) is utilized. Formula (2) calculates the probability of the fruit being a banana, while formula (3) calculates the equivalent but if the fruit was an apple.

$$P(Banana|Not\ Long\ and\ Yellow) = \frac{P(Not\ Long|Banana)*P(Yellow|Banana)*P(Banana)}{P(Not\ Long)*P(Yellow)} \quad (2)$$

$$P(Apple|Not\ Long\ and\ Yellow) = \frac{P(Not\ Long|Apple)*P(Yellow|Apple)}{P(Not\ Long)*P(Yellow)} \quad (3)$$

Calculations are made based on formula (2) and formula (3), and the results are compared. The one with the highest probability is chosen, in the example the unknown fruit will be classified as an apple. This problem illustrates how Naïve Bayes algorithm makes decisions based upon probability.

There are different versions of Naïve Bayes such as Gaussian, multinomial and Bernoulli. Gaussian is used with continuous values as stated by John et al. [13]. Multinomial is used when there are several possible outcomes of a variable. And Bernoulli were each feature is assumed to be binary valued, as stated by McCallum et al. [14]

As the problem with invoice classification has the features described by formula (1), an assumption could be that Naïve Bayes was not an option for this problem, because the function is *onto*. However, as the study done by Rish [12] concludes that Naïve Bayes works well in the cases of extremes "*completely independent features (as expected) and functionally dependent features (which is less obvious), while reaching its worst performance between these extremes*" [12, p.41]. This concludes that Naïve Bayes can be utilized with acceptable correctness.

2.5.7 Support Vector Machine

Support Vector Machines (SVM are used for classification problems in machine learning and are rather popular as stated by Campbell et al. [15]. They work with simple binary classification, *classification into two classes*. The algorithm generalizes the training data to make predictions on novel data. The training data contains a set of input vectors, containing features, where each input vector is paired with corresponding labels. The labels are labeled +1 or -1 depending on the features in the input vector. The labels from each input vector is plotted as datapoints in an input space. A directed hyperplane is then determined in the input space with support vectors. These support vectors and hyperplane is determined by the vectors with the greatest margin from the hyperplane. A greater margin means less generalization errors. The support vectors are used to determine the classification of an input.

A limitation with SVM was that it could only use integer features. What this means is if you want to include string values as features into the model these string values has to be converted into an integer key. This key then corresponds to the string value.

3 Method

This chapter presents the methods used when testing. First a definition on how to compare OCRopus and Tesseract. Afterwards a discussion about why Naïve Bayes was the chosen algorithm for machine learning and how it was tested. Lastly a discussion about how the test data was generated and evaluated.

3.1 Relevance of the chosen technologies

OCR was tested and evaluated because the current solution DGC possessed could not deal with invoices sent from every company. Instead of creating new extraction templates for each different invoice format, using OCR on these invoices to extract data for invoice processing was considered. The outcome from the survey on OCR processing, presented that no previous comparisons between Tesseract and OCRopus on invoice scanning were found without extraction templates. It was the reason text matching was not implemented. Due to this outcome a comparison was completed to evaluate which engine performed best without extraction templates.

Machine learning was tested and evaluated to determine whether machine learning could be an alternative to a human supervising the correctness of an invoice. The reason for applying machine learning was that:

- it could be implemented on the invoices that the current solution already can extract data from with extraction templates, and
- in the future it can be implemented on structured data extracted from invoices with OCR and text matching, without creating extraction templates.

Since every invoice line could have its own rules for features, it would be hard to model this using a distributed software approach with statements. The invoice lines could vary from invoice to invoice. It made machine learning trivial in this case, since it will learn from the data already present in the model and compute an answer with the data. It would make the process easier and with fewer possible errors.

3.2 Invoice comparison OCRopus vs Tesseract

Two methods were considered in how these engines were evaluated. The first method was scanning invoices with the engines without any previous training. This was done to evaluate the performance in cases when an OCR solution is needed to scan an invoice. The second method selected was to train the OCRopus before scanning the invoices. OCRopus was trained by taking an invoice image and it made image segments, containing single or group of words. These segmented images were corrected manually by inputting the correct sequence of characters corresponding to each image segment. OCRopus then taught itself how to interpret each segment. It was done to evaluate how much better OCRopus becomes after being trained to scan similar documents.

Tesseract engine was not trained because it already has knowledge of the regular fonts that are used in invoices, as described by the Tesseract Training guide by Dovev [1]. Tesseract could be trained to recognize new fonts such as handwriting, and therefore no training was needed on digital written invoices.

The correctness was calculated in two ways. First, evaluate the correctness of words by using formula (4), where R is the percentage of correctness.

$$R = \frac{Total\ number\ of\ correct\ words\ in\ the\ result}{Total\ number\ of\ words\ in\ the\ original} \quad (4)$$

Second, the result of the scan was evaluated by how interpretable it was in terms of how the words were structured in the result, compared to how they were structured in the invoice. The evaluation was done by a human with knowledge about how to use the interpreted data to extract and validate it.

Figure 3.1 shows two examples of scanned invoices and their invoice structures. These invoices were reduced to only include sample invoice lines, the actual test was completed on real invoices. Real invoices could not be shown in this thesis due to company policy. Figure 3.1 shows two different invoice structures, and another three were tested together with these two, all having different structures. Only five invoices were needed because each company always uses the same invoice structure. As such the invoices from five companies were tested to evaluate how the engines performs on different invoices.

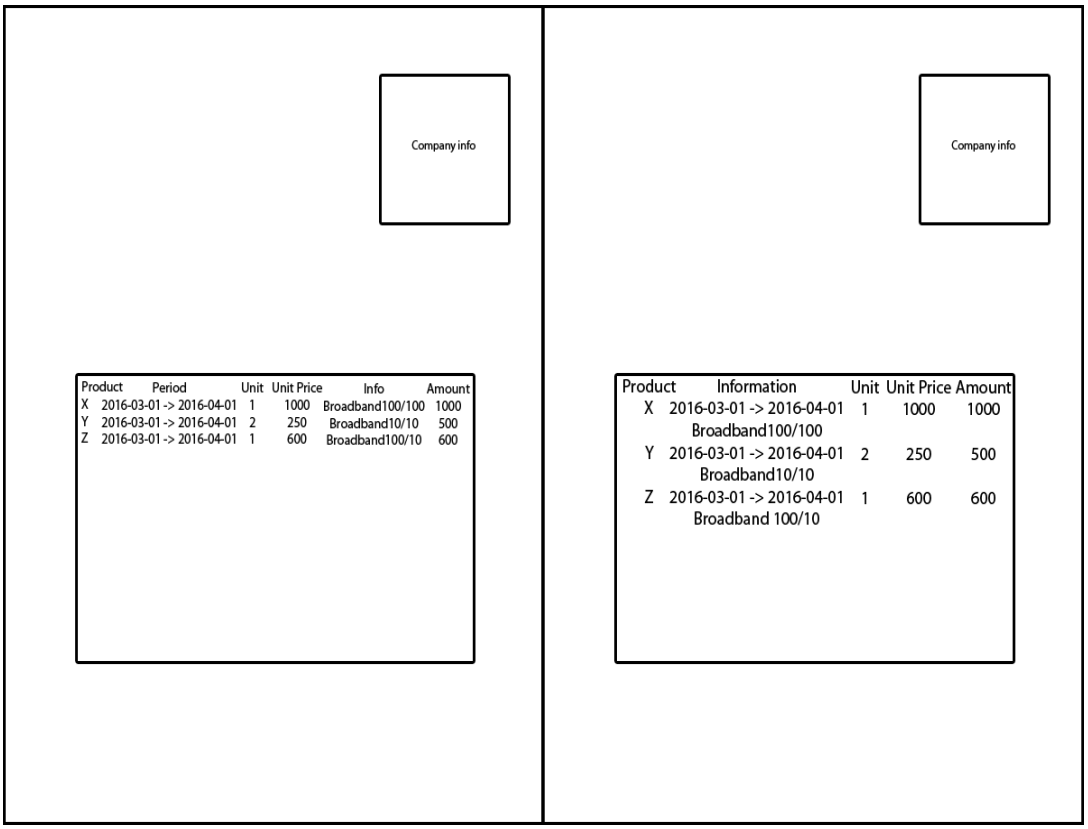

Figure 3.1 Example of two of the tested invoice structures

As can be seen in Figure 3.2, this was one of the invoice structures where the result of the scanning would likely present a correct result. If this was not the case, the interpretation would be problematic as the fields may not be in the correct position. Figure 3.3 shows an example of the expected result that should be presented after scanning the invoice with same invoice structure as in Figure 3.2.

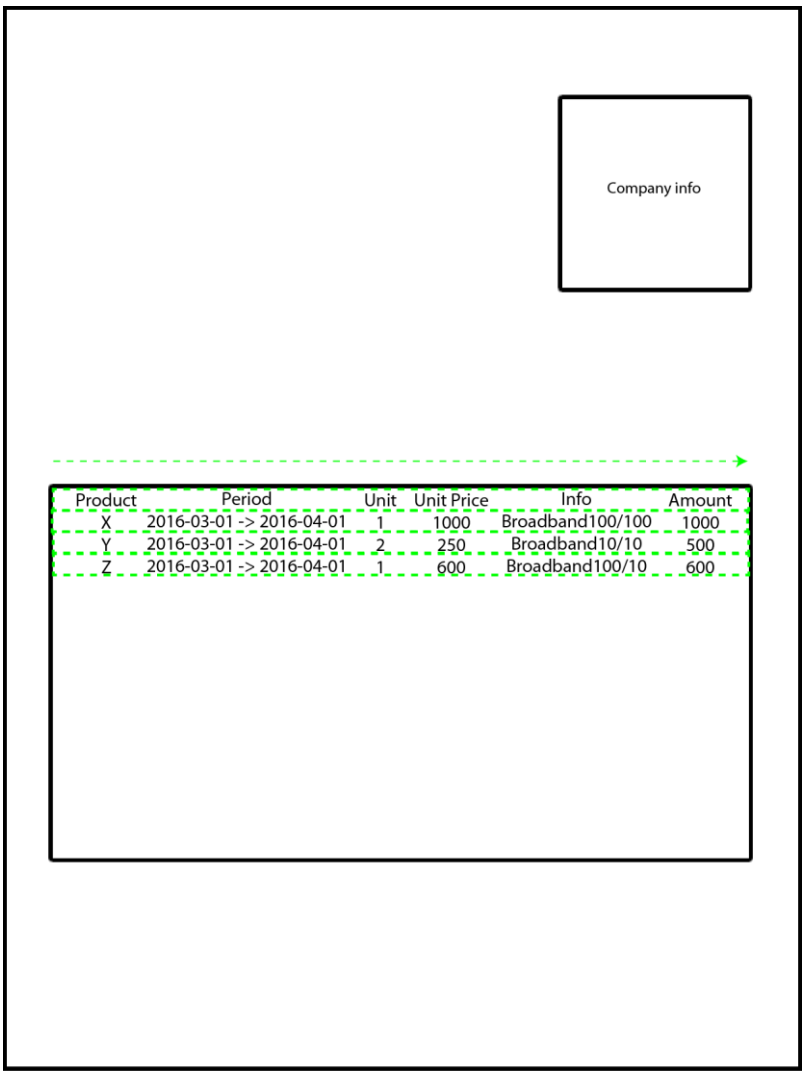

Figure 3.2 Invoice structure with correct row scanning

Product	Period	Unit	Unit Price	Info	Amount
X	2016-03-01 -> 2016-04-01	1	1000	Broadband100/100	1000
Y	2016-03-01 -> 2016-04-01	2	250	Broadband10/10	500
Z	2016-03-01 -> 2016-04-01	1	600	Broadband100/10	600

Figure 3.3 Example of ideally extracted plain text

3.3 Supervised Learning and classification

Supervised learning was chosen as the type of machine learning. Supervised learning is based upon learning with the answer present in the training data. In the context of invoice handling, an invoice can either be correct or not. It meant that the problem with invoice classification did not suit unsupervised learning, which is based upon the value to be predicted not existing in the training data.

Classification was chosen as the machine learning technique. The problem with invoice handling was to use machine learning to answer the question, “is this invoice correct or not?”. The problem was relatable to spam filtering with the question, is this text spam or not? As shown in section 2.5.4, the problem best fitted the description of classification and as such was the chosen path.

3.4 Naïve Bayes invoice classification

Supervised learning with classification was the path taken and as such an algorithm suited for this purpose was needed. Naïve Bayes was chosen as the test algorithm, the reason it was chosen for the implementation was its simplicity and studies already completed on Naïve Bayes spam filtering. Naïve Bayes also fits the classification problem because it is easy to implement string classification. An important aspect of Naïve Bayes was the ability to use both continuous and discrete values when classifying data.

Other algorithms such as Support Vector Machines have limitations in string classification as strings have to be placed into a given integer class. It caused problems because if string values, such as currency, units or product information should be included in the model, these values had to be mapped to a corresponding integer. It would increase the complexity of the model and require the strings and their corresponding integer key to be saved in a database. The increased complexity and problems with string classification were the reasons SVM was not chosen as the prototype algorithm.

The type of Naïve Bayes chosen was both multinomial and Gaussian because of the different values used. Amount, unit price and period were chosen as continuous values, currency, units and information were chosen as discrete. The reason amount, unit price and period were chosen as continuous values was because they could vary between the values of an invoice line classified as correct. Currency, units and product specifics were discrete, because they should not vary for an invoice line classified as correct, they should be either correct or incorrect.

The two different categories used for classification were correct or incorrect and if there was any uncertainty in the prediction, a warning should be sent for manual classification. This was due to the severity of accepting an incorrect invoice automatically. The initial classification was done manually by a human and was needed to create a training set used by the algorithm.

3.5 Training and testing Naïve Bayes on invoices

To make an assumption of whether an invoice line was correct or not using the Naïve Bayes algorithm, training was needed. The training data was used to generate the initial model to classify new data with, and all new classified data was added to the existing data set.

The reason there were several entries of training data was to give the model an idea of how the rules should be for each feature. If the model for example would only have a single data entry in the training set, with all six features having values that could vary, it would generate the need of manual classification every time new data were to be classified.

The training data consisted of values for the features according to formula (1) described in section 2.5.5, an example of the data is shown in Table 3.1. Two data sets of 100 invoice lines with randomly generated values for each data set were created. The reason two data sets were created was to evaluate if the classification improved with equal amount of generated data. The reason 100 invoices lines were generated for each data set was due to the amount of time spent manually classifying invoice lines. With an increased amount of data, the time spent classifying each line manually and correcting the prototype increased.

The first 25 lines of each data set were used as initial training data and was classified manually. Then, three iterations with 25 lines each were classified using the model of 25 entries as a training set. After the first iteration classified by the prototype, the lines were classified manually to correct the prototype classification and were added to the data set. The second iteration data was classified with a model containing the initial training data and the classified data from the first iteration. The last iteration was completed using a model generated by the previously classified data from iteration one and two. All invoice lines used in the iterations were manually classified.

Table 3.1 An example of a training set which was used to train the algorithm

Amount	Currency	Period (days)	Unit	Unit price	Information	Correct
1215	SEK	90	1	1215	ADSL10/10	Yes
1215	SEK	**120**	1	1215	ADSL10/10	No
1200	SEK	90	**2**	1215	ADSL10/10	No
1215	**SKE**	90	1	1215	**ADSL10/1**	No

Table 3.2 consists of a data entry with new unclassified data. As this unclassified data was estimated, the data was classified as either yes or no in the column labeled *Correct*. If the data could not be classified, a warning was triggered and classification of the data was done manually. When data was classified manually, the prototype learned how this new data was classified and added it to the existing data set.

Table 3.2 An example of unclassified data which the algorithm was supposed to classify

Amount	Currency	Period (days)	Unit	Unit price	Information	Correct
1218	SEK	90	1	**1218**	ADSL10/10	??

3.6 Generating training data

The data used to create the model were generated based on rules set up by the user. The rules were based upon the given invoice line, and different invoice lines could have different rules. The rules below were used when testing.

- Amount = 1215 ± 3.
- Currency = SEK, otherwise always incorrect.
- Period = 90 ± 7.
- Unit = min 1, max 4.
- Unit price = Amount/Unit.
- Information = Fiber100/100, otherwise always incorrect.

According to the rules above data was randomized within the ranges below.

- Amount: min 1211, max 1219.
- Currency: 90% correct with a chance of 10% error.
- Period: min 82, max 98.
- Unit: min 1, max 4.
- Unit price: Amount/Unit, with 5% chance of error.
- Information: 90% correct with a chance of 10% error.

The ranges had to create both correct and incorrect invoice lines, because the model had to be taught what the definition of correct and incorrect invoice lines were. The ranges also had to be randomized to give a close to even distribution of correct and incorrect lines, as such the ranges had to be small. If they were too large, the distribution in this case would be more incorrect than correct lines, and with smaller ranges the distribution would not hold enough permutations to create a large enough data set.

3.7 Measuring correctness

To measure the correctness of an invoice line, the algorithm had to follow the goals set up at the start of the project. From these goals, two cases where set up as how the prototype should behave.

- Does the algorithm warn users if there is any uncertainty?
- Does the algorithm improve as more data is inputted into the system?

The warning had to be visible to the user so an action could be taken. The algorithm should not put the estimated invoice data into the data set if there were any uncertainties. To trigger the warning, the assumption had to be less than 99% certain, equal to or above 99% was automatically processed. The reason 99% was chosen as the break point between automatically processed and warning sent was because of the severity processing invoices incorrectly. Choosing 99% as the break point and not 100% was that if the model should be 100% correct all the time, the amount of data needed would be too large to make the prototype useful. Because of this, 99% was chosen as high enough percentage to assume the invoice line correct or not.

As Naïve Bayes iterates through the training data and more estimated invoice data is put into the data set, it should become more accurate over time.

3.8 Prototype architecture

Two different prototypes were created. One for the comparison of output in terms of number of correct words from OCRopus and Tesseract, and another for testing machine learning for validation of data in invoices.

OCRopus was tested using Ubuntu inside a virtual machine. Command line was utilized to run OCRopus to extract plain text from images. Before extraction, a model was created which was based upon an existing English model from OCRopus. The model was trained to extract data from invoices.

Tesseract was implemented in the same Ubuntu virtual machine and command line was again utilized to extract plain text from invoices. Tesseract was used to extract the data without training. Tesseract ships with a predefined model trained for the most common fonts and languages.

The machine learning prototype was made to test how well Naïve Bayes notices deviations from the data set. The prototype used a database to store the previous processed data. The data had labels with either *correct* or *incorrect*. As can be seen in the flowchart in Figure 3.4 describing the machine learning prototype. The model is generated from the stored data and is used to classify the unclassified data. If the probability was equal or larger than 99% it would be automatically processed. If not, the user had to classify the data before it was stored.

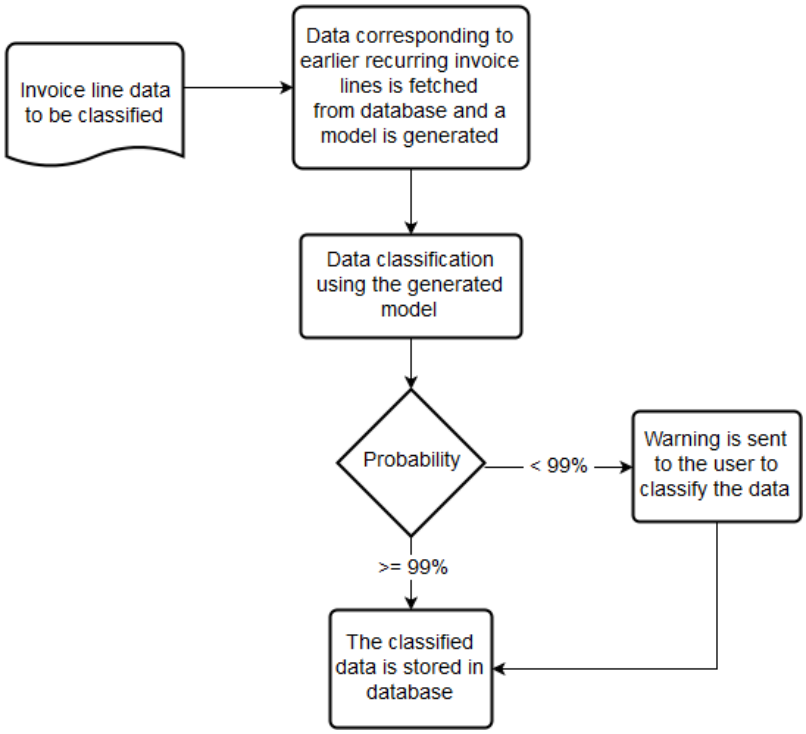

Figure 3.4 Flowchart describing the machine learning prototype

The framework chosen for implementing machine learning was Accord.NET Framework [16]. It was chosen because of the detailed documentation and the use it in other successful scientific researches [17]. A well-documented source code was important for this problem as several sources has to be able to verify the legitimacy of the framework. Accord.NET has a predefined Naïve Bayes class which is either Gaussian, if only continuous values are used, and is both Gaussian and multinomial if both discrete and continuous values are chosen as features. Because of the features chosen in this project, the Naïve Bayes model was both multinomial and Gaussian.

There is support for other techniques of machine learning techniques as well which can be interesting if other algorithms were considered. Accord was implemented into a C# project with Figure 3.4 as the origin. Accord has predefined unit tests which showed how to create, train and compute a model from a data set.

4 Result

This chapter presents the results that followed from the calculations presented in chapter 3. Firstly, the result from the OCR-engines comparison, secondly the result of the prototype design and lastly the result from the machine learning and Naïve Bayes correctness is presented.

4.1 OCRopus and Tesseract

The result in Figure 4.1 and Figure 4.2 was calculated using formula (4), the y-axis refers to R correctness percentage. The result of the comparison between the untrained engines is presented in Figure 4.1. Each invoice in Figure 4.1 and Figure 4.2 had different structures as shown from the examples in Figure 3.1 from section 3.2. It showed that they were decent enough to correct spelling mistakes to make a readable text for humans. But there was a problem matching the correct invoices lines to each other. The invoice structure was lost in the interpretation, which made it difficult for a human to use the extracted data even though the spelling was correct.

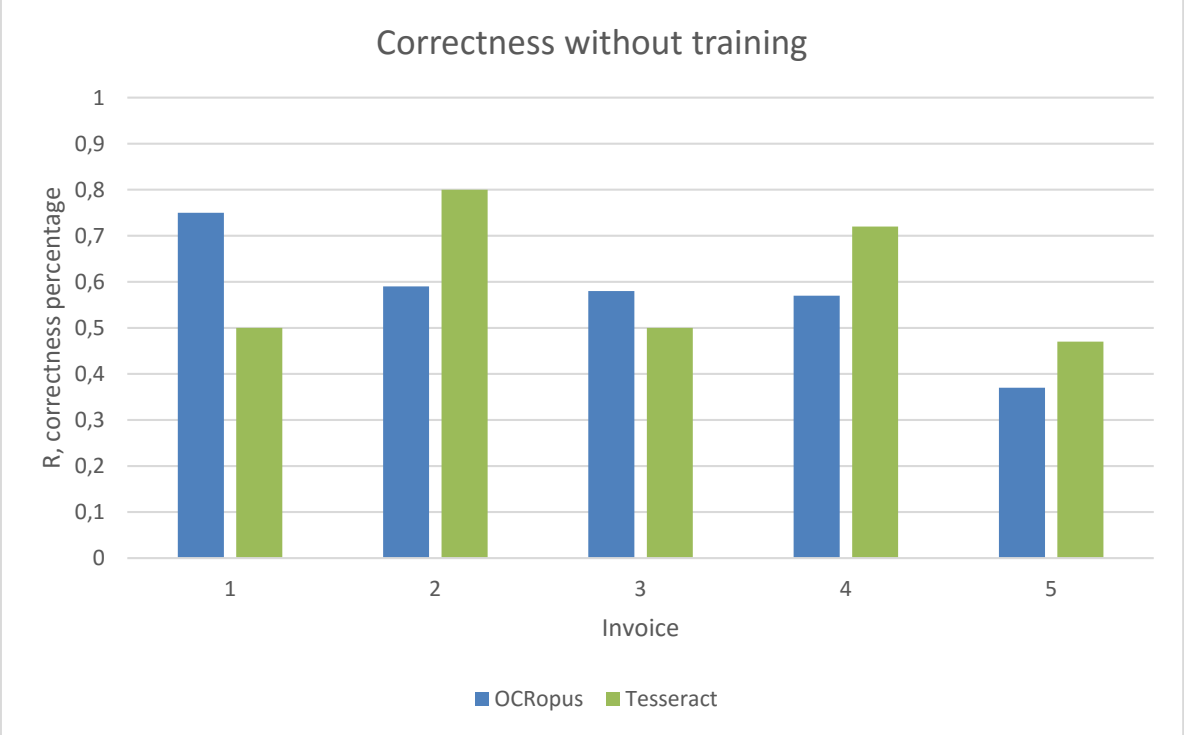

Figure 4.1 The spelling correctness of OCRopus and Tesseract without any training.

Figure 4.2 Shows the spelling correctness of OCRopus with and without training.

The spelling became even better as training was completed. As can be shown in Figure 4.2, the spelling errors improved, but the problem with structure was the same as without training. Because the training process only corrected itself on spelling and not on structure. As such the structure problems was still present and would not improve as training continued.

Figure 4.3 shows the structure problem and how the engines interpreted the invoices. The red columns represent how the fields were scanned, as a column instead of each row scanned separately. This caused an interpretation problem illustrated in Figure 4.4.

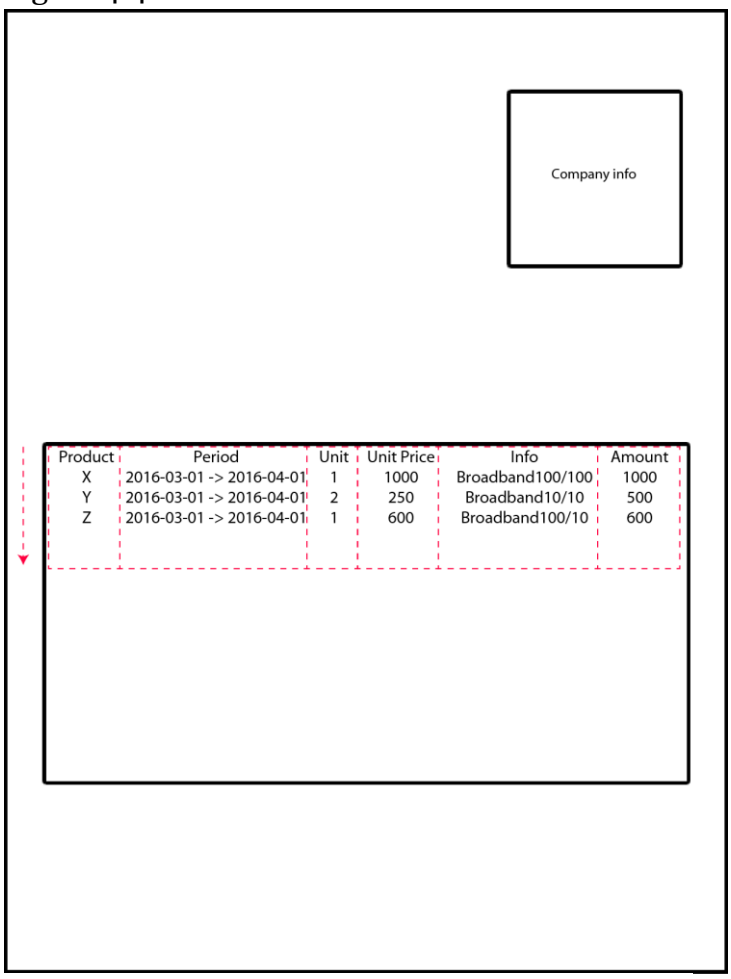

Figure 4.3 Illustrates the column scanning problem

Figure 4.4 shows a result from the scanning, were it was clearly visible how it was impossible to interpret the scanned data, as the fields were not connected to each other. Similar results were present in all five different invoices tested. The result showed it was impossible to use the plain text from the OCR-engines.

```
Product Period Unit Unit Price Info Amount
X
Y
Z 2016-03-01 -> 2016-04-01
2016-03-01 -> 2016-04-01
2016-03-01 -> 2016-04-01
1 2 1 1000
250
600
Broadband100/100
Broadband10/10
Broadband100/10 1000
500
600
```

Figure 4.4 The plain text result from the OCR scan

4.2 Naïve Bayes prototype

The prototype was made according to Figure 3.4 in section 3.8. It was constructed with modularity in mind and as such was easy to add features to. The resulting prototype used a database to store new classified data in the data set.

The prototype took an invoice as input and presented the answer to the question "Is it correct or not?". These invoices were processed and if there were any uncertainties a warning was displayed. The user had to classify the data and it was then added to the data set.

4.3 Machine learning with Naïve Bayes

By using the prototype with a set of recurring invoice lines a result was reached. Two complete data sets were used, each containing 100 invoice lines with randomly generated values, according to the rules and ranges presented in section 3.6.

The first 25 lines in each data set were used to train the model for each data set respectively. Then the other 75 lines in each data set were used for testing, over three iterations with 25 lines in each iteration. The complete set of training data for each data set is presented in Appendix A, and the complete set of iterations for each data set is presented in Appendix B.

Table 4.1 represents a subset of the 25 lines used to generate the model training data from the first data set.

Table 4.1 A subset of the 25 lines from the first data set used to train the model

Amount	Currency	Period (days)	Units	Unit price	Product specifics	Correct
1211	SEK	82	3	403,6666667	Fiber100/100	no
1216	SEK	88	4	304	**iber100/100**	no
1212	SEK	**98**	1	1212	Fiber100/100	no
1213	SEK	88	3	404,3333333	Fiber100/100	yes
1215	SEK	92	4	303,75	Fiber100/100	yes
1219	SEK	86	1	1219	Fiber100/100	no
1211	**SKE**	95	2	605.5	Fiber100/100	no

Table 4.2 presents a subset of the result from the first iteration of classified data, with predictions based on the training data. What could be observed was lines of data where discrete values, such as currency and product specifics diverged from the rule and they were classified with a percentage of 100 and equal to manual classification. All the other lines did not have high enough percentage, therefore manual classification was needed.

Table 4.2 Subset of the data in first iteration with prototype classification, percentage and manual classification

Amount	Currency	Period (days)	Unit	Unit Price	Product specifics	Prototype classification	Percentage	Manual classification
1215	SEK	96	3	405	Fiber100/100	yes	84%	yes
1211	SEK	82	4	302,75	Fiber100/100	yes	58%	no
1214	SEK	91	1	1214	**Fiber100100**	no	100%	no
1211	SEK	92	1	1211	Fiber100/100	no	93%	no
1214	SEK	89	3	404,666	Fiber100/100	yes	92%	yes
1215	SEK	96	3	405	Fiber100/100	yes	84%	yes
1218	**SKE**	84	2	609	Fiber100/100	no	100%	no

In Figure 4.5, a comparison between the results from the iterations in the first data set is presented. A line was classified correctly if the prototype classification was equal to how it would be manually classified. Lines were also automatically processed if they were classified with a percentage of 99 and higher. The result showed how the prototype improved after each iteration. The number of correctly classified lines increased after each iteration and improvements were made when more data was processed.

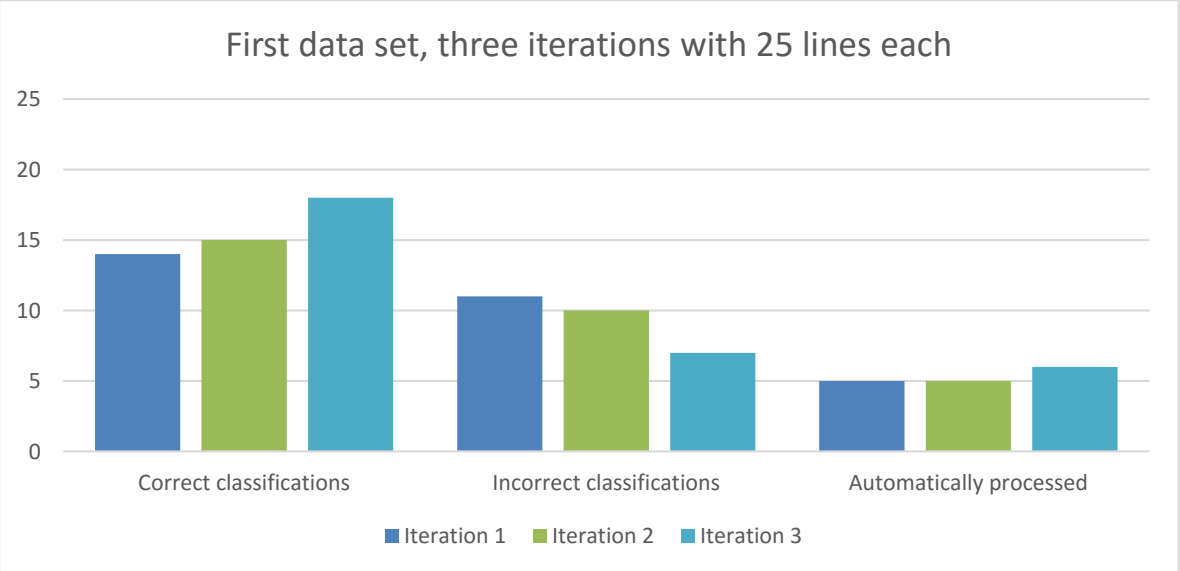

Figure 4.5 Result showing improvements between iterations in the first data set

In Figure 4.6, a comparison between the results on the second data set is presented. What could be observed was improvements in correct classifications between iterations similar to Figure 4.5. Though, the differences of automatically processed invoice was observable. The reason there were ten lines automatically processed in the second iteration compared to a lower result of seven in the third iteration, was because the randomly generated data used in the second iteration contained more lines with discrete values diverging from the rule.

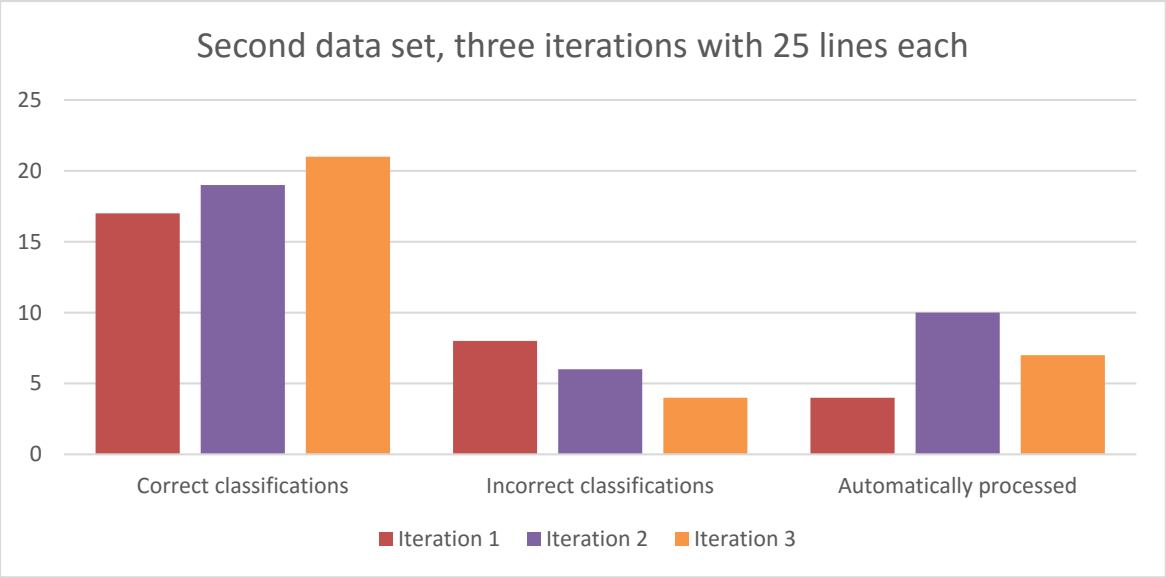

Figure 4.6 Result showing improvements between iterations in the second data set

In Figure 4.7 a comparison is presented between the both data sets after being processed by the prototype. The randomly generated data for the second data set presented a higher rate of correct classifications than the first, as well as a higher rate of automatically processed lines. What was common between the data sets was the improvement after each iteration.

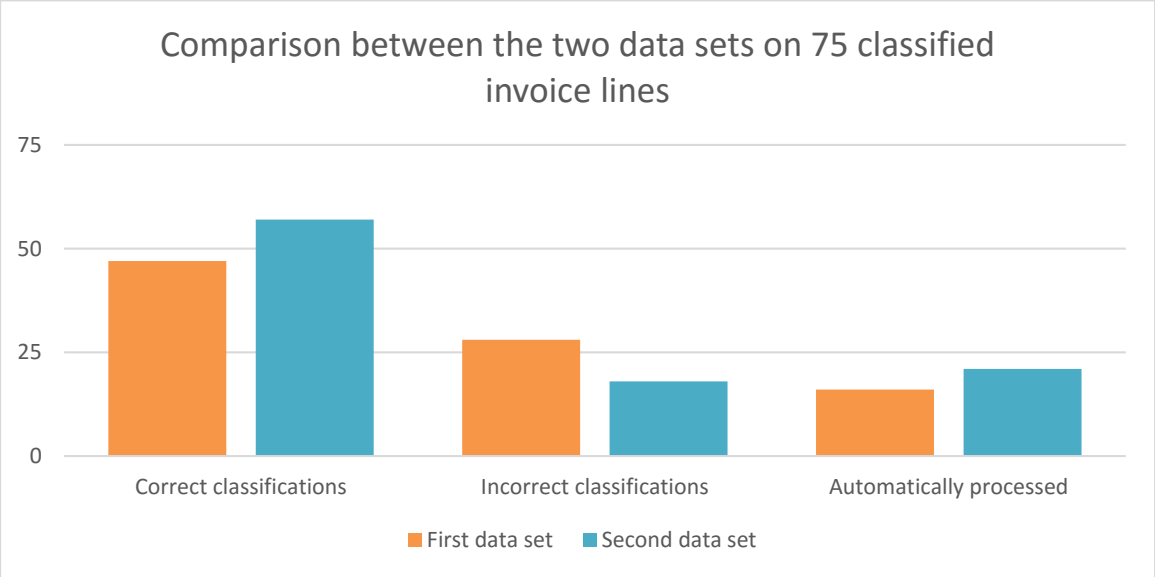

Figure 4.7 Comparison on the result from classification on 75 invoice lines

In Figure 4.8, the distribution of the generated values are presented. As stated in section 3.6 the distribution of the generated data had to be close to even, in terms of how many lines should be classified as correct and incorrect. The distribution shown in Figure 4.8 corresponds to the desired distribution with a close to equal distribution between correct and incorrect.

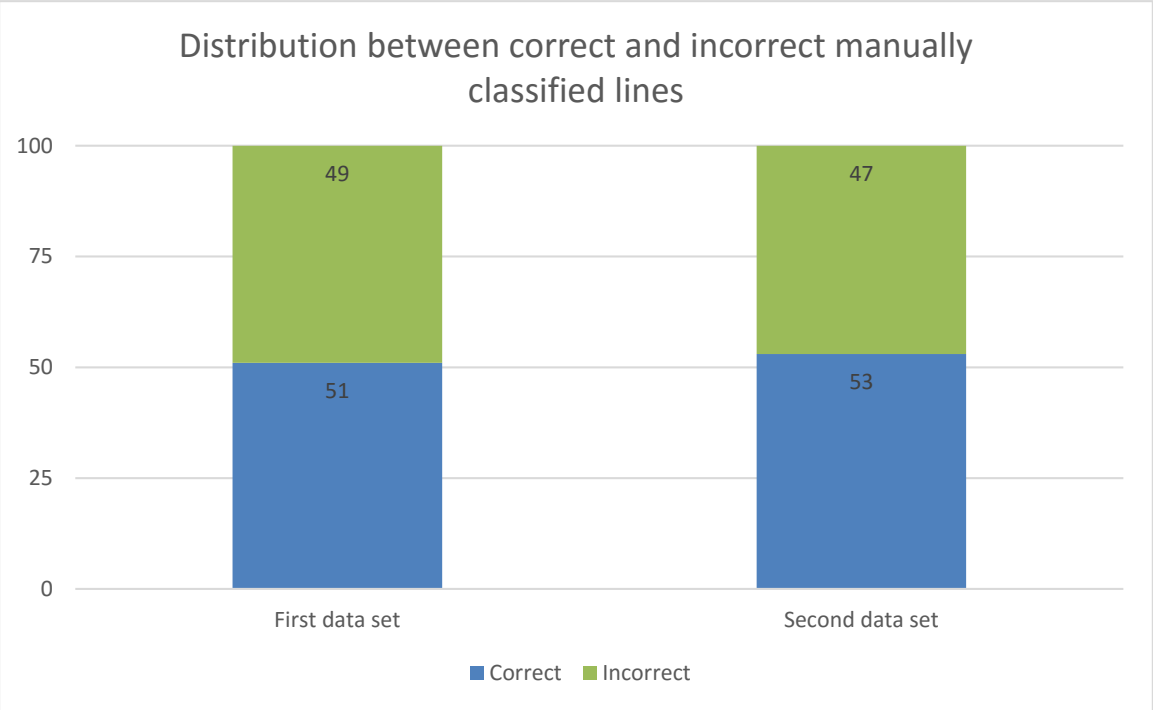

Figure 4.8 The distribution between correct and incorrect lines in the generated data

5 Analysis and discussion

In this chapter an evaluation of the results from chapter 4 in relation to the goals described in section 1.3 and the methods used in chapter 3.

5.1 Analysis of OCR

The result from chapter 4 makes an interesting point. The OCR-engines used performed well in terms of spelling mistakes. Both Tesseract and OCRopus had similar success rate and without training, Tesseract had a greater spelling correctness. But with training, OCRopus outperformed Tesseract but not to a greater degree. The training was time consuming and the errors corrected were such errors that a human would have no problem reading either of them. A computer on the other hand would have trouble reading both the trained and untrained versions and it led to the conclusion that Tesseract would be the better choice. Because training was not needed and the time spent training OCRopus would not yield a meaningful improvement.

The other problem was the structure of the invoices. The structure was not present in the plain text, which made it almost impossible for a human to understand which fields were linked. Even though the text was readable, the structure made it impossible to interpret the invoice correctly. If a human had trouble with the interpretation, it would be impossible for a computer to understand the text.

The structure problem could most likely be corrected using extraction templates which was covered in section 2.4.3. Some form of extraction template or rules of extraction was necessary to use a computer to process invoices correct.

5.2 Choice of machine learning algorithm

The choice of algorithm was Naïve Bayes. It presented acceptable results from the data presented in section 4.3. The decision to use Naïve Bayes proved to be adequate for the problem presented in chapter 1. Naïve Bayes was simple in terms of understanding and implementation, compared to the other machine learning algorithms. However, the results were good enough for the context of invoice handling. Accord.NET framework made it easy to implement Naïve Bayes into a working prototype with well documented examples.

Other algorithms such as SVM could improve the invoice handling process with an even higher correctness, but it could also present no improvement at all. Testing SVM against Naïve Bayes may yield interesting results. However Naïve Bayes was easier to implement, presented an adequate result and was used in similar solutions such as spam filters. Therefore, there was no reason for this thesis with invoice handling to use a more complex algorithm like SVM.

5.3 Machine learning on invoice handling

The prototype had the ability to present the user with the correctness of the inputted invoice lines, and also gave the option to classify the lines if the probability was less than 99%. As more data was added to the data set, the error rate and need of human supervision decreased. Using even more data than presented in chapter 4 would provide an even better correctness rating, which in turn would decrease the human workload even more.

What was non-trivial was the rules for each feature could vary between invoice lines. In a system without machine learning, each different rule for the invoice lines had to be defined in explicit code. Which meant having to release new versions of the system each time a new rule had to be applied. What the result of using machine learning in the prototype was when there were new invoice lines with new feature rules, the prototype did not need to be released as a new version. The prototype instead learned the new rules. By using this technique, it made the system a lot more durable to changes. It was important because changes always happened and with machine learning the change was taught, not decided beforehand.

A problem that might arise in the future when the collected data from an invoice line could be considered complete, was if the correct value for a feature in a stored invoice line changed. For example, if a much better price was negotiated, which differed from the data in the data set. It would result in problems with classification of the new data since the model was based on the data before the new negotiated value. The solution in this case would be to wipe the data set for the invoice line and start over with a new set. This created a problem with the model having to relearn everything and human supervision would be needed, until the data set contained sufficient data again.

If the training set used for classification was randomized according to rules and ranges, the training process would consist of randomizing data and the only manual time spent would be to correct the model. When the model had enough data to provide an adequate result it could be used for classification. How much data chosen for the model depended on the complexity of the rules and ranges for features, if the ranges were larger, more data had to be processed because of the increased possible permutations of the data. If the amount of data needed to make an adequate classification were reduced, the problem with wiping the data set when a rule changed would possibly be solved.

Another problem with machine learning in distributed systems, was that it was more complex and harder for others without any experience with machine learning to add to the solution. Some knowledge in machine learning was needed to add features to the project. In companies this might create a problem due to the technique being more complex than usual solutions. Also, there was no data presented in how Accord.NET handled massive data sets. If there was a problem with the time it took for the invoice to process, it may create problems for the users expecting a fast response.

5.4 Interpretations of machine learning result

What could be observed from the results from the measuring of correctness of invoice lines with the prototype, was that it clearly improved when more data was processed and classified. In an ideal case when recurring invoice lines never changed, the prototype would be an alternative to human supervision. What the result showed though was that the prototype needed manual classification in almost every case from the beginning, except from those where discrete values differed from the rule. As there were many features to consider in the assumption, a lot of manual classification was needed if the training data was not enough. Leading to longer time before the prototype could replace human supervision.

5.5 Machine learning production implementation

In case the prototype would be used in production, a lot of problems needed to be solved. One problem was how much human supervision was needed from the beginning, when the data sets were not enough. Another problem was the data had to be classified manually when an invoice line with no previously stored data was processed. One example of this would be when a new item was bought. This new invoice line had to always be classified manually in the beginning. It might create an initialization phase that made use of the prototype less valuable.

5.6 Society, economic, ethic and environmental aspects

The solution that is presented in this thesis will decrease the amount of repetitive work and will help decrease the workload of humans. As these processes can be automated, humans can put their attention on more important problems. By introducing automated systems with OCR and machine learning, companies can save money and use less resources on tasks that can be automated by computers. Thus humans can focus on more important things.

From an ethic point of view, by replacing tasks with automated systems people can lose their jobs. It will increase the unemployment rate which is a problem for society. This is something that needs to be evaluated when automating processes. The possible environmental impact of printing invoices for manual inspection for those who prefer to read on paper, is removed if the system can tell if the invoice is correct. When the process is automated the need of printing invoices to paper will be removed.

6 Conclusion

The OCR-engine test concluded that Tesseract was the better engine to use due to tedious training of OCRopus, and the spelling errors made by Tesseract still made the plain text readable. The time spent on training OCRopus did not present an adequate decrease of errors. Still, the problem with structure remained which made it impossible for a human to interpret the invoice correct. This concluded that the goal for OCR-engines was not reached as without a structure template, the scanning would not retain the invoice structure. It was however an important study nonetheless, because it proved that it was not possible to only use an OCR-engine without extraction templates for scanning invoice to usable plain text.

The prototype using machine learning with Naïve Bayes was able to automate the process of invoice handling. It improved over time but if there was any change in an invoice line already saved in the data set, it would present a problem to change this data, resulting in having to remove it and start over again. The use of machine learning instead of conventional programming methods could prove advantageous if a lot of similar data was processed.

The conclusion on machine learning with the prototype was that it could determine if data in an invoice was correct or not. It presented the user with a warning when the classification percentage was not high enough. It was also modular and easy to change the classification algorithm if there was a desire to test another algorithm.

The use of machine learning to solve similar problems could also automate processes within companies which could make them more efficient. This thesis showed that machine learning could be a valid choice of technology.

6.1 Future studies

Evaluate how an implementation of the extraction template algorithms from section 2.4.3 can be utilized and how it would improve OCR-extraction. As OCR-engines and invoice handling without templates does not work and does not give adequate results. And a study into extraction templates and how they would improve the result is recommended.

Future studies should include how problems similar to invoice handling could also be automated using the same technique. Machine learning is the future and presenting more studies on how it could improve different situations is important for future development. It would be interesting to analyze how much data Accord and Naïve Bayes could process before the classification becomes correct for all possible situations and the outcome of using even larger data sets than used in this thesis. A big data set might create other interesting challenges not covered in this thesis.

References

1. Dovev A. "Tesseract Github". [Online].; 2016 [cited 2016 05 09]. Available from: https://github.com/tesseract-ocr/tesseract/wiki/TrainingTesseract.

2. Springmann U, Najock D, Morgenroth H, Schmid H, Gotscharek A, Fink F. "OCR of historical printings of Latin texts: problems, prospects, progress". Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage (DATeCH '14). 2014.

3. Aoe Ji. "Computer Algorithms String Pattern Matching Strategies" Los Alamitos: IEEE Computer Society Press; 1994.

4. Haldar R, Mukhopadhyay D. "Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved". CoRR. 2011 Oct.

5. Hamza H, Belaïd Y, Belaïd A. "A case-based reasoning approach for invoice structure extraction". Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on. 2007 Oct: p. 327-331.

6. Aamodt A, Plaza E. "Case-based reasoning: Foundational issues, methodological variations, and system approaches". AI communications. 1994 Mar: p. 39-59.

7. Sorio E, Bartoli A, Davanzo G, Medvet E. "A Domain Knowledge-based Approach for Automatic Correction of Printed Invoices". Information Society (i-Society), 2012 International Conference on. 2012 Jun: p. 151-155.

8. Kulkarni P. "Knowledge Augmentation: A Machine Learning Perspective". In Reinforcement and Systemic Machine Learning for Decision Making.: Wiley-IEEE Press; 2012. p. 209 - 236.

9. Panigrahi KP. "A Comparative Study of Supervised Machine Learning Techniques for Spam E-mail Filtering". Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on. 2012 Nov: p. 506 - 512.

10. Yang Z, Nie X, Xu W, Guo J. "An Approach to Spam Detection by Naive Bayes Ensemble Based on Decision Induction". Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on. 2006 Oct: p. 861 - 866.

11. Rennie JDM, Shih L, Teevan J, Karger DR. "Tackling the poor assumptions of naive bayes text classifiers". ICML. 2003 Aug: p. 616-623.

12. Rish I. "An empirical study of the naive Bayes classifier". IJCAI 2001 workshop on empirical methods in artificial intelligence. 2001: p. 41-46.

13. John GH, Langley P. "Estimating continuous distributions in Bayesian classifiers". In Proceedings of the Eleventh conference on Uncertainty in artificial intelligence. 1995 Aug: p. 338-345.

14. McCallum A, Nigam K. "A Comparison of Event Models for Naive Bayes Text Classification". In AAAI-98 workshop on learning for text categorization. 1998 Jul: p. 41-48.

15. Campbell C, Ying Y. "Learning with Support Vector Machines" San Rafael: Morgan & Claypool Publishers; 2010.

16. Souza CR. "The Accord.NET Framework". [Online].; 2014 [cited 2014 04 12]. Available from: http://accord-framework.net.

17. Souza CR. "A Tutorial on Principal Component Analysis with the Accord.NET Framework". Technical Report. Federal University of Sao Carlos, Department of Computing; 2012.

Appendix A

Below the training data is presented for both data sets.

First data set

Amount	Currency	Period (days)	Units	Unit price	Product specifics	Correct
1219	SEK	86	1	1219	Fiber100/100	no
1211	SEK	82	3	403,6666667	Fiber100/100	no
1217	SEK	82	4	304,25	Fiber100/100	no
1212	SEK	98	1	1212	Fiber100/100	no
1213	SEK	88	3	404,3333333	Fiber100/100	yes
1215	SEK	92	4	303,75	Fiber100/100	yes
1219	SEK	94	1	1219	Fiber100/100	no
1216	SEK	88	4	304	iber100/100	no
1213	SEK	83	4	303,25	Fiber100/100	yes
1211	SKE	95	2	605,5	Fiber100/100	no
1219	SEK	82	3	401,3333333	Fiber100/100	no
1215	SEK	93	1	1215	Fiber100/100	yes
1211	SEK	83	3	396,6666667	Fiber100100	no
1215	SEK	90	3	405	Fiber100/100	yes
1219	SEK	98	4	304,75	Fiber100/100	no
1216	SEK	90	4	304	Fiber100/100	yes
1215	SEK	85	1	1215	Fiber100/100	yes
1212	SEK	97	3	404	Fiber100/100	yes
1217	SEK	87	3	405,6666667	Fiber100/100	yes
1211	SEK	98	1	1211	Fiber100/100	no
1216	SEK	94	3	405,3333333	Fiber100/100	yes
1216	KSE	84	1	1212	Fiber100/100	no
1216	SEK	85	4	304	Fiber100/100	yes
1212	SEK	83	4	303	Fiber100/100	yes
1211	SEK	97	2	605,5	Fiber100/100	no

Second data set.

Amount	Currency	Period (days)	Units	Unit price	Product specifics	Correct
1212	SEK	96	2	606	Fiber100/100	yes
1214	SKE	89	1	1214	iber100/100	no
1217	SEK	94	2	608,5	Fiber100/100	yes
1212	SEK	88	4	303	Fiber100/100	yes
1213	SEK	83	2	606,5	Fiber100/100	yes
1219	SEK	91	3	406,3333333	Fiber100/100	no
1216	SEK	85	4	304	Fiber100/100	yes
1219	SEK	94	3	406,3333333	Fiber100/100	no
1219	KSE	82	2	609,5	Fiber100/100	no
1212	SEK	88	1	1212	Fiber100/100	yes
1214	SEK	93	4	303,5	Fiber100/100	yes
1214	SEK	93	1	1214	Fiber100/100	yes
1215	SEK	83	4	303,75	Fiber100/100	yes
1216	KSE	90	1	1216	Fiber100/100	no
1211	SEK	88	1	1211	Fiber100/100	no
1219	SEK	92	1	1219	Fiber100100	no
1212	SEK	84	2	606	Fiber100/100	yes
1214	SEK	95	4	299,5	Fiber100/100	no
1216	SEK	82	3	405,3333333	Fiber100/100	no
1214	SEK	92	2	607	Fiber100/100	yes
1213	SEK	94	3	404,3333333	Fiber100/100	yes
1212	SEK	87	3	404	Fiber100/100	yes
1215	SEK	87	3	405	Fiber100/100	yes
1219	SEK	86	3	406,3333333	Fiber100/100	no
1211	SEK	84	1	1211	Fiber100/100	no

Appendix B

Below the data from each iteration is presented.

First data set with first iteration data.

Amount	Cur-rency	Period (days)	Units	Unit price	Product specifics	Prototype classifica-tion	Percentage	Manual classifica-tion
1211	SEK	89	3	403,6667	Fiber100/100	yes	68%	no
1213	SEK	86	1	1213	Fiber100/100	no	75%	yes
1216	SEK	85	2	608	Fiber100/100	no	93%	yes
1214	SEK	92	4	299,5	Fiber100/100	yes	92%	no
1211	SEK	85	4	302,75	Fiber100/100	yes	67%	no
1215	SEK	96	3	405	Fiber100/100	yes	84%	yes
1211	SEK	82	4	302,75	Fiber100/100	yes	58%	no
1218	SEK	95	3	406	Fiber100/100	yes	55%	yes
1214	SEK	89	3	404,6667	Fiber100/100	yes	92%	yes
1214	SEK	85	1	1214	Fiber100/100	no	71%	yes
1216	SEK	89	3	405,3333	Fiber100/100	yes	90%	yes
1216	SEK	90	2	608	Fiber100/100	no	87%	yes
1215	SEK	98	3	405	Fiber100/100	yes	76%	no
1217	SEK	97	3	405,6667	Fiber100/100	yes	65%	yes
1211	SEK	93	4	302,75	Fiber100/100	yes	63%	no
1213	SEK	96	2	606,5	Fiber100/100	no	85%	yes
1219	SEK	84	2	609,5	Fiber100/100	no	100%	no
1212	SEK	91	4	303	Fiber100/100	yes	83%	yes
1211	SEK	92	1	1211	Fiber100/100	no	93%	no
1211	SEK	91	1	1211	Fiber100/100	no	93%	no
1214	SEK	91	1	1214	Fiber100100	no	100%	no
1218	SKE	84	2	609	Fiber100/100	no	100%	no
1212	SEK	92	3	404	Fiber100/100	yes	81%	yes
1211	SEK	95	2	605,5	Fiber100/100	no	100%	no
1219	SEK	90	2	609,5	Fiber100/100	no	100%	no

First data set with second iteration data.

Amount	Cur-rency	Period (days)	Units	Unit price	Product specifics	Prototype classifica-tion	Percentage	Manual classifica-tion
1218	SEK	97	3	406	Fiber100/100	yes	66%	yes
1217	SEK	90	1	1217	Fiber100/100	no	75%	yes
1212	SEK	95	3	404	Fiber100/100	yes	76%	yes
1212	KSE	83	1	1212	Fiber100/100	no	100%	no
1214	SEK	83	4	303,5	Fiber100/100	yes	69%	yes
1214	SEK	90	1	1214	Fiber100/100	no	65%	yes
1212	SEK	93	1	1212	Fiber100/100	no	82%	yes
1214	SEK	84	1	1214	Fiber100/100	no	73%	yes
1214	SEK	96	2	607	Fiber100/100	yes	56%	yes
1215	SEK	88	1	1215	Fiber100/100	no	65%	yes
1215	KSE	92	3	405	Fiber100/100	no	100%	no
1211	SEK	86	4	302,75	Fiber100/100	no	61%	no
1216	SEK	93	1	1206	Fiber100/100	no	68%	yes
1219	SKE	97	4	304,75	Fiber100/100	no	100%	no
1219	KSE	89	3	406,333	Fiber100/100	no	100%	no
1214	SEK	89	4	303,5	Fiber100/100	yes	79%	yes
1214	SEK	98	2	607	Fiber100/100	no	51%	no
1215	SEK	98	4	303,75	Fiber100/100	yes	68%	no
1216	SEK	82	3	405,333	iber100/100	no	100%	no
1211	SEK	87	1	1211	Fiber100/100	no	91%	no
1214	SEK	97	1	1214	Fiber100/100	no	75%	yes
1211	SEK	92	2	605,5	Fiber100/100	no	73%	no
1214	SEK	82	1	1214	Fiber100/100	no	79%	no
1217	SEK	88	1	1217	Fiber100/100	no	76%	yes
1216	SEK	84	1	1216	Fiber100/100	no	75%	yes

First data set with third iteration data.

Amount	Cur-rency	Period (days)	Units	Unit price	Product specifics	Prototype classifica-tion	Percentage	Manual classifica-tion
1211	SEK	87	2	605,5	Fiber100100	no	100%	no
1215	SKE	93	3	405	Fiber100/100	no	100%	no
1216	SEK	96	1	1216	Fiber100/100	yes	75%	yes
1212	SEK	85	4	303	Fiber100/100	no	58%	yes
1211	SEK	96	1	1211	Fiber100/100	no	61%	no
1218	SEK	84	2	609	Fiber100/100	no	74%	yes
1215	KSE	88	1	1215	Fiber100/100	no	100%	no
1213	SEK	95	3	404,333	Fiber100/100	yes	74%	yes
1217	SEK	98	3	405,666	Fiber100/100	yes	65%	no
1217	SEK	84	1	1217	Fiber100/100	yes	64%	yes
1214	SEK	91	2	607	Fiber100/100	yes	59%	yes
1217	SEK	87	2	608,5	Fiber100/100	no	54%	yes
1219	SEK	89	3	406,333	Fiber100100	no	100%	no
1219	SEK	98	2	609,5	Fiber100/100	no	87%	no
1212	SEK	82	3	404	Fiber100/100	no	51%	yes
1218	SEK	91	1	1218	Fiber100/100	yes	60%	yes
1217	SEK	96	2	608,5	Fiber100/100	no	57%	yes
1218	SKE	83	3	406	Fiber100/100	no	100%	no
1219	SEK	98	3	406,333	Fiber100/100	no	68%	no
1219	SEK	86	4	304,75	Fiber100/100	no	76%	no
1215	SEK	93	4	303,75	Fiber100/100	yes	70%	yes
1213	SEK	84	1	1213	Fiber100100	no	100%	no
1212	SEK	87	1	1212	Fiber100/100	yes	60%	yes
1212	SEK	88	2	606	Fiber100/100	no	63%	no
1217	SEK	86	2	608,5	Fiber100/100	no	56%	yes

Second data set with first iteration data.

Amount	Cur-rency	Period (days)	Units	Unit price	Product specifics	Prototype classifica-tion	Percentage	Manual classifica-tion
1218	SEK	87	3	406	Fiber100100	no	100%	no
1218	SEK	87	4	304,5	Fiber100/100	no	55%	yes
1214	SEK	96	2	607	Fiber100/100	yes	97%	yes
1218	SEK	86	1	1218	Fiber100/100	no	98%	yes
1213	SKE	98	3	402,333	Fiber100/100	no	100%	no
1217	SEK	86	3	398,666	Fiber100/100	no	65%	no
1219	SEK	98	2	609,5	Fiber100/100	no	81%	no
1214	KSE	83	4	298,5	Fiber100/100	no	100%	no
1218	SEK	91	3	406	Fiber100/100	no	85%	yes
1218	SEK	85	2	609	Fiber100/100	no	56%	yes
1213	SEK	90	2	606,5	Fiber100/100	yes	97%	yes
1215	SEK	90	3	405	Fiber100/100	yes	78%	yes
1215	SEK	87	4	303,75	Fiber100/100	yes	95%	yes
1216	SEK	97	4	304	Fiber100/100	yes	92%	yes
1215	SEK	87	2	607,5	Fiber100/100	yes	95%	yes
1216	SEK	95	1	1216	Fiber100/100	no	88%	yes
1214	SEK	95	2	607	Fiber100/100	yes	97%	yes
1217	SEK	88	3	405,666	Fiber100/100	no	64%	yes
1218	SEK	82	2	609	Fiber100/100	no	59%	yes
1211	SEK	89	1	1211	Fiber100/100	no	76%	no
1216	SEK	82	1	1216	Fiber100/100	no	92%	no
1215	SEK	89	2	607,5	Fiber100100	no	100%	no
1218	SEK	95	2	609	Fiber100/100	yes	51%	yes
1212	SEK	83	1	1212	Fiber100/100	no	74%	yes
1214	SEK	97	2	607	Fiber100/100	yes	97%	yes

Second data set with second iteration data.

Amount	Cur-rency	Period (days)	Units	Unit price	Product specifics	Prototype classifica-tion	Percentage	Manual classifica-tion
1219	SEK	93	2	609,5	Fiber100/100	yes	82%	no
1214	SEK	87	2	607	Fiber100/100	yes	94%	yes
1217	SEK	82	2	608,5	Fiber100/100	yes	86%	no
1218	SEK	92	4	304,5	Fiber100/100	yes	86%	yes
1215	SEK	85	4	303,75	Fiber100/100	yes	91%	yes
1213	SEK	92	4	303,25	Fiber100100	no	100%	no
1216	SKE	84	3	405,333	Fiber100/100	no	100%	no
1217	SEK	89	1	1217	Fiber100/100	no	71%	yes
1215	SEK	97	3	405	Fiber100100	no	100%	no
1217	SEK	83	1	1217	Fiber100100	no	100%	no
1216	SEK	89	2	608	Fiber100/100	yes	93%	yes
1216	SEK	85	3	405,333	Fiber100/100	yes	69%	yes
1219	SEK	87	2	609,5	Fiber100/100	yes	79%	no
1212	SEK	88	3	396	Fiber100/100	yes	74%	no
1217	SEK	94	4	304,25	Fiber100/100	yes	90%	yes
1214	SEK	98	1	1214	Fiber100100	no	100%	no
1216	SEK	93	4	304	Fiber100/100	yes	92%	yes
1213	SEK	92	4	303,25	Fiber100/100	yes	94%	yes
1217	SEK	96	2	608,5	Fiber100100	no	100%	no
1214	SEK	86	3	404,666	Fiber100100	no	100%	no
1217	KSE	96	4	304,25	Fiber100/100	no	100%	no
1219	SEK	91	4	304,75	Fiber100/100	yes	79%	no
1213	SEK	90	3	404,333	Fiber100/100	yes	78%	yes
1217	SEK	88	3	405,666	Fiber100100	no	100%	no
1219	SKE	90	4	304,75	Fiber100/100	no	100%	no

Second data set with third iteration data.

Amount	Cur-rency	Period (days)	Units	Unit price	Product specifics	Prototype classifica-tion	Percentage	Manual classifica-tion
1215	SEK	94	1	1215	Fiber100/100	no	56%	yes
1218	SEK	89	2	609	Fiber100/100	yes	77%	yes
1217	SEK	87	1	1209	Fiber100/100	no	67%	no
1212	SEK	84	1	1212	Fiber100/100	no	61%	yes
1213	SEK	91	1	1213	iber100/100	no	100%	no
1214	KSE	86	4	303,5	Fiber100/100	no	100%	no
1218	SEK	86	4	299,5	Fiber100/100	yes	74%	no
1214	SEK	90	3	404,666	Fiber100/100	yes	75%	yes
1211	SEK	94	1	1211	Fiber100/100	no	60%	no
1214	SEK	86	1	1214	Fiber100/100	no	55%	yes
1212	SEK	84	3	404	Fiber100/100	yes	67%	yes
1213	SEK	94	2	606,5	Fiber100/100	yes	89%	yes
1214	SEK	88	3	404,666	Fiber100/100	yes	74%	yes
1216	SEK	91	1	1214	Fiber100/100	no	58%	no
1214	SEK	83	3	404,666	Fiber100/100	yes	67%	yes
1219	SEK	94	1	1219	Fiber100/100	no	83%	no
1215	SEK	90	4	303,75	Fiber100/100	yes	89%	yes
1219	SEK	90	1	1219	Fiber100/100	no	82%	no
1215	KSE	93	3	405	Fiber100/100	no	100%	no
1217	SEK	98	1	1217	Fiber100100	no	100%	no
1219	SEK	87	3	406,333	Fiber100100	no	100%	no
1212	SEK	83	3	404	iber100/100	no	100%	no
1212	SEK	86	3	404	Fiber100/100	yes	71%	yes
1211	SEK	95	3	403,666	Fiber100100	no	100%	no
1218	SEK	82	4	304,5	Fiber100/100	yes	67%	yes

TRITA 2016:53