Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Document Separation in Digital Mailrooms

by

J.Valberg

LIU-IDA/LITH-EX-A–15/056–SE

October 5, 2015

Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings universitet
581 83 Linköping

Linköpings universitet
Institutionen för datavetenskap

Final thesis

Document Separation in Digital Mailrooms

by

J.Valberg

LIU-IDA/LITH-EX-A–15/056–SE

October 5, 2015

External Supervisor: Anders Törnqvist-Svedbergh

Internal Supervisor: Olov Andersson

Examiner: Cyrille Berger

Abstract

The growing mail volumes for businesses worldwide is one reason why they are increasingly turning to digital mailrooms. A digital mailroom automatically manages the incoming mails, and a vital technology to its success is document classification. A problem with digital mailrooms and the document classification is separating the input stream of pages into documents. This thesis investigates existing classification theory and applies it to create an algorithm which solves the document separation problem. This algorithm is evaluated and compared against an existing algorithmic solution, over a dataset containing real invoices.

Acknowledgments

I take this opportunity to express gratitude to all the help I have received during the work of this thesis. Firstly I would like to thank my external supervisor Anders Törnqvist-Svedbergh who not only gave me the opportunity of this assignment but also helped out with all problems i encountered throughout it. I am also grateful for the help and guidance I have received from my internal supervisor Olov Andersson and for all his input on the thesis and alternative solutions.

Contents

List of Figures

List of Tables

Abbreviations

RS ..... Rough Separation

FS ..... Fine Separation

SS ..... Statistical Separation

RFS ..... Rough and Fine Separation

BSA ..... Batch Separation Accuracy

IBSA ..... Intra-Batch Separation Accuracy

BOSR ..... Batch Over-Separation Rate

BUSR ..... Batch Under-Separation Rate

Chapter 1

Introduction

This thesis was done on behalf and with the help of Readsoft AB in Stockholm. In this chapter the background and the problem is introduced, followed by a concrete problem formulation and scope. The chapter is then concluded with an outline of the report.

1.1 Background

Digital mailrooms are systems which automate the incoming mail processes for businesses. These systems use a wide range of technologies to achieve this automation, such as document capture, document classification and workflow applications. Document capture is the process of scanning and retrieving documents from electronic and ordinary mail, inputting them into the system. Document classification uses machine learning techniques to recognize the class of each document, in order to determine which workflow should continue handling it. The different workflow applications can then e.g. sort, redistribute or extract information from documents, depending on what is needed.

Document classification, or in other words text classification, is a widely researched area with many successful applications such as search engines and of course digital mailrooms. Classifying text typically requires text to be transformed into a uniform comparable feature representation, such as the Bag-of-Words vector. This representation allows Support Vector Machines, Naive Bayes or other classification algorithms to learn and distinguish between the different classes. The different feature representations and classification algorithms all have different advantages and disadvantages, effecting e.g. the resulting classification accuracy and speed. As such, the practical applications requirements and its domain have a major impact on the choice of classification system.

1.2 Problem Description

Documents are continuously input into digital mailrooms in batches. These batches contain any number of documents, and is structured as a sequence of pages. The classification system of a digital mailroom is used on a document level and as such the batch has to be separated into documents. Batch separation can be regarded as a type of change-point detection or clustering problem. A graphical representation of the batch separation problem is shown in Figure 1.1.

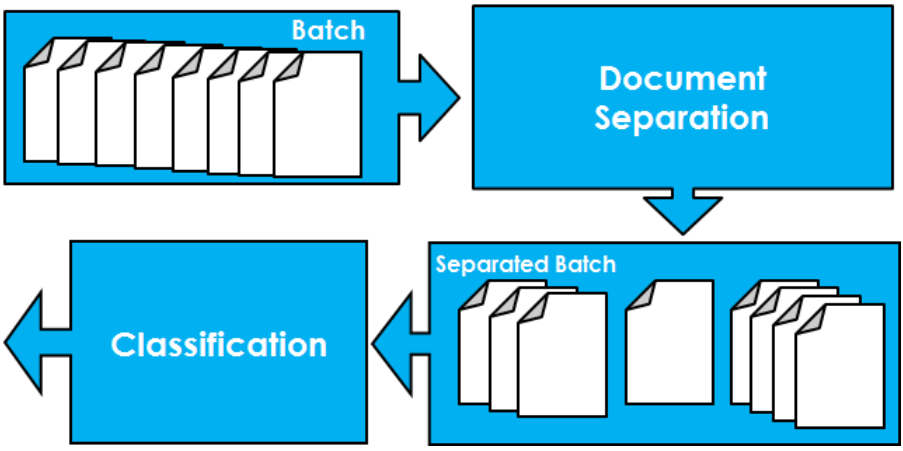


Figure 1.1: An example of the batch separation problem in a simplified and larger picture. The pages of the batch are separated according to their corresponding document.

Henceforth the term *separation point* will be used to denote a breakpoint in a batch, which separates one document from the other. Figure 1.2 shows these separation points in the same example scenario as Figure 1.1.

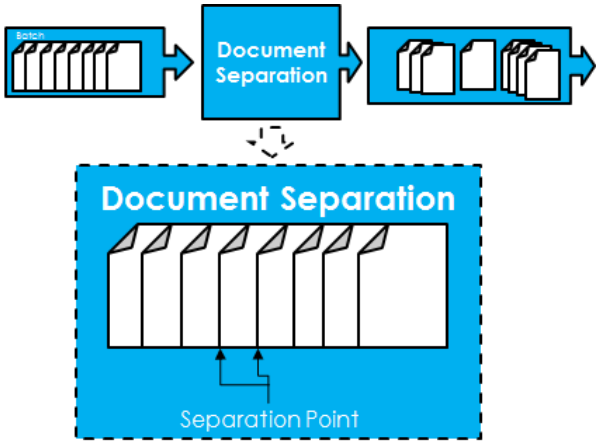


Figure 1.2: An example of the batch separation problem with separation points.

To further explain the document separation problem take the following example. A large company has a digital mailroom implemented to manage their mail volumes. The company receives invoices from its two suppliers daily and wants to handle these efficiently. These two suppliers' invoices are managed differently. The invoices from the first supplier, supplier A, is simply forwarded to the financial department. Invoices from the second supplier, supplier B, is also forwarded to the financial department but is also into their business system. To enable this one class for each supplier is created, class A and class B. The document capture phase gathers these invoice documents throughout the day from both electronic and regular mails. The document capture phase gathers a number of documents into a batch, before dispatching the it to the remaining parts of the system. Because of this the document classification phase now has to find the individual documents in the batch, which is done by the document separation.

One solution is to use a blank-page separation technique to find the separation points in the batches. This technique is done by manually adding a blank page between each document. This blank page can be added during the input phase of the digital mailroom, but it requires overhead, be it manual or machine. The blank page then acts as a separation point in a batch during the classification phase. This method is quite effective, as it can achieve a perfect separation rate, at the cost of an increased overhead.

There has been rather little research done regarding algorithmic batch separation. Collins-Thompson and Nickolov (2002) created a clustering-based technique, which had the advantage of not requiring a classification platform. This algorithm depends on certain features, such as the page number, to be extracted to achieve a higher separation accuracy.

Gordo et al. (2013) also derived an algorithm for separating a batch. This

method uses probabilistic reasoning to find the most likely valid documents in a batch. A requirement with this method is a classification platform, or a similar substitution, which gives a likelihood estimate for classes.

Due to the fact that there has been little research done into the batch separation domain there seems to be ample opportunities for other solutions. One approach could be to apply other classification- and clustering techniques to solve the batch separation problem. Furthermore it is interesting to investigate if any of these batch separation algorithms manage to achieve a performance which renders them usable in practice.

1.3 Problem Formulation

This thesis investigates and compares algorithms which solve the batch separation problem. To aid in this task we use an existing proprietary classification platform, provided by ReadSoft. The questions this thesis aims to answer are the following:

1. How can a batch of documents be separated algorithmically?
2. How well do the proposed algorithms as well as existing solutions perform in comparison to each other?
3. Are any of the examined solutions good enough to be useful in practice?

1.4 Scope

This thesis will not derive an algorithm for classifying documents. The thesis will instead use the provided proprietary classification platform and focus on the separation problem exclusively.

To solve the separation problem the thesis will examine and investigate solutions using similarity measures between documents and their text. Thus the thesis will focus on and use elements and concepts from the classification and information retrieval fields, experimenting with similarity measures and feature representations.

This thesis focuses on invoice documents and not general documents. The evaluation of the algorithms is performed with three suites of data received from ReadSoft.

1.5 Outline

1. **Introduction**
2. **Text Classification Theory**
   In the first chapter we examine the theoretical foundation of textual classification. This chapter investigates feature representations, similarity measures and some common classification algorithms.
3. **Document Separation Algorithms**
   The thesis then continues with descriptions of three different document separation algorithms.
4. **Experiments**
   This chapter contains detail about the the experiments which were performed. The chapter contains descriptions about the used performance measures, the suites of data which were used and also details about the experiments which were performed.

5. **Results**
   In this chapter, all the results from the experiments are presented. These results are presented using several bar charts.

6. **Discussion and Conclusion**
   The thesis is then concluded with a discussion about the evaluated solutions and discussions about improvements as well as possible future work.

Chapter 2

Text Classification Theory

This chapter starts with an overview of text classification, and how it is generally performed. The chapter then continues by giving more in depth details about the following areas; feature representations, feature selection, classification algorithms and finally algorithms for measuring the similarity of text.

2.1 Classification Overview

Classifying text can be done through either supervised or unsupervised machine learning algorithms. Supervised classification trains algorithms with pre-labeled sets of data. Unsupervised classification generates clusters of documents, where each cluster contains documents that are similar to each other. This requires no human interaction and can be completely autonomous. Text classification is generally done through the following steps;

1. Data is extracted from texts to create a simplified representation. The data elements which are extracted are known as features. The simplified representation is known as a feature representation.

2. The feature representation generated from the previous step is typically very large and high-dimensional. The next step of the process reduces the feature representation by removing redundant information. This is known as either feature selection or extraction.

3. After a document has been transformed into a feature representation it is input to a algorithm which classifies it. There exists a multitude of classification algorithms, such as the support vector machine and the k-nearest neighbor algorithm.

2.2 Feature Representations

Texts which are used in classification platforms are transformed into uniform comparable structures known as a feature representations. This section describes two different types of feature representations, vectors and graphs.

2.2.1 Vector Space Model

A fundamental concept within classification- and information retrieval areas is the vector space model developed by Salton (1968). The model is based on the notion that objects, such as textual documents, can be represented by algebraic models. In particular that objects are represented as vectors in a common space. The implications of the vector space model is that it transforms abstract objects into comparable vectors.

$$d_i = (w_{1,i}, w_{2,i}, \ldots, w_{m,i}) \tag{2.1}$$

An example of the vector space model is found in Equation 2.1. Document i is transformed into the feature vector d_i. The value $w_{m,i}$ is the frequency of word w_m in document i. This feature vector is also known as the *Bag-of-Words* (Manning et al., 2009). Figure 2.2.1 shows an example of this.

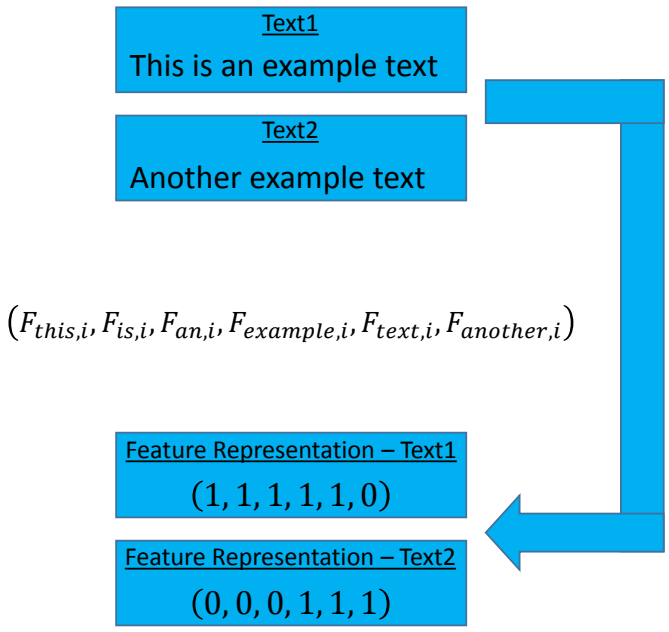


Figure 2.1: A figure showing the Bag-of-Words feature vectors in the same vector space, for two examples texts. The variable $F_{a,b}$ denotes the frequency for the word a in text b.

Below follows a list of common vector-based feature representations for documents which previous literature have used.

Bag-of-Words
Also known as the word frequency vector. This was described above.

Length Normalized Frequency Vector
The length normalized frequency vector builds upon the Bag-of-Words feature representation. The vector contains the frequency of a word divided by the number of words in the document (Wajeed and Adilakshmi, 2011). If the same notion is used as in Equation 2.1 and the number of words in a document is c, the length normalized frequency vector can be described as

$$d_i = (\frac{w_{1,i}}{c}, \frac{w_{2,i}}{c}, \ldots, \frac{w_{m,i}}{c}) \tag{2.2}$$

n-gram
The n-gram feature representation uses a combinations of n words as features. If a document i contains the words w_j the n-gram feature vector d_i can be described as

$$d_i = (f_1, f_2, \ldots, f_n) \tag{2.3}$$
$$f_j = w_j w_{j+1} \ldots w_{j+n} \tag{2.4}$$

The order $w_j w_{j+1} \ldots w_{j+n}$ is in no particular fashion. It could simply be a left-to-right order of the words in a sentence. The value of a f_j could be the frequency of that feature in document i.

Tf-idf Vector
Tf-idf stands for term frequency-inverse document frequency. The purpose of this feature representation is to lower the impact of common redundant words while increasing the impact of rare, unique words (Manning et al., 2009, p. 117-118). The features of this representation are the words of the document. The feature vector is calculated as

$$idf_w = log(\frac{N}{df_w}) \tag{2.5}$$

$$f_w = tf_w \cdot idf_w \tag{2.6}$$

$$d_i = (f_{w_1}, f_{w_2}, \ldots, f_{w_j}) \tag{2.7}$$

Here, w denotes a word, df_w denotes the number of document where word w occurs, N denotes the number of documents and tf_w is word frequency of word w. Furthermore, d_i denotes the feature vector for document i, where w_j denotes the words in document i.

2.2.2 Graph-based Representation

The common feature representation Bag-of-words assumes an independence property between words of documents. The representation ignores the sequence in which words occur. This simplification has been proven to be effective, but incorporating the sequence into a feature representation can improve the accuracy. For example the words "social", "economy" are different from the concatenation "social economy", and have different contextual meanings. To incorporate the contextual and spatial meaning of terms Chow et al. (2009) proposed a graph-based feature representation.

Sequential Graph

A document can be represented by a graph $G = (V, E, \phi, \theta)$. V denotes the vertices of the graph, which represent a unique word in a document. E denotes the edges of the graph, which is a sequential connection between a documents words. The function $\phi : V \rightarrow tf_{V,d}$ assigns the word frequency attribute to the vertices. Lastly, $\theta : E \rightarrow c_{E,d}$ assigns the connection frequency attribute to the edges.

The graph can be either directed or undirected, but Chow et al. (2009) proposes the undirected graph. The reason for this was, to quote; "In many cases the sequence of words are convertible, although it conveys the same semantics for human language. For example, "computer science" can be expressed as "science of computer", which delivers the same meaning."

Projection of Graph to Vector

A graph $G = (V, E, \phi, \theta)$ can be represented simply by using the adjacency matrix denoted by $A^k = [A^k_{ij}]$ for a graph G^k. An element $A^k_{ij} = tf_{ij}$ in the adjacency matrix is equal to the word frequency between word i and j. Chow et al. (2009) states that using just the adjacency matrix to calculate similarity leads to a waste of both time and space, because the matrix is so sparse. A solution to create a sparser representation, while maintaining the statistical properties is to apply principal component analysis or linear discriminant analysis.

2.3 Feature Selection & Extraction

The feature representation space dimension will grow rapidly when classifying more and larger documents. To combat this, feature selection or extractions methods are employed to remove unnecessary and redundant features from the feature representations. Feature selection methods filter features from the representation, while feature extraction methods transforms the feature representation from a larger feature space to a smaller one.

2.3.1 Stemming

Stemming, or lemmatization, is a variant of feature selection which reduces words to their base forms. As an example, stemming would transform the words "walking" and "walked" into the base word "walk". This ultimately decreases the amount of features, while still maintaining the same information. (Manning et al., 2009)

2.3.2 Stop-Words

Another feature selection approach is to simply remove all the common words, such as "the" or "what", from the feature representation. These words are typically given in a stop-words list, or can be derived by removing words with high global frequency (Liu et al., 2005; Manning et al., 2009).

Manning et al. (2009) argue that in the information retrieval field it used to be popular to create huge lists of stop-words that were ignored, but in more recent time however these stop-word lists are decreasing in size and may not even be used. However the authors Ikonomakis et al. (2005) claim that ignoring or decrease the impact of these stop words can possibly increase the classification accuracy. Ignoring these stop words also helps reducing the curse of dimensionality (Ikonomakis et al., 2005; Bandyopadhyay and Saha, 2013, p. 7).

2.3.3 Feature Clustering

Feature clustering is a method of feature extraction. This method uses clustering techniques to group similar features together. These groups are then used as new features, transforming a feature representation into a new feature space and lowering the dimensionality. Feature extraction is practically recognized as more effective then feature selection, but is more computationally costly (Jiang et al., 2011).

One noteworthy method of clustering features for text classification is *the information bottleneck method* (IB). The features used in the IB method are words. The IB method is based on the idea that the similarity between two features is determined by comparing their individual joint probability distribution of occurrences over all the classes. If the two features share the same probability distribution, they contribute equally as much information in determining the class, and as such can be clustered together. (Slonim and Tishby, 2001)

2.4 Machine Learning Algorithms

Inductive machine learning is learning from examples (Kotsiantis et al., 2006). There are numerous different machine learning approaches to classification. Broadly speaking they can be categorized into logic-, perceptron-,

statistics-, instance- and support vector machine based learning. (Kotsiantis et al., 2006) Some examples of algorithms from the different categories are as follows;

- Tree Based
  - Decision Tree
- Perceptron Based
  - Single layer perceptron
  - Multi layer perceptron
- Probabilistic Based
  - Naive Bayes Classifier
- Instance Based
  - k Nearest Neighbors
- Support Vector Machines

When choosing a machine learning algorithm one has to take questions and aspects into account such as; How much training data is available? (Manning et al., 2009) How fast should it be able to classify? How accurate does it have to be? Are the features discrete or continuous? (Kotsiantis et al., 2006) The authors Kotsiantis et al. (2006) argue that

> The key question when dealing with ML classification is not whether a learning algorithm is superior to others, but under which conditions a particular method can significantly outperform others on a given application problem. (Kotsiantis et al., 2006)

For further details and in-depth research the authors Caruana and Niculescu-Mizil (2006) compare these algorithms against each other on well known sets of data.

2.4.1 Naive Bayes Classifier

The Naive Bayes Classifier (NBC) is one of the simplest classification algorithms. NBC calculates the probability for the class C given the observed set of words W. The class C can assume a label from a set $c = c_1, \ldots, c_2$, i.e. this is not only a binary classifier. The random variable W is a set of random variables $W_1, \ldots, W_2$ representing individual features. The algorithm builds upon Bayes Theorem, see Equation 2.8. (Smola and Vishwanathan, 2008, p. 22-23)

$$p(C|W) = \frac{p(W|C)p(C)}{p(W)} \tag{2.8}$$

The probability $p(W)$ can be difficult to calculate, and is regarded as constant for different calculations of C. To avoid estimating this probability, the fraction R between $p(C_i|W)$ and $p(C_j|W)$ is calculated, yielding Equation 2.9.

$$R = \frac{p(C_i|W)}{p(C_j|W)} = \frac{p(W|C_i)p(C_i)}{p(W|C_j)p(C_j)} \tag{2.9}$$

The Equation 2.9 can be further simplified by an independence assumption. The features are all regarded as independent of each other, and only dependent on the class. Even though this assumption is clearly wrong, it yields good results (Kotsiantis et al., 2006). This final assumption results in the simpler Equation 2.10.

$$R = \frac{p(W|C_i)p(C_i)}{p(W|C_j)p(C_j)} = \frac{p(C_i)\prod_q p(W^q|C_i)}{p(C_j)\prod_q p(W^q|C_j)} \tag{2.10}$$

Equation 2.10 only requires $p(C_i)$ and $p(W^q|C_i)$ to be estimated during the training phase.

A major advantage with NBC is the low computational training time which is required. According to the comparison table given by the authors Ikonomakis et al. (2005) the NBC has, compared to the other algorithms, a high tolerance of missing values, a high speed of training and a good classification speed. On the negative side the NBC has a low tolerance of redundant and irrelevant features, and a low classification accuracy. (Ikonomakis et al., 2005) But the low classification accuracy of the NBC is a bit disputed. According to Rish (2001) the NBC is effective in practice because even though the probability estimates may be inaccurate, the classification decision may be correct.

2.4.2 Support Vector Machine

The support vector machine (SVM) algorithm is a modern ML algorithm developed by Cortes and Vapnik (1995). SVMs constructs a hyperplane, see Figure 2.2 for an example, that attempts to separate the data by maximizing the margin between a subset of the data points denoted "support vectors" and the plane. SVMs also allow the incorporation of different kernels that transform the classification task to a higher dimensional space. This can improve classification accuracy in cases where the data is not well separated by a plane. Popular such kernels include polynomials and radial basis functions.

The support vector machine can achieve a high classification accuracy, but it requires an expensive training, requiring more time and data then e.g. NBC and kNN classification. The SVM and artificial neural networks are

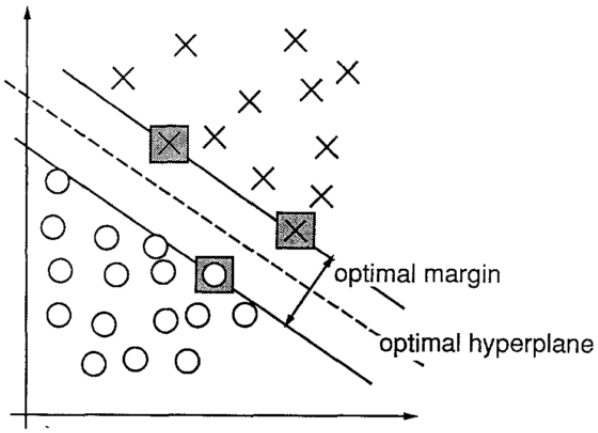


Figure 2.2: An example of linearly separable data in 2-dimensional space, from Cortes and Vapnik (1995).

generally said to deal with continuous and multi-dimension features better then the other classification algorithms. SVM is a binary classifier, but it can be extended to a multiclass classifier. (Ikonomakis et al., 2005)

2.4.3 kNN Classification

One of the simplest instance based classification algorithms is the k nearest neighbor (kNN). The kNN algorithm is built upon the assumption that instances of the same class are similar to each other, and are therefore located close to each other in the feature space.

The kNN algorithm does not need a training phase, because the instances which are used to measure against are simply stored. When classifying new data the kNN measures the similarity between the data and the stored instances. This is also known as *lazy learning*. The k closest resembling instances are chosen and the class which is the most frequent in these k selected instances is the class to which the data is assigned. (Kotsiantis et al., 2006; Wajeed and Adilakshmi, 2011; Smola and Vishwanathan, 2008, p. 24-26)

There are a lot of different ways of measuring the similarity of two instances. Typically the instances are represented by some vector, and the similarity can be measured by some distance between the two locations in the space. Some examples are the Manhattan distance, see Equation 2.11, and the Euclidean distance, see Equation 2.12. (Wajeed and Adilakshmi, 2011) There are many other ways of measuring similarities between instances. The topic of similarity will be further discussed in Section 2.5.

$$D(x, y) = \sum_t |x_t - y_t| \tag{2.11}$$

$$D(x, y) = \sqrt{\sum_t |x_t - y_t|^2} \tag{2.12}$$

kNN uses a lot of storage compared to other ML algorithms. This can be explained by the fact that it uses lazy learning, where it simply stores everything. Unlike NBC the kNN can handle continuous features, but it has a lower classification speed. (Kotsiantis et al., 2006)

2.5 Similarity

Measuring the similarity between documents is often done pairwise. In the thesis *Analysing Document Similarity Measures*, written by Grefenstette and Pulman (2009), the authors try to establish a scientific notion of similarity with a basis from areas such as philosophy and cognitive science.

Grefenstette and Pulman (2009, p. 19) argue that two objects which are similar will share properties. It is important to note that when measuring similarity one has to regard contextually significant properties, otherwise it leads to a faulty comparison. As an example, take the situation when one wants to measure the similarity between two books. If the similarity context is the content of a book, a poor similarity estimate will be given if the only properties that are analyzed and compared are connected to the images on the front page. Measuring the similarity between two documents can be regarded as the comparison of what features they contain, for example in the bag-of-words feature representation shared properties could be word frequencies. Sometimes certain properties are more important then others in determining the similarity, and thus should have a bigger weight or impact (Grefenstette and Pulman, 2009).

Measuring the similarity between objects, and more importantly documents, can be quite difficult. The choice of feature representation for documents has a big impact on how the similarity measure performs.

2.5.1 Similarity Measures

Euclidean Similarity

The Euclidean similarity measure is based on using the Euclidean distance between two feature vectors, see Equation 2.13.

$$S(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_t |x_t - y_t|^2} \tag{2.13}$$

The similarity value $S(\mathbf{x}, \mathbf{y})$ is inversely proportional to the feature vectors real similarity. The best similarity is achieved when $\mathbf{x} = \mathbf{y}$, yielding a similarity $S(\mathbf{x}, \mathbf{y}) = 0$

A drawback of the euclidean similarity measure is the sensitivity of disproportionate differences in the feature values of $\mathbf{x}$ and $\mathbf{y}$. This increases the distance between the vectors, resulting in a worse similarity, when this may not be the case.

Cosine Similarity

The cosine similarity measure uses the angle between the feature vectors. This measure is derived from the dot product of vectors, as shown below.

$$\mathbf{x} \bullet \mathbf{y} = ||\mathbf{x}|| \cdot ||\mathbf{y}|| \cdot cos(\theta)$$
$$S(\mathbf{x}, \mathbf{y}) = cos(\theta) = \frac{\mathbf{x} \bullet \mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||}$$

The effect of using the angle between vectors as a measure is that the length of a vector has no effect on the similarity, which is the case in the euclidean similarity measure.

Similarity Measure for Text Processing

Another similarity measure for documents was presented by Lin et al. (2014) and is called Similarity Measure for Text Processing (SMTP). The measure is based on the presence or absence of different features giving certain similarity scores. This similarity measure is calculated according to Equation 2.14.

$$S(\mathbf{x}, \mathbf{y}) = \frac{F(\mathbf{x}, \mathbf{y}) + \lambda}{1 + \lambda} \tag{2.14}$$

$$F(\mathbf{x}, \mathbf{y}) = \frac{\sum_{j=1}^{m} N_*(w_{j,x}, w_{j,y})}{\sum_{j=1}^{m} N_\cup(w_{j,x}, w_{j,y})} \tag{2.15}$$

$$N_*(x_j, y_j) = \begin{cases} 0.5 \cdot (1 + exp(-(\frac{w_{j,x} - w_{j,y}}{\sigma_j})^2)), \\ \qquad\qquad if\ w_{j,x}, w_{j,y} > 0 \\ 0, \ if\ w_{j,x} = 0\ and\ w_{j,y} = 0 \\ -\lambda, \ otherwise \end{cases} \tag{2.16}$$

$$N_\cup(w_{j,x}, w_{j,y}) = \begin{cases} 0, \ if\ w_{j,x} = 0\ and\ w_{j,y} = 0 \\ 1, otherwise \end{cases} \tag{2.17}$$

The feature representations $\mathbf{x}$ and $\mathbf{y}$ are vectors of features. The σ_j value is the standard deviation for non-zero features w_j, and λ is a constant.

This similarity measure is derived on properties, which are used as an inspiration for the similarity measure in the fine separation algorithm described in Subsection 3.5.5. These properties are (Lin et al., 2014):

1. The absence or presence of features is more important when determining the similarity then the difference in magnitude of a feature.

2. Documents are more similar if the difference in magnitude of a factor is smaller. As an example; Two documents are transformed into the feature vector $\mathbf{x}$ and $\mathbf{y}$. The documents are more similar if the vectors are $\mathbf{x} =< 3, 0 >$ and $\mathbf{y} =< 4, 0 >$ than $\mathbf{x} =< 4, 0 >$ and $\mathbf{y} =< 20, 0 >$.

3. As the number of presence absence features increases, the documents similarity degree should decrease. An example; the feature vectors $< 1, 0, 1 >$ and $< 1, 1, 0 >$ should have a lower degree of similarity than the vectors $< 1, 0, 0 >$ and $< 1, 0, 1 >$.

4. Documents have the least degree of similarity to each other if they have no common non-zero or zero features with each other.

The SMTP has a good overall classification accuracy. When used in a single label classification purpose, with a word frequency feature representation and the SL-kNN classifier, it outperforms both the euclidean and cosine similarity measures. SMTP generally outperforms other similarity measures on kNN multi-label classification k-means clustering. (Lin et al., 2014)

Chapter 3

Document Separation Algorithms

This chapters presents the solutions for the document separation problem. It starts off by first presenting two existing solutions. The first existing algorithm is described in the Related Algorithm section, because it was not implemented and only used as inspiration. The other algorithm is described in the statistical separation algorithm section. The chapter then continues by presenting the new document separation algorithm which was developed specifically for this thesis, called the rough & fine separation algorithm. The chapter concludes by describing how the provided classification system was used.

3.1 Related Algorithm

The authors Collins-Thompson and Nickolov (2002) evaluated a clustering based technique for finding or separation points in a stream of pages. The technique was based on the premise that documents can be found by clustering similar pages, and that there were clear separating discrepancies on the separation points.

Achieving good separation results with the algorithm thus requires a good feature representation and similarity measure. The feature representation needs to contain the necessary information required to distinguish instances of documents and a similarity measure which can accurately estimate this.

3.1.1 Feature Representation

Collins-Thompson and Nickolov (2002) used a variety of different features from the documents. The features used by the authors contain information about; document structure, layout structure and text similarity.

The text similarity features contained information about the words found in a document. For every page a Bag-of-Words was used, where common words were stemmed.

The document structure features was meta-information extracted from the document. These features were derived from the headers, footers and page numbers of the documents.

Finally, the layout structure features incorporated information about e.g. word height, character width, line spacing and line indentation.

3.1.2 Similarity Measures

Every feature representations required their own similarity measure calculations. To measure the similarity for text features the cosine similarity measure was used. The other feature representations utilized unique similarity measures which Collins-Thompson and Nickolov (2002) had derived for each representation. The resulting similarity factors were then used by a linear classifier, in their case a SVM, which estimated the probability that two pages are related.

3.1.3 Clustering Algorithm

To separate the pages a two-phase bottom-up clustering algorithm was used. The first phase begins with all pages in their own clusters, then iteratively combining the closest page clusters together using a single linkage criterion. The single linkage criterion means that the distance between two clusters is the distance between their two most similar pages, which was calculated by their similarity measure. The second phase relaxed the the linkage criterion, which allows the remaining single pages to be clustered into the appropriate nearby clusters.

The two phases of the algorithms are used to handle a situation where a document contains a small amount of pages which differ significantly from the other pages. The relaxation in phase two allows the significantly differing pages to merge into their document clusters.

3.2 Statistical Separation Algorithm

Gordo et al. (2013) claim that the existing page stream segmentation solutions rely heavily on the homogeneity of pages in document classes. The solutions neglect the use of classification and the knowledge which can be exploited from it. To examine this new possibility, Gordo et al. (2013) created a new algorithm, henceforth called the statistical separation (SS) algorithm.

3.2.1 Feature Representation

In order to incorporate classification into the separation process the authors created a feature representation for multi-page documents. There is no set length for the multi-page documents. The authors argued that explicitly encoding the page order into the feature representation can be inefficient and opted to use a simpler and more general model. Let $P = \{p_1, p_2, \ldots, p_n\}$ denote the pages of a multi-page document. The pages p_i are represented by a vectorial feature-representation with a dimension d, i.e. $p_i \in \mathbb{R}^d$. The authors do not specify any specific feature representation to use, but they briefly mention the Bag-of-Words as a possible choice. The aggregated feature representation for a multi-page document P, denoted $\phi(P)$, is created by Equation 3.1.

$$\phi(P) = \frac{1}{\mathrm{n}} \sum_{i=1}^{\mathrm{n}} p_i \tag{3.1}$$

This equation allows the dimension for the feature representation of a multi-page document to be independent of the length of the document, while still maintaining vital information. Despite the crudeness, or simplicity, of the feature representation it proved to be equal or even better then more complex representations (Gordo et al., 2013).

3.2.2 Validity Estimation

To evaluate different separations of a page stream, a document validity estimator was used. The purpose of the validity estimator was to assess whether a given set of pages form a valid document of one of the classes. This validity estimator, denoted V, was defined as a function accepting a set of pages P and a feature representation transformer ϕ. To assess the validity of a proposed document the estimator used probabilistic reasoning. The number of k classes which exist are defined as $C = \{c_1, c_2, \ldots, c_k\}$, the feature representation of P is defined as $x = \phi(P)$ and the length of the document is defined as $n = |P|$. The resulting function was defined as Equation 3.2.

$$V(P, \phi) = \sum_{c \in C} p(x, n|c) \tag{3.2}$$

$V(P, \phi)$ is the probability distribution that the feature representation x of length n was generated by class c. To avoid an ambiguity issue, the authors proposed that one should maximize the difference between the first and second most likely classes instead, as defined in Equation 3.3.

$$V(P, \phi) = \max_{\hat{c} \in C} \{p(x, n|\hat{c})\} - \max_{c \in C \setminus \{\hat{c}\}} \{p(x, n|c)\} \tag{3.3}$$

This equation was further simplified by assuming an independence property between x and n and applying Bayes rules. The probability $p(x, n|c)$ was then written as Equation 3.4.

$$p(x, n|c) \propto \frac{p(c|x)p(n|c)}{p(c)} \tag{3.4}$$

The probability $p(c|x)$ was estimated from the classification results. Both the probability $p(n|c)$ and $p(c)$ were estimated from the training data used by the classification algorithm. The probability $p(x)$ was disregarded because it was assumed to be uniform.

3.2.3 Separation Algorithm

A stream of n pages was denoted $S = \{p_1, p_2, \ldots, p_n\}$. This stream is to be split into an unknown number of documents of unknown lengths which belong to one of the k classes. $S_{c:v} = \{p_c, \ldots, p_v\}$ was defined as a subsequence of pages from S from page c to page v, where $1 \leq c < v \leq n$. The algorithm was defined as P_j, seen in Equation 3.5.

$$P_j = \begin{cases} 0 & \textit{if } j = 0 \\ \underset{i<j}{max}(P_i + \log V(S_{i+1:j}, \phi)) & \textit{otherwise} \end{cases} \tag{3.5}$$

This algorithms calculates the best separation score of a stream up until page j, similar to a dynamic programming algorithm. To find the separation points of the whole stream up until n, calculate P_n and keep track of the selected separations.

3.3 Rough & Fine Separation Algorithm Overview

The rough & fine separation (RFS) algorithm is the new separation algorithm. This algorithm uses two steps to locate the correct separation points. This section provides a general overview of how these two steps function. The steps themselves are implemented as two different algorithms, and described in their own respective sections.

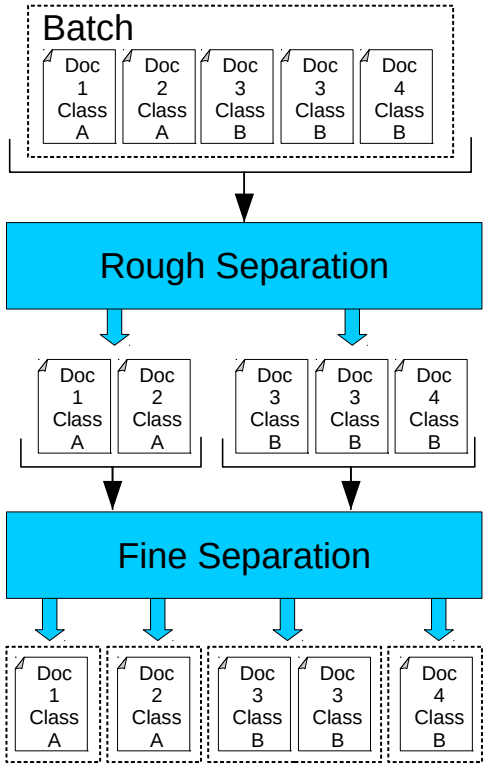


Figure 3.1: The conceptual flow of the RFS algorithm.

3.3.1 Rough Separation

The first step of the the RFS is called rough separation (RS). This step performs a class-wise separation of the batch, dividing the batch based on the class of the pages, as can be seen in Figure 3.1.

3.3.2 Fine Separation

The second step performs a document separation on each class separation individually. It checks if a class separation is one document or if it is actually

multiple documents of the same class. The result of this is a batch which is document separated. The second step is called fine separation (FS), and is seen in the Figure 3.1.

3.4 Rough Separation Algorithm

The purpose of the RS algorithm is to separate the pages of a batch based on their classes.

3.4.1 Separation Algorithm

The algorithm uses a greedy approach. It iterates the sequence of page data from left to right, and at every page it evaluates the hypothesis "A separation point should be placed here". The hypothesis evaluation is based on comparing the current separation subsequence's classification result with the classification result of the next page. If the two classification results differ, such as the next page is more similar to another class, a separation point is added. The pseudocode for the heuristic separation algorithm is displayed by Algorithm 1.

Algorithm 1 Heuristic Rough Separation Algorithm

```
1:  function RoughSeparation(Pages)
2:      separations ← empty
3:      currentSeparation ← empty
4:      for currentPage in Pages do
5:          sepScore ← Classify(currentSeparation)
6:          pageScore ← Classify(currentPage)
7:          if ShouldSeparate(sepScore, pageScore) then
8:              separations.Add(currentSeparation)
9:              currentSeparation ← NewSeparation()
10:         currentSeparation.Add(currentPage)
        Return separations
```

The variable *currentSeparation* is a set of pages and *separations* is a set containing sets of pages. The lines $5 - 6$ use the previously described classification algorithm. The function "ShouldSeparate" on line 7 checks if the two classifications are of the same class as previously described.

The motivation behind this greedy approach is based on the fact that the pages of invoices should individually conform well to their class, because they typically contain the same information in headers, footers and so forth. Also, subsequences of pages which have been successfully classified and separated are not going to change class. Searching the entire solution space is therefore unnecessary.

3.4.2 Analysis & Limitations

The RS algorithm only needs to iterate a sequence once and thus has time complexity of $\mathcal{O}(n)$, where n is the number of pages in a batch. In order for the algorithm to work every individual page has to be able to match and receive a good classification result. This might not always be the case because sometimes a document may contain pages with some graphs, images, or perhaps an unrelated table.

3.5 Fine Separation Algorithm

The purpose of the FS algorithm is to separate each class separation into document separations. The FS algorithm uses a similar approach as Gordo et al. (2013), with the same separation algorithm but with a different approach regarding the feature representation and validity estimation.

3.5.1 Approach

The class of all pages are equal and known during the document separation. As such, the algorithm does not have to evaluate over multiple classes. Instead, the algorithm tries to distinguish one document from another by looking for repetitions of data. The repetitions of data which are important are those which are unique for documents, such as the beginning and end data of the class. To find these repetitions the sequences of words are used.

3.5.2 Motivation

For formal documents such as invoices, business memos and mail the data is bound to contain some unique common words. These formal documents contain some unique sequence of data, be it by starting with some header containing e.g. corporation name, ending with some typical footer or even some unique word-sequence in the middle. Therefore, the information about these sequences of words need to be encoded in the feature representation.

Another possible approach would be to use the lengths of a classes documents. This approach is not useful in this scenario, as invoices of specific classes can vary greatly in lengths.

3.5.3 Relational Feature Representation

A sequence of words can be represented by the relations the words have to each other. As an example, take case where a word sequence is ”Lorem ipsum dolor”. ”Lorem” only has a right relation with the two other words, while ”ipsum” has a left relation with ”Lorem” and a right relation with ”dolor”, and ”dolor” only has left relations with the other two words. The bi-gram feature representation, discussed in section 2.2.1, can be used to represent this if it is generated with the specific relation-sequences in mind.

Chow et al. (2009) have successfully implemented a relational feature representation, by generating a graph representation of the relations between the words in the document. The following feature representation uses the ideas of the bi-gram and the representation created by Chow et al. (2009).

Let $W = w_1, w_2, \ldots, w_n$ denote the words of a subsequence of pages. The function $\gamma(W)$ transforms the the words W into a sequence S upon a certain sort criteria. This sort function can be based upon any criteria such as the vertical and/or horizontal position of the words on a document. The sequence S is then transformed into a graph-based feature representation $G = (V, E, \phi, \theta)$ as described in Section 2.2.2. Instead of using the graph G, the adjacency matrix $F_{direction}$ is used. *direction* denotes which sequential direction is used from the graph G, which is either left or right. An example of this feature matrix is displayed in Figure 3.5.3.

$$W = \{Ipsum, Dolor, Lorem, Ipsum\}$$

$$S = \gamma(T) = \{Lorem, Ipsum, Dolor, Ipsum\}$$

$$F_{right} = \begin{array}{c} \\ Lorem \\ Ipsum \\ Dolor \end{array} \begin{array}{c} \begin{array}{ccc} Lorem & Ipsum & Dolor \end{array} \\ \begin{pmatrix} 0 & 2 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$F_{left} = \begin{array}{c} \\ Lorem \\ Ipsum \\ Dolor \end{array} \begin{array}{c} \begin{array}{ccc} Lorem & Ipsum & Dolor \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

Figure 3.2: An example of how two feature representations of a document with the words W can be created. The function γ uses a left-to-right sort criteria. F_{right} is a feature matrix based on a right-relational criteria, and F_{left} is based on a left-relational criteria.

The feature representation will most likely contain a lot of zeroes when represented by a matrix. This is something that the authors Chow et al. (2009) argue results in a poor performance. Chow et al. (2009) incorporated a principal component analysis to remove what they perceived to be unnecessary information. This will not be the case for the feature representation used by the FS algorithm, because the zeroes are quite vital. The zeroes, and also low values, signal unique sequential words, which was previously argued to be vital to discover multiple documents.

3.5.4 Validity Estimation

The validity estimation function assesses if a subseqeunce of pages contains valid sequences of words. Valid sequences of words exist if a subsequence of

pages does not contain repetitions of a classes unique sequences of words. This assessment is performed by measuring the similarity between the subsequence of pages feature matrix with the classes feature matrix. The validity estimation function $V(P)$ is calculated according to Equation 3.6.

$$V(P) = S(\alpha(P), DCM)) \tag{3.6}$$

Here, $\alpha(P)$ denotes the feature matrix for a subsequence of pages P. DCM denotes the feature matrix for the class, henceforth known as the document class model. The similarity measure and the document class model is described in the following sections.

3.5.5 Similarity Measure

The similarity measure used by the validity estimator uses concepts from the previously presented similarity measures from section 2.5. $S(x, y)$ denotes the similarity between the feature matrix $\mathbf{x}$ and $\mathbf{y}$. This thesis evaluates two similarity measures which are called the simple similarity measure and the smoothed similarity measure.

Simple Similarity Measure

The simple similarity measure is based on the same principles as described in the SMTP-measure, from Section 2.5. If both representations have the presence, i.e. a value greater than zero, of a shared feature the similarity increases. Similarly if both representations contain the absence of the same feature this further indicates an increased similarity. If one of the representations contain features which the other does not this indicates a decrease in similarity. The resulting similarity measure is found in Equation 3.7.

$$S(\mathbf{x}, \mathbf{y}) = \sum_{(i,j) \in F} Q(x_{i,j}, y_{i,j}) \tag{3.7}$$

$$Q(x_{i,j}, y_{i,j}) = \begin{cases} 1 & \textit{if } x_{i,j} = 0,\ y_{i,j} = 0 \\ 1 & \textit{if } x_{i,j} > 0,\ y_{i,j} > 0 \\ -1 & \textit{otherwise} \end{cases}$$

Note that x_i and y_i are the values the representations have for feature i. The set of features which is being compared is denoted by F.

Smoothed Similarity Measure

Sometimes an increased difference between the magnitude of features indicates a decreased similarity between two objects. The SMTP-measure handled this situation by calculating a score based upon the presence-abscence of features, and the difference in magnitude. This measure is presented in Equation 3.8.

$$S(\mathbf{x}, \mathbf{y}) = \sum_{(i,j)\in F} Q(x_{i,j}, y_{i,j}) \tag{3.8}$$

$$Q(x_{i,j}, y_{i,j}) = \begin{cases} 1 & if\ x_{i,j} = 0,\ y_{i,j} = 0 \\ \frac{1}{|x_{i,j} - y_{i,j}|+1} & if\ x_{i,j} > 0,\ y_{i,j} > 0 \\ -1 & otherwise \end{cases}$$

3.5.6 Document Class Model

The document class model is the feature matrix for a class. The classes training data is used to generate this feature matrix. Two methods for generating the document class model are evaluated in this thesis, the simple document model (SDM) and the unique shared words document model (USWDM).

Simple Document Model

A naive approach is to assume that one training example is enough, i.e. assuming that a class is homogeneous. This may be enough to generate a practically feasible document model when using invoices as documents. To generate the model simply take one document from the training data of the class and create the feature matrix based solely upon this.

Unique Shared Words Document Model

Another approach is to try and capture the fundamental structure of a class. This is achieved by using only the shared words of the training examples. These words exist in all documents, and the important sequences of words are most likely found here. These words are extracted and used to create the feature matrix.

Sometimes documents from a class will vary slightly between each other. Such variations will exist on some portion of the training examples. This may improve the accuracy, and as such is also evaluated. These two document models are referred to as *Unique Shared Words Document Model* (USWDM) and *2/3 Unique Shared Words Document Model* (2/3-USWDM), where 2/3 denotes the required portion of training examples to contain a unique word.

3.5.7 Separation Algorithm

Given a class-separation the fine algorithm builds the document class model, denoted DCM, as explained in subsection 3.5.6. The pages of a class-separation is denoted $D = \{d_1, d_2, \ldots, d_n\}$, and $D_{c:b} = \{d_c, \ldots, d_b\}$ denotes a subsequence of pages from d_c to d_b, where $1 \leq c < b \leq n$. The function to

find the separation points is defined as a recursive function R_j as shown in Equation 3.9.

$$R_j = \begin{cases} 0 & \textit{if } j = 0 \\ \max\limits_{i<j}(R_i + V(D_{(i+1):j})) & \textit{otherwise} \end{cases} \tag{3.9}$$

R_j calculates the best separation points up until index j in the sequence D. The best separations are the ones which achieve the highest similarity value from $S(x, y)$. To get the best separations for the class-separation simply calculate R_n and keep track of the selected separations.

3.5.8 Analysis & Limitations

The resulting FS algorithm is an exponential algorithm, with a time complexity of $\mathcal{O}(m2^n)$ where n is the number of pages in a class-separation and m is the number of class-separations in a batch. The performance of the algorithm can be increased by caching and reusing the results of the similarity for subsequences, but this leads to an increase in the memory complexity of $\mathcal{O}(2^p)$ where p is the amount of pages in a class separation.

The feature representation is built upon the premise that instances of documents can be identified with the help of sequences of the words. This may not always be true, e.g. in cases where news articles or heterogeneous classes are used.

The document class model may not contain some words or features if the USWDM or 2/3-USWDM is used. This may increase the accuracy because the dropped information may not be useful in separating instances. Some information which might be dropped include OCR errors and other word-noise which can exist, but there may be instances when vital information might be dropped and may in result negatively affect the separation.

3.6 Classification System

Both the SS algorithm, described in Section 3.2, and the RS algorithm, described in Section 3.4, use a classification system. The classification system which was used in the implementations of both these algorithms was provided by ReadSoft. This system is a proprietary software developed by experts at ReadSoft, and as such will not be disclosed here. The classification system is successfully used by their clients worldwide.

The implementations of both these algorithms are independent of this proprietary system. Any other classification platform could be used instead, as long as it produces the expected results by the algorithms. The SS algorithm requires the classification system to return the class label together with the probability, while the RS algorithm only requires the class label.

Chapter 4

Experiments

In this chapter the experiments of the statistical- and rough & fine separation algorithm are presented. The following sections describe the metrics, datasets and methods which where used to evaluate the performance of the two document separation algorithms.

4.1 Metrics

The following section presents the metrics which where used to evaluate the performance of the algorithms.

4.1.1 Batch Separation Accuracy

The purpose of the algorithms is to find every document in a batch, or in other words, find all the separation points. The obvious performance metric for evaluating the algorithms is batch separation accuracy (BSA). This metric can simply be calculated by the fraction between the number of correctly separated batches and the number of batches, as shown in Equation 4.1.

$$\text{BSA} = \frac{\text{\# Correctly Separated Batches}}{\text{\# Batches in dataset}} \tag{4.1}$$

This metric gives a good general performance overview of an algorithm, but it doesn't provide much else. Therefore the following three complementary metrics were also used to provide a more complete performance evaluation.

4.1.2 Batch over- and under-Separation Rate

The batch over-separation rate (BOSR) and the batch under-separation rate (BUSR) provide an insight to what went wrong on the incorrectly separated

batches. If an incorrectly separated batch contains more separation points then required it is regarded as an over-separation, and vice-versa, an incorrectly separated batch which contains fewer separations points then required is regarded as an under-separated batch. The rates are calculated by the fraction of over- or under-separations and the number of batches in the dataset, as shown in Equation 4.2 and 4.3.

$$\text{BOSR} = \frac{\#\text{ Over-separated batches}}{\#\text{ Batches}} \tag{4.2}$$

$$\text{BUSR} = \frac{\#\text{ Under-separated batches}}{\#\text{ Batches}} \tag{4.3}$$

4.1.3 Intra-Batch Separation Accuracy

The intra-batch separation accuracy (IBSA) adds an additional aspect to the BSA. Sometimes a batch may be correctly separated except for one separation. This results in a wrongly separated batch, but the algorithm actually separated many documents successfully. This is useful because sometimes a batch may contain bad pages or documents, rendering the batch inseparable and in results indicating that an algorithm is performing poorly, when it may in fact be okay. The IBSA is calculated by the fraction between number of correct separations and number of possible separations, as shown in Equation 4.4.

$$\text{IBSA} = \frac{\#\text{ correctly separated documents}}{\#\text{ documents}} \tag{4.4}$$

4.2 Data

The data which was used in the experiments was provided by ReadSoft. The data came in three suites, or datasets. These datasets are described in in the subsection below. The material contained 1143 OCR:ed [1] invoice documents spread over 377 different classes. The Figure 4.1 visualizes the spread of these documents over the classes. The data did not contain any prepared batches. This allows a ground truth for the comparison. The method used to generate batches and training data from a dataset is unique for the three different experiments. As such, the method used to generate the batches is presented in their corresponding section.

As the Figure 4.1 shows there were very few examples per class. Generally this is too few examples to properly train a machine learning algorithm, but in this case it is good enough. This is in part because the classes were internally very homogeneous.

[1] Optical Character Recognition is the processes of extracting the machine-encoded text from an image of the text.

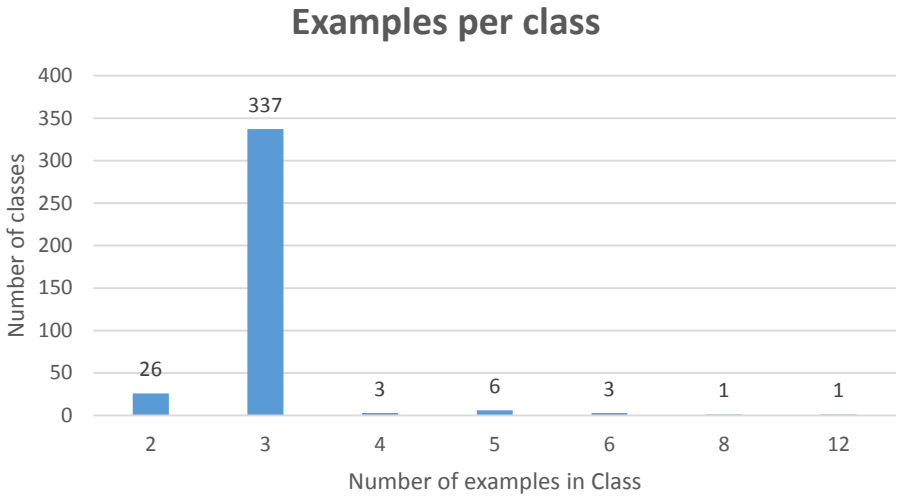


Figure 4.1: A histogram showing the number of examples every class has in the dataset All.

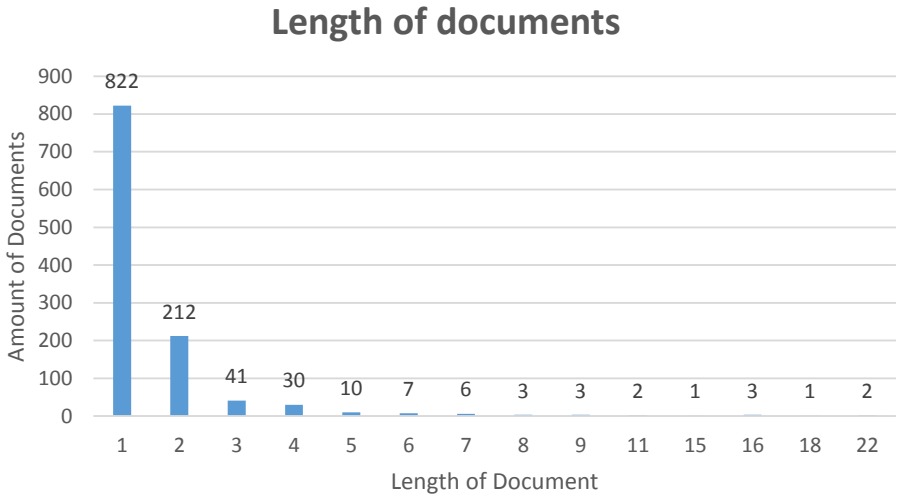


Figure 4.2: A histogram showing the number of pages the documents contain in the dataset All.

4.2.1 Datasets

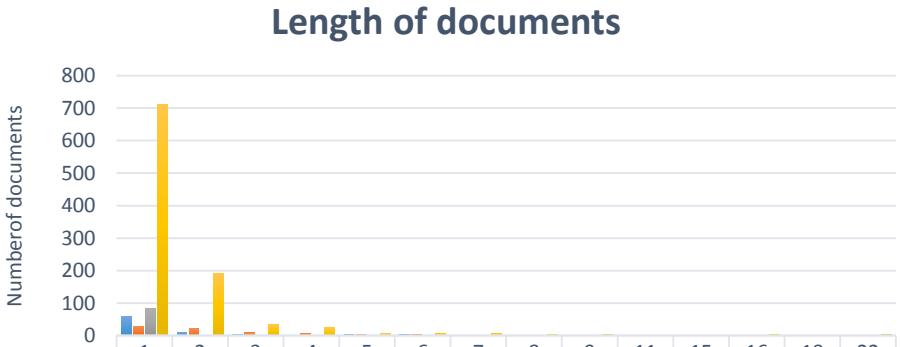
The data which was received came in three datasets. These different datasets where used individually and also combined during experimentation, in an attempt to evaluate different scenarios. These datasets provided realistic scenarios as they originated from realistic environments.

The first dataset contained invoices in the languages German and Dutch, and was therefore called Foreign Invoices (F). The second and third dataset contained invoices which were in Swedish. These two datasets were called Swedish Invoices Small (SI-Small) and Swedish Invoices Large (SI-Large), where small and large indicate the size of the datasets.

A fourth dataset was created from a subset of the previous three datasets, which was called ClassDocsSequence (CDS). This dataset was created with the only intention of evaluating the fine separation algorithm. The spread and size of these datasets is shown in Figure 4.3 and 4.4. All four datasets were also combined into a dataset called All.

	2	3	4	5	6	8	12
CDS	0	0	3	6	3	1	1
FI	12	2	1	4	1	0	1
SI-Small	13	6	2	2	2	1	0
SI-Large	1	329	0	0	0	0	0

Figure 4.3: A histogram showing the number of examples per class has for the different datasets.



	1	2	3	4	5	6	7	8	9	11	15	16	18	22
CDS	60	10	4	1	2	2	0	0	0	1	0	0	0	0
FI	29	21	8	6	3	2	1	0	1	1	0	0	0	0
SI-Small	82	0	0	0	0	0	0	0	0	0	0	0	0	0
SI-Large	711	191	33	24	7	5	5	3	2	1	1	3	1	2

Figure 4.4: A histogram showing the length, in number of pages, of the documents in the different datasets.

As can be seen from Figures 4.3 and 4.3 the SI-Large dataset contained 330 classes and the majority of all documents. The CDS dataset only contained classes with more then 3 examples per class. SI-Small only contained documents of one page length and FI contained the most varied documents.

4.2.2 Classification Accuracy

The RS and SS algorithms both utilized the provided proprietary classification platform. The performance of the separation algorithms depend on the classification algorithm being able to accurately classify data. The classification accuracy of the datasets can also indicate if the data is problematic and difficult. The classification accuracy for the datasets can be found in Table 4.1.

Dataset	**Classification Accuracy**
All	93,55%
Foreign Invoice	84,62%
Swedish Invoices	97,78%
Swedish Invoices Few	93,78%
ClassDocsSequenced	93,02%

Table 4.1: A table containing the classification accuracy for every dataset.

4.3 Baseline Algorithm

The baseline algorithm was created both to justify the increased complexity of the two separation algorithms and to be used as a minimum requirement on the performance.

The algorithm is a trivial and naive approach towards solving the separation problem. The algorithm regards every page as a new document, and it simply adds a separation point after every page. The scope of this thesis is to evaluate algorithms for separating invoices, and it is not uncommon that invoice documents are one page.

The RFS and SS algorithms implement concepts and techniques to learn where to separate the batches. Both algorithms should achieve a better performance then the baseline algorithm. If they do not achieve a better performance this can be an indicator that something is wrong with the ideas or implementations of the statistical- and R&F separation algorithms. This could also be an indication that the two algorithms are too complex and as a result they are under- or overfitting.

The material which was used to evaluate the algorithms contained a majority of one-paged documents, which is shown in the Figure 4.2. From this figure we can estimate the probability of receiving a one-paged document to be 71,96%. As such, the baseline algorithm should achieve an IBSA of 71,96%. The minimum IBSA requirement is therefore 71,96%, if either of the two separation algorithms should be deemed useful.

4.4 Experimentation Method

The RFS, SS and baseline algorithm were implemented using C# and the .NET framework. The datasets did not contain any predefined batches. As such, the used batches were generated in each experiment. Three different experiments were performed and in all of them the baseline algorithm was used and compared with.

4.4.1 Rough Separation Experiments

The purpose of the RS algorithm is to perform a class-based separation of a batch. As such the algorithm will only be able to find the correct separation points of a batches where no documents of the same class are located after one another. To correctly isolate and evaluate the RS algorithm, batches have to be generated which conform to this logic.

The generation of the batches and the training data from a given dataset was done with the help of a holdout method. The dataset was split in half, where one half of the documents was used as training material while the other half was used when generating batches. The batches were then generated by first randomizing the size of the batch and then randomly adding documents from the test-half of the documents. This was done until

all test-documents were added to a batch. The training data was used to train the classification platform. The size of the batches were between two and five documents. The evaluation of the rough separation algorithm is performed over the datasets FI, SI-Small and SI-Large.

4.4.2 Fine Separation Experiments

The purpose of the FS algorithm is to find separation points between sequenced documents of the same class. The fine separation algorithm only separates sequences of pages which have been pre-separated by class. To correctly isolate and evaluate the FS algorithm, batches have to be generated which adhere to this logic.

The generation of batches was performed in the same way as for the RS algorithm. A holdout method was used to split the dataset into a training-half and a test-half. The batches were then created class wise, where a batch contained all the test-examples of each class, in a sequence. To create batches which were usable, only classes which contain more then one test-examples was used. The only dataset which entirely conforms to this is the CDS-dataset, as seen in Figure 4.3, which is also the only dataset used to evaluate the fine separation algorithm.

4.4.3 Final Separation Experiments

The best RFS algorithm was compared against the SS algorithm from Gordo et al. (2013) as well as the baseline algorithm. The comparison was done over the four datasets FI, SI-Small, SI-Large and finally All. The generation of batches was done with the help of randomization functions and a hold-out strategy. Half of the documents in the used dataset was set to training documents, and the other half was set as test documents. To generate the batches a randomization function was used to generate batches of varying sizes with random documents from the test documents. The sizes of the batches varied between two and five documents per batch and every document was only used once. All three algorithms were evaluated on the same generated training and test data for every dataset.

Chapter 5

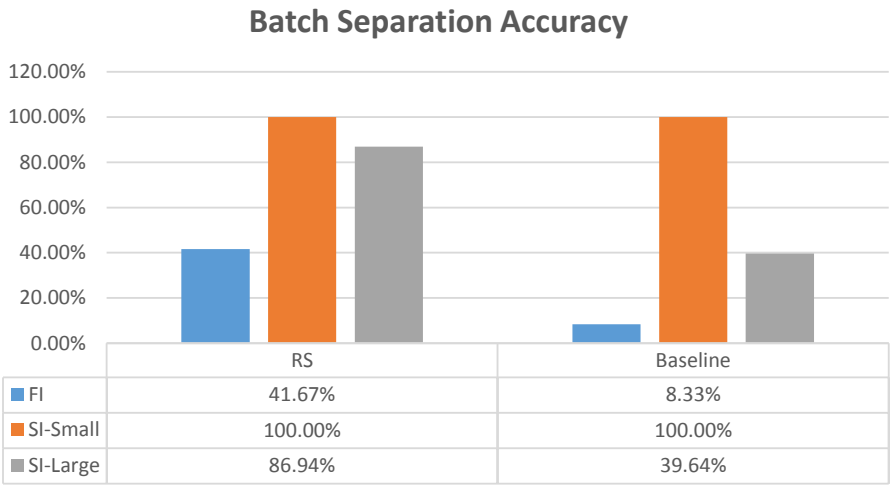
Results

This chapter first presents the results of the isolated rough separation and fine separation experiments. The chapter then concludes with the results from the final experiments with the rough & fine separation, statistical separation and baseline algorithms.

5.1 Rough Separation Experiment Results

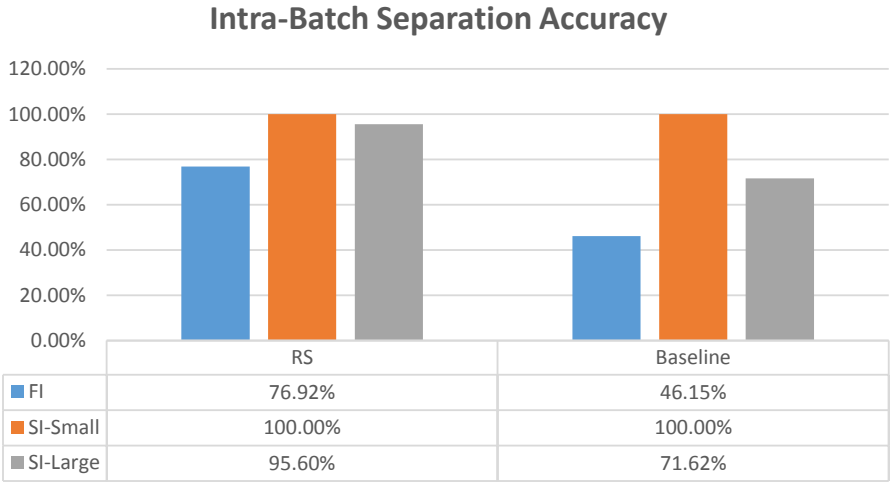
The rough separation experiments only investigated the performance of the RS algorithm. These experiments were performed over three datasets, and was compared against the baseline algorithm.

The RS algorithm outperformed the baseline algorithm on both BSA and IBSA, over all datasets. As can be seen in Figure 5.1 and 5.2, the RS algorithm achieved a BSA of 86.94%, with an IBSA of 95.6%, on the largest dataset available. When wrong separations occured, the RS algorithm only overseparated the batches, as seen in Figure 5.3 and 5.4.



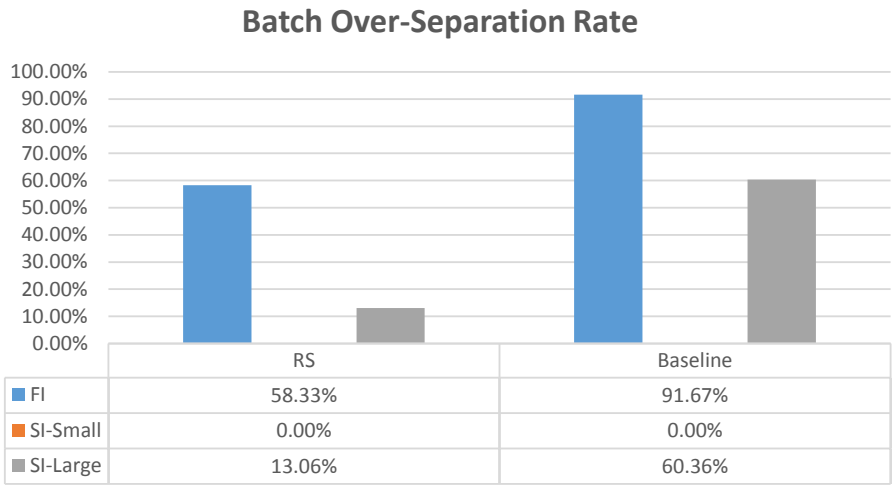
	RS	Baseline
FI	41.67%	8.33%
SI-Small	100.00%	100.00%
SI-Large	86.94%	39.64%

Figure 5.1: A comparison of the BSA of only the RS algorithm with the baseline over three datasets.



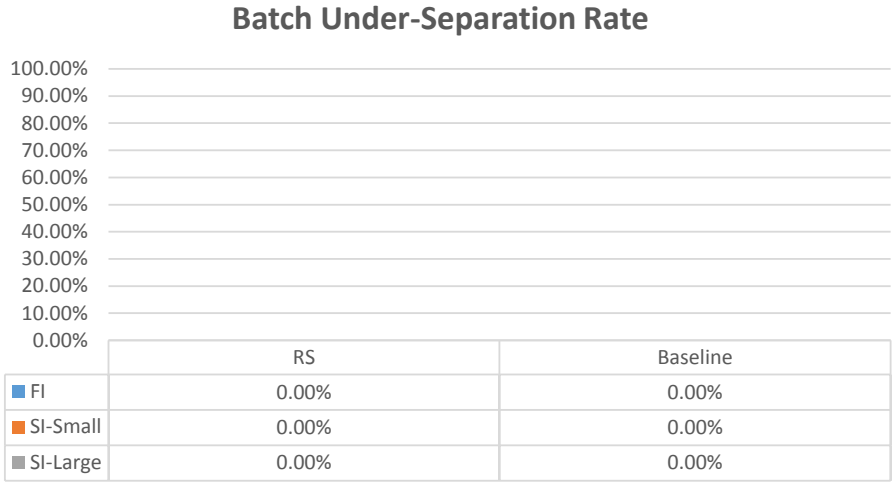
	RS	Baseline
FI	76.92%	46.15%
SI-Small	100.00%	100.00%
SI-Large	95.60%	71.62%

Figure 5.2: A comparison of the IBSA of only the RS algorithm with the baseline over three datasets.



	RS	Baseline
FI	58.33%	91.67%
SI-Small	0.00%	0.00%
SI-Large	13.06%	60.36%

Figure 5.3: A comparison of the BOSR of only the RS algorithm with the baseline over three datasets.



	RS	Baseline
FI	0.00%	0.00%
SI-Small	0.00%	0.00%
SI-Large	0.00%	0.00%

Figure 5.4: A comparison of the BUSR of only the RS algorithm with the baseline over three datasets.

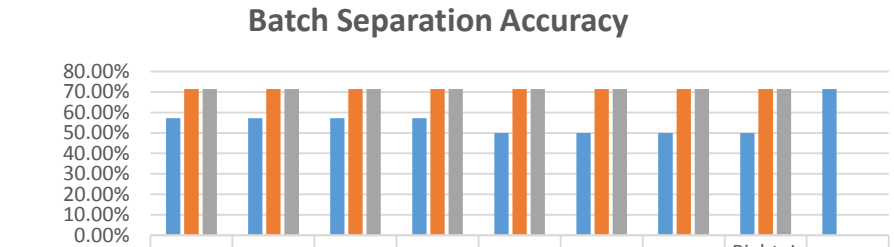
5.2 Fine Separation Experiment Results

The fine separation experiments were only performed on the FS algorithm in isolation, and in comparison with the baseline algorithm. These experiments were done to examine the impact of different feature representations, document models and similarity measures.

From Figures 5.5 and 5.6 it is discernible that the FS algorithm at its best variant only achieves a similar accuracy as the baseline algorithm. Both algorithms manage to achieve a BSA of 71,43% and an IBSA of 79,07%.

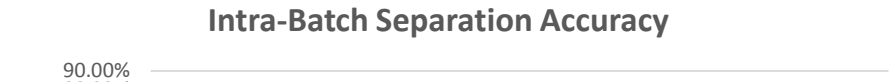
The Figures 5.7 and 5.8 show that the FS algorithm only performs underseparations of batches when using the SDM, achieving a BUSR of 42.86%. When using either the USWD or 2/3-USDWM the FS algorithm only tends to perform underseparations, with a BUSR of 21.43% compared to a BOSR of 7.14%.

The best variants of the FS algorithm used any of the feature representations, the USWDM and the smoothed similarity measure. The feature representation which was used in the final experiments was right-relations.



	Right-Simple	Left-Simple	Right+Left Simple	Right+Left +BoW Simple	Right-Smoothed	Left-Smoothed	Right+Left Smoothed	Right+Left +BoW Smoothed	Baseline
SDM	57.14%	57.14%	57.14%	57.14%	50.00%	50.00%	50.00%	50.00%	71.43%
USWDM	71.43%	71.43%	71.43%	71.43%	71.43%	71.43%	71.43%	71.43%	
2/3-USWDM	71.43%	71.43%	71.43%	71.43%	71.43%	71.43%	71.43%	71.43%	

Figure 5.5: A comparison of the BSA for the different feature representations, similarity functions and document models over the CDS dataset.

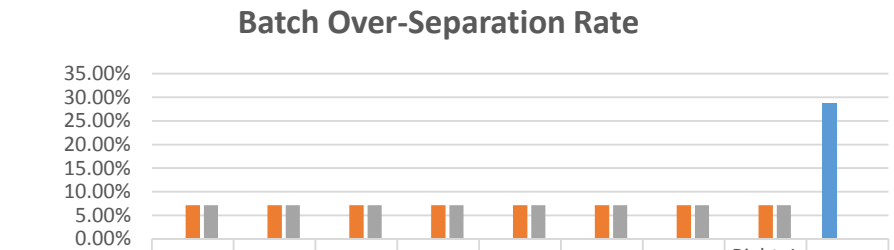


	Right-Simple	Left-Simple	Right+Left Simple	Right+Left +BoW Simple	Right-Smoothed	Left-Smoothed	Right+Left Smoothed	Right+Left +BoW Smoothed	Baseline
SDM	65.12%	65.12%	65.12%	65.12%	58.14%	58.14%	58.14%	58.14%	79%
USWDM	74.42%	74.42%	74.42%	74.42%	79.07%	79.07%	79.07%	79.07%	
2/3-USWDM	74.42%	74.42%	74.42%	74.42%	74.42%	74.42%	74.42%	74.42%	

Figure 5.6: A comparison of the IBSA for the different feature representations, similarity functions and document models over the CDS dataset.

	Right-Simple	Left-Simple	Right+Left Simple	Right+Left +BoW Simple	Right-Smoothed	Left-Smoothed	Right+Left Smoothed	Right+Left +BoW Smoothed	Baseline
SDM	42.86%	42.86%	42.86%	42.86%	50.00%	50.00%	50.00%	50.00%	0.00%
USWDM	21.43%	21.43%	21.43%	21.43%	21.43%	21.43%	21.43%	21.43%	
2/3-USWDM	21.43%	21.43%	21.43%	21.43%	21.43%	21.43%	21.43%	21.43%	0.00%

Figure 5.8: A comparison of the BUSR for the different feature representations, similarity functions and document models over the CDS dataset.



	Right-Simple	Left-Simple	Right+Left Simple	Right+Left +BoW Simple	Right-Smoothed	Left-Smoothed	Right+Left Smoothed	Right+Left +BoW Smoothed	Baseline
SDM	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	28.57%
USWDM	7.14%	7.14%	7.14%	7.14%	7.14%	7.14%	7.14%	7.14%	
2/3-USWDM	7.14%	7.14%	7.14%	7.14%	7.14%	7.14%	7.14%	7.14%	

Figure 5.7: A comparison of the BOSR for the different feature representations, similarity functions and document models over the CDS dataset.

5.3 Final Experiment Results

The final experiments were performed with the best variant of the RFS algorithm, the SS algorithm derived by Gordo et al. (2013) and the baseline algorithm. These experiments were done over the four datasets FI, SI-Small, SI-Large and All.

From Figures 5.9 and 5.10 it is clear that the RFS algorithm outperformed the other two algorithms in terms of accuracy. The RFS algorithm almost achieved twice as high BSA over the SI-Large, FI and All datasets, with an accuracy of 71.03% on All. The differences in IBSA were smaller, but the RFS algorithm beat the other algorithms on all datasets, with an accuracy of 88.43% on All. The only algorithm which managed to perform underseparations on a batch was the SS algorithm.

It is noteworthy that the SS algorithm only managed to perform equally well as the baseline algorithm. The only instance which the SS algorithm outperformed the baseline algorithm was on IBSA over the FI dataset, where it achieved an accuracy of 71.79% compared to 46.15%.

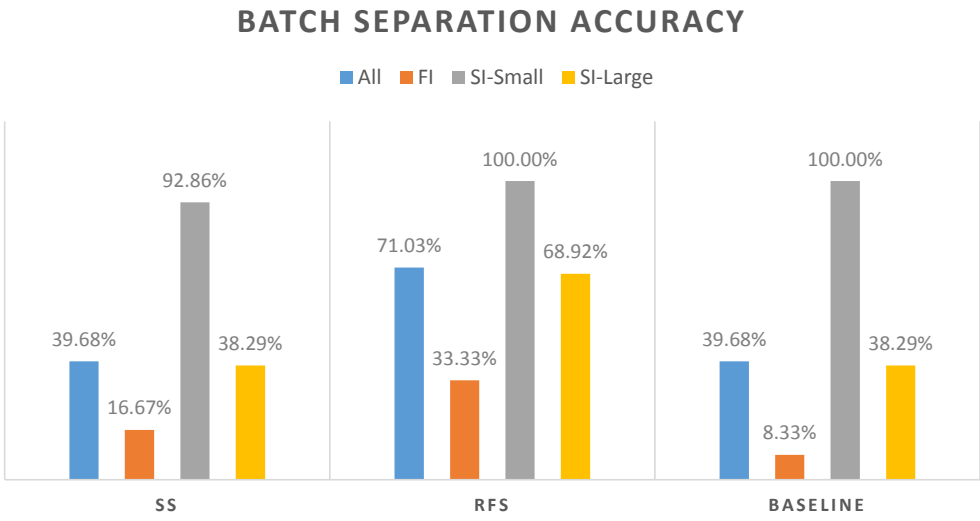


Figure 5.9: A comparison of the BSA for the algorithms over all the datasets.

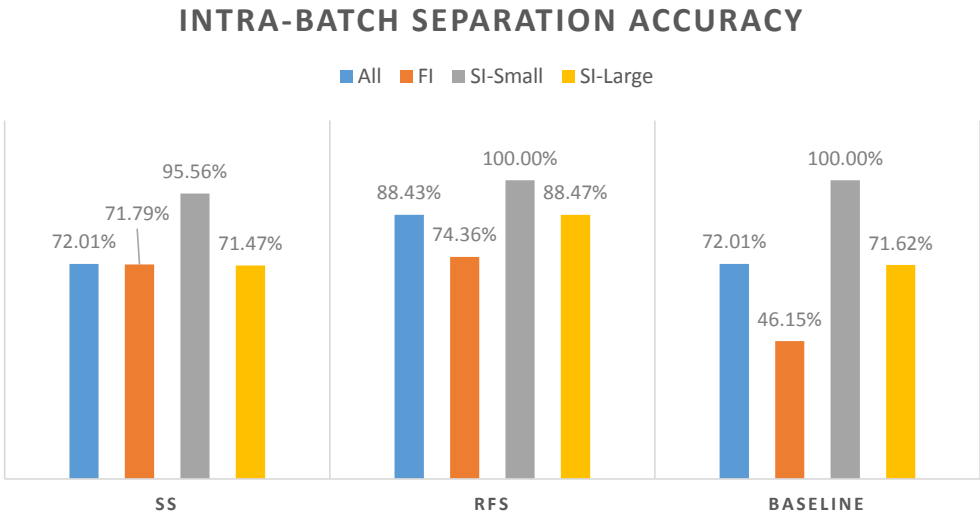


Figure 5.10: A comparison of the IBSA for the algorithms over all the datasets.

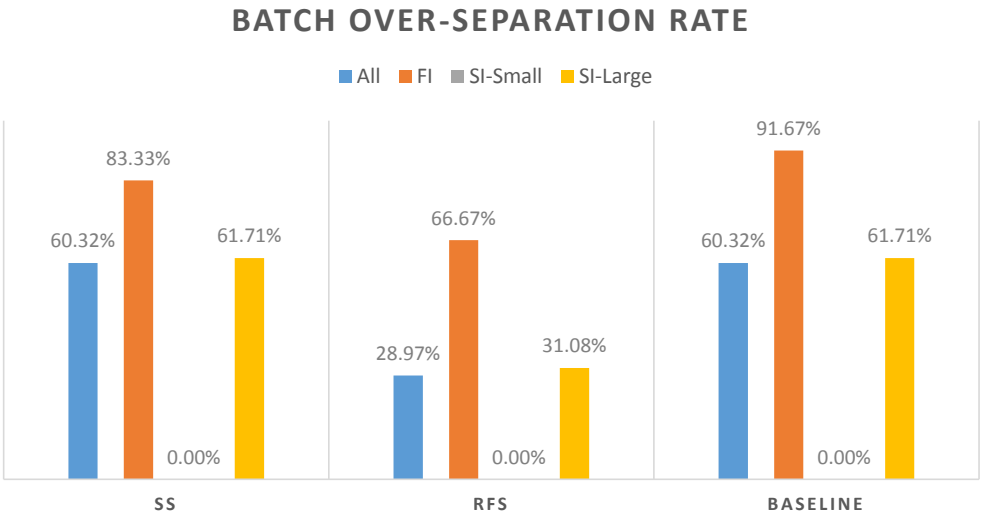


Figure 5.11: A comparison of the BOSR for the algorithms over all the datasets.

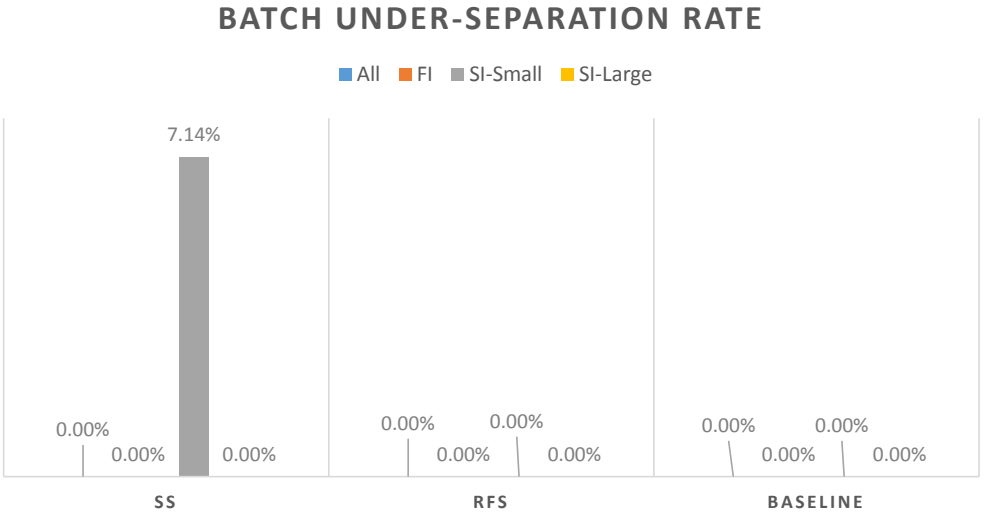


Figure 5.12: A comparison of the BUSR for the algorithms over all the datasets.

Chapter 6

Discussion and Conclusion

This chapter briefly discusses the results of the experiments, continuing with a conclusion of the thesis and finally presents potential improvements and future work.

6.1 Discussion

The RFS algorithm achieved an intra-batch separation accuracy of 88.43% over all test data available. Compared to the SS algorithm, which achieved an accuracy of 72.01% and also a worse performance on the other metrics. It is therefore clear that the RFS algorithm is the stronger choice. The final variant of the FS algorithm used the right-relation feature representation, the USWDM and the smoothed similarity measure.

6.1.1 Results

It is clear from the results that the RFS algorithm was the most successful separation algorithm. While it achieved substantially higher BSA than the alternatives, it still lagged behind its IBSA in absolute terms. A reason for this may be that even though the algorithm has a high IBSA, the incorrect separations get spread over the multiple batches causing the batches to become incorrectly separated.

The FS algorithm achieved atleast a 20% BUSR regardless of the feature representation, similarity measure or document model it used. The FS algorithm still achieves equal accuracy as the baseline algorithm, but the algorithms differ in the BOSR and BUSR. This implies that the algorithms achieve equally many document separations but in different instances. Given that baseline only separates on every page, the FS algorithm successfully separates the less common but more interesting case of multi-paged documents in a sequence. This is proof that the fine separation algorithm actually works.

From the fine separation experiments it is clear that the type of similarity measure and document class model had impact on the performance of the algorithm. The best performance was achieved with the USWDM. This document class model only selects the features which are common on all training examples, which is a feature selection technique. This result indicates that the FS algorithm could be further improved by finer feature selection techniques. The smoothed similarity measure resulted in a better performance when using the USWDM. The key element of the smoothed similarity measure was that it provided a varying similarity score based on the difference in factor of two features. This indicates that a more sensitive and finer similarity measure may further increase the performance. The improvement of the fine separation algorithm is further discussed in the improvements section below.

All algorithms performed notably worse on the FI-dataset. The classification accuracy of the FI dataset, seen in Table 4.1, was 84.62% which is also notably lower then the accuracy of the other datasets. This indicates that the dataset may contain problematic and difficult documents and classes. As such, the resulting performance drop for the document separation algorithms is not unexpected.

As the experiments showed, the SS algorithm was at best only equally good with the baseline algorithm. The SS performed worse then the baseline on the SI-Small dataset, where it had some under-separations. This indicates that something may be wrong wrong with the statistical separation algorithm. This issue was not further investigated , but the problem could be explained by the few examples, as described in the subsection below. Otherwise, this could indicate that either the implementation was faulty or that the algorithm was under- or overfitting.

6.1.2 Method

The data which was used in the experiments contained very few training examples for every class, and as such it is almost expected that a statistical algorithm should perform poorly. The SS algorithm requires a larger sample size if it is to estimate the two probability distributions correctly, which was not the case in this scenario. It should be noted that the SS algorithm uses the same classification system as the RFS algorithm to estimate the probability that a given sequence of pages belongs to a class. As such, this factor is not as affected by the fewer training samples. On the other hand, it can be argued that the used data is realistic. In many practical applications it is very hard to find a larger set of training examples, and as such performing relatively well with as little information as possible can be a requirement.

6.1.3 Practical Applications

It is difficult to conclude if the RFS algorithm is usable in practice. The reason is that it is going to be extremely difficult, costly or even impossible to overcome the inherent error which exists in a document separation algorithm. One factor which effects the inherent error is the classification accuracy, which is not 100%. The answer is really based upon what error rate is allowed in the application. Take a bank which handles invoices for its customer as an example. It is vital for the entire business that these invoices are handled in time, as people will loose money otherwise. In this instance, it is definitively not acceptable that a portion of invoices are incorrectly separated and possibly misplaced. On the other hand, in a small company using a digital mailroom to sort and distribute documents to departments internally this could be acceptable. It could be okay as it can be easy to re-place these documents and the consequences are not as expensive. This inherent error rate in document separation algorithms creates a possibility for further improvement, discussed in the improvements section below.

6.2 Conclusion

With the help of feature representations, similarity measures and classification algorithms a document separation algorithm was derived, called the rough & fine separation algorithm. This algorithm was compared against an existing solution, called the statistical separation algorithm. The rough & fine separation algorithm achieved a batch separation accuracy of 71,03%, with an intra-batch separation accuracy of 88,43%. The statistical separation algorithm managed to achieve 39,68%, with an intra-batch separation accuracy of 72,01%. From these metrics it is clear that the rough & fine separation algorithm is the more successful algorithm. The experiments were performed with invoices and datasets which did not contain large amounts of data, but they simulated realistic scenarios. The batch separation- and intra-batch separation accuracy of the R&F separation algorithm indicates that it could potentially be usable in practice, depending on the application and domain.

6.3 Improvements & Further Research

This section contains suggestions for potential improvements for the algorithms. These ideas were discovered during the work on this thesis but were outside the initial scope.

6.3.1 Degree of Uncertainty

The inherent error rate which exists in the document separation algorithms could be overcome by attaching a degree of uncertainty to a separation.

This is a method which is implemented in the classification system to allow human operators to intervene and help the machine when it becomes unsure, and as such it avoids some of the classification errors which can occur, such as a new class of documents.

This same method could be implemented in the document separation algorithms to overcome some separation errors, such as trying to separate a sequence of pages which belong to a class which has not been learnt yet. A successful implementation of a degree of uncertainty could render document separation algorithm more useful in practical applications.

6.3.2 Rough Separation Clustering

The clustering algorithm created by Collins-Thompson and Nickolov (2002) can overcome a common issue with certain documents. Some documents contain pages which are completely unrelated to a class, e.g. a page containing an image or an appendix of tables. The RFS algorithm would be unable to separate a batch with these documents, because it requires every page to be similar to the class it belongs too.

Their clustering algorithm incorporates a two-step clustering approach, where the first step clusters pages which clearly belong together. The second step looks at the remaining pages which do not seem to belong to any class, and tries to place them in their appropriate cluster. This allows the clustering separation algorithm to handle documents where some pages are completely unrelated to the class. This exact technique could potentially be incorporated in the rough separation algorithm, allowing it to handle these documents and as such achieve a higher batch separation accuracy.

6.3.3 Improved Fine Separation

As it was argued in the discussion, their is potential for improvements of the fine separation algorithm. The results indicate that feature selection techniques improved the accuracy of the algorithm, and as such it would be interesting to see if further development leads to better accuracy. Documents contains a lot of words which are so common that they are likely to not be part of the vital unique sequences of words, and may even be detrimental to the separation. These words could be removed by filtering inverted stopwords, which is discussed in Section 2.3. Other feature selection techniques such as stemming could also prove to be effective.

Furthermore, there were indications that an improved similarity measure can lead to a better accuracy. This could be investigated by developing a finer similarity measure or by using the SMTP, described in Section 2.5.

Bibliography

Bandyopadhyay, S. and Saha, S. (2013). *Unsupervised Classification*. Springer, Berlin.

Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, volume C, pages 161–168.

Chow, T. W. S., Zhang, H., and Rahman, M. K. M. (2009). A new document representation using term frequency and vectorized graph connectionists with application to document retrieval. *Expert Systems with Applications*, 36(10):12023–12035.

Collins-Thompson, K. and Nickolov, R. (2002). A clustering-based algorithm for automatic document separation. *Proc. SIGIR Workshop on Information Retrieval and OCR*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20:273–297.

Gordo, A., Rusinol, M., Karatzas, D., and Bagdanov, A. D. (2013). Document classification and page stream segmentation for digital mailroom applications. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pages 621–625.

Grefenstette, E. and Pulman, S. (2009). *MSc Computer Science Dissertation Analysing Document Similarity Measures*. PhD thesis, University Of Oxford.

Ikonomakis, M., Kotsiantis, S., and Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS Transactions on Computers*, 4(8):966–974.

Jiang, J. Y., Liou, R. J., and Lee, S. J. (2011). A fuzzy self-constructing feature clustering algorithm for text classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(3):335–349.

Kotsiantis, S. B., Zaharakis, I. D., and Pintelas, P. E. (2006). Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review*, 26:159–190.

Lin, Y.-S., Jiang, J.-Y., and Lee, S.-J. (2014). A Similarity Measure for Text Classification and Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1575–1590.

Liu, L. L. L., Kang, J. K. J., Yu, J. Y. J., and Wang, Z. W. Z. (2005). A comparative study on unsupervised feature selection methods for text clustering. *2005 International Conference on Natural Language Processing and Knowledge Engineering*, 00:597–601.

Manning, C. D., Raghavan, P., and Schütze, H. (2009). *An Introduction to Information Retrieval*. Number c. Cambridge University Press, Cambridge, England, online edition.

Rish, I. (2001). An empirical study of the naive Bayes classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 22230:41–46.

Salton, G. (1968). Search and Retrieval Experiments in Real-Time Information Retrieval. Technical report, Cornell University.

Slonim, N. and Tishby, N. (2001). The Power of Word Clustering for Text Classification. *European Colloquium on IR Research*, pages 1–12.

Smola, A. and Vishwanathan, S. (2008). *Introduction to machine learning*. Cambridge University Press, Cambridge, England, first edition.

Wajeed, M. A. and Adilakshmi, T. (2011). Different similarity measures for text classification using KNN. *2011 2nd International Conference on Computer and Communication Technology, ICCCT-2011*, pages 41–45.

På svenska

In English