Detection of Hospital Acquired Infections in sparse and noisy Swedish patient records.

A machine learning approach using Naïve Bayes, Support Vector Machines and C4.5.

Claudia Ehrentraut
Abstract

Hospital Acquired Infections (HAI) pose a significant risk on patients’ health while their surveillance is an additional work load for hospital medical staff and hospital management. Our overall aim is to build a system which reliably retrieves all patient records which potentially include HAI, to reduce the burden of manually checking patient records by the hospital staff. In other words, we emphasize recall when detecting HAI (aiming at 100%) with the highest precision possible. The present study is of experimental nature, focusing on the application of Naïve Bayes (NB), Support Vector Machines (SVM) and a C4.5 Decision Tree as well as different preprocessing methods to the problem, and the evaluation of the efficiency of this approach. The three machine learning algorithms are applied in three drafted classification tasks: binary classification, two-step classification and multi-class classification. Our machine learning approach is presented as an alternative to rule-based systems which are more common in this task. The results of the three classification task were overall similar. SVM yielded the highest recall 91% with the best overall performance, i.e., a $F_2$-score of 87.4%. In each classification task, the classifiers were applied on a small and noisy dataset, generating results which pinpoint the potentials of using learning algorithms for detecting HAI. Further research will have to focus on optimizing the performance of the classifiers and to test them on larger datasets.
# Contents

| List of Figures | iv |
| List of Tables | v |
| Acknowledgment | vi |

## 1 Introduction

1.1 Motivation ........................................ 1
1.2 Outline of the thesis .............................. 2

## 2 Background

2.1 Hospital Acquired Infections ...................... 3
  2.1.1 Definition .................................. 3
  2.1.2 Preventing HAI .............................. 4
  2.1.3 Detect-HAI project .......................... 5
2.2 Supervised Machine Learning ..................... 6
2.3 Definition of measurements ...................... 6
  2.3.1 Recall and precision vs sensitivity and specificity ... 6
  2.3.2 F-measure .................................. 7

## 3 Related Research

3.1 Rule-based approach ................................ 9
3.2 Machine learning approach ....................... 11

## 4 Data

4.1 Patient records .................................... 12
  4.1.1 Data collection ................................ 12
  4.1.2 Exclusion criteria ............................ 13
  4.1.3 Internal build-up ............................. 13
4.2 Manual evaluation sheet .......................... 15
4.3 IST-terminology database ......................... 15

## 5 Method

5.1 Experimental setup ............................... 16
  5.1.1 Binary classification .......................... 16
  5.1.2 Two-step classification ....................... 18
  5.1.3 Multi-class classification .................... 20
5.2 Learning algorithms .............................. 21
  5.2.1 Naive Bayes ................................ 21
  5.2.2 SVM ....................................... 21
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Internal build-up of patient records</td>
<td>14</td>
</tr>
<tr>
<td>5.1</td>
<td>Overview of the three classification tasks</td>
<td>16</td>
</tr>
<tr>
<td>5.2</td>
<td>Three perspectives of numerical distribution</td>
<td>17</td>
</tr>
<tr>
<td>5.3</td>
<td>Binary classification flow</td>
<td>17</td>
</tr>
<tr>
<td>5.4</td>
<td>Two-step classification flow</td>
<td>19</td>
</tr>
<tr>
<td>5.5</td>
<td>Multi-class classification flow</td>
<td>20</td>
</tr>
<tr>
<td>5.6</td>
<td>SVM illustration</td>
<td>22</td>
</tr>
<tr>
<td>5.7</td>
<td>Decision tree example</td>
<td>22</td>
</tr>
<tr>
<td>5.8</td>
<td>Generate ARFF-file</td>
<td>24</td>
</tr>
<tr>
<td>5.9</td>
<td>Directory layouts of the three classification tasks</td>
<td>25</td>
</tr>
<tr>
<td>5.10</td>
<td>Example ARFF-file</td>
<td>26</td>
</tr>
<tr>
<td>6.1</td>
<td>Recall for the binary classification</td>
<td>32</td>
</tr>
<tr>
<td>6.2</td>
<td>Percentage of HSRs containing HAI that were lost during the first classification step</td>
<td>35</td>
</tr>
<tr>
<td>6.3</td>
<td>Percentage of HSRs containing CAI that were lost during the first classification step</td>
<td>35</td>
</tr>
<tr>
<td>6.4</td>
<td>Percentage of HSRs containing NoINF that were misclassified as INF during the first classification step</td>
<td>36</td>
</tr>
<tr>
<td>6.5</td>
<td>Recall for the two-step classification</td>
<td>38</td>
</tr>
<tr>
<td>6.6</td>
<td>Recall for the multi-class classification</td>
<td>40</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Possible prediction outcomes in a binary classification</td>
<td>7</td>
</tr>
<tr>
<td>6.1</td>
<td>Precision, Recall and F₁-score (in %) for the binary classification</td>
<td>31</td>
</tr>
<tr>
<td>6.2</td>
<td>Precision, Recall and F₁-score (in %) for the first classification step</td>
<td>33</td>
</tr>
<tr>
<td>6.3</td>
<td>Precision, Recall and F₁-score (in %) for the second classification step</td>
<td>37</td>
</tr>
<tr>
<td>6.4</td>
<td>Precision, Recall and F₁-score (in %) for the multi-class classification</td>
<td>39</td>
</tr>
<tr>
<td>6.5</td>
<td>Recall and F₂-scores for the binary classification task</td>
<td>41</td>
</tr>
<tr>
<td>6.6</td>
<td>Recall and F₂-scores for the two-step classification task</td>
<td>41</td>
</tr>
<tr>
<td>6.7</td>
<td>Recall and F₂-scores for the multi-class classification task</td>
<td>41</td>
</tr>
</tbody>
</table>
Acknowledgments

I would like to thank Jörg Tiedemann for many constructive ideas, support and excellent guidance throughout the thesis. I am very grateful to Hercules Dalianis for giving me the opportunity to take part in the Detect-HAI project and taking the initiative and lead to turn parts of this thesis into an article.

I would further like to thank Maria Kvist for proofreading the thesis as well as her and Elda Sparrelid’s insightful classifications of Hospital Acquired Infections, Hideyuki Tanushi for extracting and providing the patient records in their initial form, Martin Hassel for suggestions regarding preprocessing and computing methods and all of you for helpful tips, comments and feedback whenever questions arose.
1 Introduction

1.1 Motivation

Patient security in hospitals is crucial. Various risk factors for patients can be found within clinical settings an example of which are Hospital Acquired Infections (HAIs), i.e., infections which are acquired in a health facility. HAIs pose a public health problem worldwide: A survey conducted under the patronage of WHO in 2002 found that, for 55 hospitals of 14 countries, in average 8.5% of all hospital patients suffered from HAI.

Many attempts have been made to confine HAIs, e.g., better hygiene or manual surveillance performed by infection control professionals, constituting an additional work load for hospital medical staff and hospital management. Nevertheless, the presence of HAIs remains unvaried in modern health facilities. Hospital Information Systems which are standard in most health facilities today and the increasing amount of digital data has pioneered the way for automatic surveillance systems. In the course of this development, research which focuses on the automatic detection of HAI has emerged throughout the past years. The exact approaches vary, yet it is noticeable that numerous attempts focus on implementing rule-based detection- or monitoring systems while machine learning based approaches seem quite scarce.

Our study is of experimental nature and focuses on applying machine learning techniques to the problem of detecting HAI. The approach complements a rule-based system which is currently developed in a collaborative project between Karolinska University Hospital and the Department of Computer and System Science (DSV) at Stockholm University. For our task, three well known learning algorithms were chosen, Naïve Bayes (NB), Support Vector Machine (SVM) and the C4.5 decision tree, and were applied to the data. The data used in this study comprised patient records which were provided by Karolinska University Hospital.

The focus of our study lies on the recall values obtained by the different classifiers. We aim at approaching 100% recall with the highest precision possible. In order to obtain our goal, we try different configurations, i.e., we set up three classification tasks, where different subdivisions are chosen for the data, as well as apply different data preprocessing methods and three classification algorithms. The study will focus on answering the following questions: (1) which recall is possible for our approach while keeping a reasonable $F_2$-score, (2) does any classifier outperform another, making it (them) more applicable for our task, (3) does any preprocessing method tend to yield better recall values than another and (4) how do the performance results differ between the three classification tasks.
Algorithms with high recall are especially suitable for the screening of infections; cf. (Klompas and Yokoe 2009, p. 1273). Thus, this study is an important step towards implementing a system which is expected to constantly screen patient records and determining whether they contain HAI. Automatic HAI screening is especially valuable for medical staff and hospital management, since it would significantly reduce the burden of manually checking patient records for HAI, which is a time-consuming task even for highly trained experts; cf. (Blacky et al. 2011, p. 366). Instead of analyzing all, the hospital staff would only have to check those patient records which were preselected by the system to contain HAI.

1.2 Outline of the thesis

The remaining parts of the thesis are organized as follows: Chapter 2 introduces the reader to Hospital Acquired Infections (Section 2.1) and briefly outlines supervised machine learning (Section 2.2), before providing the reader with information about elementary terms used throughout the paper (Section 2.3).

Chapter 3 surveys related work on HAI detection, concentrating on studies which have applied rule-based systems (Section 3.1) and those which have chosen machine learning techniques (Section 3.2).

Chapter 4 begins with introducing the primary data, i.e. the patient records, used in this study (Section 4.1). After that, the metadata used is presented: (1) an external Excel file which contains information about the patient records (Section 4.2) and (2) a terminology database which was constructed as part of the Detect-HAI project (Section 4.3).

Chapter 5 elaborates on the method used in this study. Section 5.1 introduces the experimental setup by presenting the three classification tasks. Section 5.2 briefly demonstrates the three machine learning algorithms before Section 5.3 presents the preprocessing techniques deployed. The chapter is finalized by the description of the evaluation methods (Section 5.4).

In Chapter 6 the results of the three classifiers given the different preprocessing methods are presented for the binary classification (Section 6.1), the two-step classification (Section 6.2), and the multi-class classification (Section 6.3). The chapter finishes by comparing the three classification tasks to one another (Section 6.4) before summarizing the results in Section 6.5.

Chapter 7 completes the paper and elaborates on future research ideas.
2 Background

This chapter starts by introducing the reader to Hospital Acquired Infections (Section 2.1). Subsequently supervised machine learning is outlined (Section 2.2), before the reader is provided with information about elementary terms used throughout the paper (Section 2.3.1).

2.1 Hospital Acquired Infections

This section starts with giving a definition of HAIs (Section 2.1.1) and elaborating on which methods have been applied in order to prevent and eliminate HAIs (Section 2.1.2), before presenting the Detect-HAI project at Stockholm University (Section 2.1.3).

2.1.1 Definition

In the 2002 paper by the World Health Organization, a HAI is defined as “[a]n infection occurring in a patient in a hospital or other health care facility in whom the infection was not present or incubating at the time of admission. This includes infections acquired in the hospital but appearing after discharge, and also occupational infections among staff of the facility” (Ducel et al., 2002, p. 1).

Alternative names mentioned in the literature are nosocomial infections (NI), cf. (Ducel et al., 2002, p. 1), healthcare-associated infections (HCAI), cf. (for Disease Control and Prevention, 2012) or (Blacky et al., 2011, p. 365), or healthcare-facility-acquired infections, cf. Wiegand et al. (2012). Throughout the paper the term Hospital Acquired Infection (HAI) will be used consistently.

To understand the importance of HAI monitoring and prevention, it is inevitable to look at estimates of how many HAIs occur in different nations’ hospitals. HAIs pose a public health problem worldwide, in developed as well as resource-poor countries; cf. (Ducel et al., 2002, p. 1). A survey conducted under the patronage of WHO in 55 hospitals of 14 countries\(^1\) found that, in average, 8.5% of all hospital patients suffered from HAI. Klevens et al. (2007) and Kelly and Monson (2012) found that 1.7 million HAIs occurred in US hospital of which 99,000 were deadly. The authors in Breathnach (2009) found that 5 to 10% of in-patients in British and Irish hospitals suffer from HAI while the researchers in Lamma et al. (2000) state that in Italy, even around 15% of all patients admitted to the hospital develop HAI.

\(^1\)which represented four WHO Regions Europe, Eastern Mediterranean, South-East Asia and Western Pacific
Besides the fact that HAIs are one of the leading causes of death among hospitalized patients, they impose an enormous economic burden on the hospitals, which is mainly due to the prolonged stay of the infected patients and further because of “increased use of drugs, the need for isolation, and the use of additional laboratory and other diagnostic studies” (Ducel et al., 2002, p. 1). See also (Kelly and Monson, 2012, p. 640).

In the literature, cf. (Ducel et al., 2002, p. 5-6), (Klompas and Yokoe, 2009, p. 1269), (Klevens et al., 2007, p. 163) or (Kelly and Monson, 2012, p. 640), the following types of HAI are commonly presented:

- surgical site infections,
- urinary tract infections,
- lower respiratory tract infections,
- bloodstream infections and
- pneumonia.

While the types are multifaceted, a recent study by Kelly and Monson (2012) see the common cause in resistant healthcare associated pathogens such as methicillin-resistant Staphylococcus aureus (MRSA) or Clostridium difficile. Most commonly, HAIs are found in surgical and intensive care units; cf. (Breathnach, 2009, p. 557).

For a more comprehensive presentation on HAIs, e.g., the way these infections are caused, definitions of the different types of HAI, the reader is referred to for example Ducel et al. (2002) or Breathnach (2009).

2.1.2 Preventing HAI

Ever since HAIs are known in the medical domain, efforts are made, e.g. by the Center for Disease Control and Prevention (CDC)\(^2\) in the US, to estimate the magnitude of HAIs as well as to monitor these infections to eventually minimize respectively eliminate the risk they pose within health facilities; cf. (Gaynes et al., 2001, p. 295 f.). Various techniques and approaches have been tried in order to eliminate HAIs, as mentioned by for instance (Proux et al., 2001, p. 35), (Breathnach, 2009, p. 557) or (Halpin et al., 2011, p. 271):

- better hand hygiene, aseptic handling of wounds or restrained antibiotic use,
- manual surveillance performed by infection control professionals or
- automated surveillance

The vast amount of electronic data, which is available due to Hospital Information Systems, permits the comprehensive identification and monitoring of HAIs, cf. (Adlassnig et al., 2009, p. 13). However, “until recently, hospital infection surveillance methods have relied almost exclusively on the manual

\(^2\)The CDC sponsors a collaborative surveillance system, called the National Nosocomial Infections Surveillance system, which collects national data about HAIs. The aim is to monitor HAIs and eventually advance prevention and detection methods. For more detailed information see Gaynes et al. (2001).
review of laboratory data and patient records” (Halpin et al., 2011, p. 270). This is attended by the fact that automated surveillance still faces doubts in its validity as (Klompas and Yokoe, 2009, p. 1273) and (Leal and Laupland, 2008, p. 221) point out.¹

Despite multifarious attempts to confine HAIs, their presence remains unvaried in modern health facilities. In recent years this has, according to (Breathnach, 2009, p. 557), led to increased political interest and public anxiety about these infections. As a result, healthcare authorities increasingly demand from healthcare facilities to install or regularly use HAI surveillance to reduce rates of infection. However, due to financial limitations, time constraints or the unavailability of experts at the local or regional level, manual surveillance is often inhibited; cf. (Blacky et al., 2011, p. 266). This, in return, favors the development of automated surveillance systems.

In this respect, research on automatic HAI surveillance has emerged throughout the last years. The project Detect-HAI which provides the framework for this thesis is one example and is presented in the following section. Other research projects which focus on automatic HAI-detection are presented in Chapter 3.

2.1.3 Detect-HAI project

The research presented in the paper at hand is part of the Detect-HAI (Detection of Hospital Acquired Infections through language technology) project which is currently conducted in collaboration between Karolinska University Hospital and the Department of Computer and System Science (DSV) at Stockholm University.⁴ The aim of the Detect-HAI project is to ultimately build a system which automatically can detect HAI in Swedish patient records.

Throughout the first year of the project (01/01/2012 - 12/30/2012) three approaches to build such a system were elaborated: Approach (1) focuses on a rule-based system while approach (2) aims at a system based on machine learning. Both of these approaches attempt to detect HAI in general. That means, they do not differentiate between the type of HAI, i.e., whether it is a urinary tract or bloodstream infection as outlined in Section 2.1.1. Both of these systems are in their research stage, whereat for the latter, primary results and findings are presented in this study. For approach (3), a prototype of a rule-based system for detecting health care-associated urinary tract infections - the UVI-tool - was developed. The system uses mainly structured infection-type specific data. Preliminary results show 80% recall, 67% specificity and 57% precision.

Attempts in detecting HAI differ as to whether the developers strive for high specificity or precision with a stable recall, or high recall, approaching 100% with the highest precision possible. Since the outcome differs substantially, it depends on the surveillance objectives, that is in most cases set by the hospital,

³A detailed discussion of manual surveillance methods as well as detailed comparison of automated and manual surveillance is beyond the scope of this paper. For examples of manual surveillance methods see Klompas and Yokoe (2009), Leal and Laupland (2008); for a more detailed comparison of manual and automated surveillance see ....

⁴For more information on the project see http://dsv.su.se/en/research/health/projects/detect-hai-1.113070

5
which approach to choose. Systems with a high specificity or precision can be used for tracking trends while systems which yield a high recall can be useful as a screening tool which is implemented as part of a Hospital Information System (cf. (Chapman et al., 2005, p. 448) and (Klompas and Yokoe, 2009, p. 1273)). In this regard, the built rule-based prototype aims for high precision, whereas the machine learning approach presented in this paper targets high recall.

2.2 Supervised Machine Learning

Machine learning describes the task of finding the concept which underlies some example data, in order to use it for making predictions on unseen data. The concept, or “the thing to be learned” as (Witten et al., 2011, p. 39) term it, is variously referred to by other expressions such as finding and describing structural patterns in data, inducing knowledge, or to build or learn a model from data. Different algorithms can be applied to the task and, depending on the algorithm, the concept is represented in different ways. The algorithms deployed in this study are described in Section 5.2, alongside with how they represent what has been learned from the data.

There are different types of machine learning, e.g., unsupervised learning, reinforcement learning or supervised machine learning. Text classification which takes center in this study is a type of supervised machine learning. Automatic text classification describes the task of assigning unseen instances, i.e., documents, to a predefined set of classes, using a machine learning algorithm. In order to do so, a learning algorithm is trained on a training set, where each instance is labeled with the class it belongs to. Based on this training data, a classification model is built by generalizing and inducing knowledge, i.e., identifying common ‘core’ characteristics of all instances in each class. This model is then used to assign unseen instances to one of the predefined classes.


2.3 Definition of measurements

This section begins with a definition of recall and precision as well as sensitivity and specificity before setting the terms in relation to one another (Section 2.3.1. Subsequently, the F-measure is briefly outlined in Section 2.3.2.

2.3.1 Recall and precision vs sensitivity and specificity

Recall and precision are frequently used terms in the context of language technology. Sensitivity and specificity are, however, comparatively unknown in this field. This section will briefly explain, how these terms can be related to another in order to ensure comprehensibility in the remainder of the thesis.

All four terms are statistical measures. However, their prominence differs depending on the field. Sensitivity as well as specificity are fundamental terms in medicine, used to evaluate clinical tests; cf. (Lalkhen and McCluskey, 2008).
Recall and precision are frequently used in computer science to measure the performance of a system, e.g., in machine learning or information retrieval. In sub-areas such as medical language processing, it remains the author’s decision or depends on the authors academic background, which terms to use. This results in the appearance of all four terms in the referenced literature.

Sensitivity and recall refer to the exact same measure. The confusion matrix depicted in Table 2.1 shows the four possible outcomes of a single prediction for a binary classification with classes A and B.

<table>
<thead>
<tr>
<th>Really is A</th>
<th>Really is B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classified as A</td>
<td>True positive</td>
</tr>
<tr>
<td>Classified as B</td>
<td>True negative</td>
</tr>
</tbody>
</table>

Table 2.1: Possible prediction outcomes in a binary classification

Based on these constitutive terms, recall and sensitivity are measured as follows:

\[
\text{Recall/Sensitivity} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}
\]

Adapted to our problem of classifying patient records into those which contain HAI and those which do not, recall and sensitivity, respectively, reflect how good the system is in retrieving all patient records which in fact contain HAI. To assure readability, the author of this paper decided to consistently use recall in this study. That means, in those cases where referenced authors have used the term sensitivity, recall is used when citing or paraphrasing instead. Specificity and precision define slightly different measures and are used accordingly. While precision is defined as:

\[
\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}
\]

specificity is measured as follows:

\[
\text{Specificity} = \frac{\text{True negatives}}{\text{True negatives} + \text{False positives}}
\]

Precision measures how many of the patient records that were classified as HAI are actually relevant, i.e., contain HAI. Specificity can be considered the inverse of recall/sensitivity, i.e., while recall/sensitivity measures the system’s performance in identifying positive results, specificity measures the system’s performance in identifying negative results. That means, specificity measures how good the system is in retrieving all patient records which in fact do not contain HAI. For a more comprehensive description of sensitivity and specificity in the medical context see Lalkhen and McCluskey (2008) or Akobeng (2007). For a more elaborate description of precision and recall see for instance Manning et al. (2008).

2.3.2 F-measure

It is sometimes necessary to express the performance of a system in terms of a single measure. The so-called F-measure is one possible measure which is
commonly deployed. It represents the weighted harmonic mean of precision and recall and is defined as follows:

\[
F_\beta = \frac{(\beta^2 + 1) \times \text{Recall} \times \text{Precision}}{(\beta^2 \times \text{Precision}) + \text{Recall}}
\]

According to Van Rijsbergen who has derived the F-measure, \(F_\beta\) “measures the effectiveness of retrieval with respect to a user who attaches \(\beta\) times as much importance to recall as precision” (Van Rijsbergen, 1979, p. 174). Thus, by making (1) \(\beta < 1\), weight is assigned to precision, (2) \(\beta > 1\), weight is assigned to recall or (3) \(\beta = 1\), precision and recall are weighted equally. The latter is referred to as the balanced F-measure or \(F_1\) which is represented by the reduced formula, as presented in (Witten et al., 2011, p. 175):

\[
F_1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Precision} + \text{Recall}}
\]

The results in Sections 6.1 to 6.3 are measured using \(F_1\).

As mentioned previously the aim of this study is to emphasize recall when detecting HAI (aiming at 100%) with the highest precision possible. This means in other words that we weigh the fact of finding all HSRs which contain HAI higher than the fact that all the HSRs retrieved may solely contain HAI, i.e., we accept the fact of obtaining a number of false positives. With the importance attached to recall, we are searching to point out the preprocessing method, classifier respectively classification task which show the best performance. In this regard, \(\beta = 2\) was chosen, resulting in the following formula:

\[
F_2 = \frac{5 \times \text{Recall} \times \text{Precision}}{(4 \times \text{Precision}) + \text{Recall}}
\]

\(F_2\) is used in Section 6.4 in order to compare the different classification tasks.
3 Related Research

This chapter surveys related work on automatic HAI-detection, concentrating on studies which have applied rule-based systems and machine learning techniques, respectively.

Research which focuses on automatically detecting HAIs has emerged throughout the last years. The carried out studies and what they have focused on, can be roughly classified according to the following categories:

- Unit of hospital studied, e.g., intensive care units (ICU) or emergency departments (ED)
- Type of HAI studied, e.g., surgical-site infections or ventilator-associated pneumonia (VAP)
- Using unstructured or structured data of patient records
- Applying a rule-based or machine-learning based approach

Section 3.1 will present a number of studies which implement a rule-based approach before Section 3.2 will illustrate research using machine-learning techniques.

3.1 Rule-based approach

A number of studies focus on an exclusive or primary use of the structured data of patient records. Klompas and Yokoe (2009) for example cite a number of studies were mainly microbiological data, antimicrobial criteria and diagnosis codes were used in automatic surveillance tools for identifying central line-associated bloodstream infections, surgical site infections, and VAP. The authors also pinpoint the fact that data, which is recorded as free text in patient records, they exemplify radiographic reports, eludes easy analysis by computers.

Blacky et al. (2011) have developed a software called MONI-ICU\textsuperscript{1} which is being used at the Vienna General Hospital and which is for example suited for day-to-day follow up of infections. Their system uses structured clinical, laboratory, nursing as well as administrative data from 12 intensive care units. Algorithm and rule-based knowledge is applied onto the data and abstract concepts are deduced stepwise in order to make a decision as to whether a certain HAI is present or not. The usage of fuzzy set theory and fuzzy logic\textsuperscript{2} allows the system to present the results in the form of a numerical value from

\textsuperscript{1}MOnitoring of Nosocomial Infections in intensive care units.
\textsuperscript{2}For more details on these concepts the reader is referred to Adlassnig (2001).
0 to 1, indicating whether HAI is contained or not. They compare the output of their system to that generated by trained surveillance staff, using 99 ICU patient admissions representing 1007 patient days. Their system obtained a recall value of 90.3%, i.e., in the case of presence of an HAI condition, 28 out of 31 cases have been identified. For specificity they obtain 100%, i.e., in the case of absence of HAI, all 68 cases were identified correctly.

Bouzbid et al. (2011) focus on comparing different strategies for detecting HAIs in intensive care units (ICUs). These strategies differ by whether data from a single or a combination of hospital databases is considered. Of all their strategies tested, the 'Drug prescription database or Microbiological database combined strategy' provided the highest recall (99.3%), however at the expense of a low specificity (56.8%). The best ratio between recall and specificity was obtained for the 'Microbiological database strategy' and the 'Electronic hospital discharge summaries database', The latter is, however, based on a manual analysis of the discharge summaries and no automatic detection tool yet. Still, according to the authors it demonstrated “the feasibility of detecting [HAI] using textual medical documents” (Bouzbid et al., 2011, p. 42), which marked a valuable step to the ALADIN project.

The researcher in the ALADIN project focus on automatically detecting HAIs in textual documents using Natural Language Processing; cf. (Bouzbid et al., 2011, p. 39). Their rule-based system utilize unstructured data, i.e., 205 patient discharge summaries from several hospital units. As first results, their system obtains a recall of 87.6% and specificity of 97.4%, cf. (Proux et al., 2011, p. 46).

Similar to Proux et al. (2011), the authors in Haas et al. (2005) use unstructured data in order to detect HAI. The researchers developed a surveillance system for pneumonia in neonatal intensive care units (NICU). The system processes free text chest x-ray reports with help of the MedLEE-tool which extracts clinical information related to pneumonia. Subsequently, rules are applied in order to detect hospital acquired pneumonia. The authors obtain a recall value of 71%, and specificity value of 99.8%.

More research in this areas has been conducted by for instance Woeltje et al. (2008), Trick et al. (2004) or Fiszman et al. (2000).

The above cited articles clearly illustrate the vast amount of research which focuses on automatic detection of HAI by analyzing structured or unstructured data in patient records. All of these approaches are rule-based, where the rules are handcrafted and not created by machine learning techniques. By contrast, to the knowledge of the author of this paper, studies which describe machine learning approaches in order to detect HAI are relatively rare.

3Collaboration of the Xerox Research Centre Europe in France and the Lyon University Hospital, http://www.aladin-project.eu/index-en.html.

4A large number of projects, focusing on automated rule-based HAI surveillance, are carried out in various constellations by researchers around Denys Proux, M-H. Metzger, S. Bouzbid and S. Gerbier in collaboration with Lyon University Hospital. They do, however, have different foci in there studies, some of which are cited in the course of this article.

5http://www.cat.columbia.edu/medlee.htm
3.2 Machine learning approach

The research group around Cohen et al. (2006) at the University Hospital of Geneva chose to apply machine learning techniques. The main focus of the researchers is to address the problem of class imbalance which can be observed in many real world classifications, especially in the medical domain. In their data, out of 683 patients, 75 (11%) were infected while 608 were not. The researcher apply various techniques in order to detect patients with HAI.

In Cohen et al. (2003), the authors tested (1) random and AHC oversampling, (2) K-means subsampling and random subsampling\(^6\) and (3) combined AHC oversampling and K-Means subsampling. They compared them by using five different classifiers: IB1, Naive Bayes, C4.5, AdaBoost and SVM. They achieved a recall ranging from 49% (IB1) to 87% (NB) for the five different classifiers when applying combined AHC oversampling and K-Means subsampling. Specificity ranges from 74% (NB) to 86% (IB1).

In Cohen et al. (2004), the researchers applied one-class SVMs to the problem. This adaption of SVM can be trained to distinguish two classes by ignoring one of the two classes and learning from one class only. Their best results yielded a recall of 92.6% at the cost of a very low specificity of 43.73%.

In Cohen et al. (2006), the researchers compare the resampling strategy which has shown the best results in Cohen et al. (2003): combined AHC oversampling and K-means subsampling to an asymmetrical soft-margin SVM. The asymmetrical soft-margin SVM obtained a recall of 92% and a specificity of 72%, thus clearly outperforming their resampling method which obtained the highest recall of 87% and 74% specificity for Naive Bayes.

Classification learning has been barely applied to the problem of detecting HAI. Yet, it is widely used in the medical domain, whereat diagnostic classification marks one of the main application areas. A detailed presentation of classification learning approaches in medicine and other areas is beyond the scope of the paper and is only outlined by means of two example application areas: One application range can be exemplified by the work of Illán et al. (2011), who built an automatic SVM-based diagnosis system which is meant to assist in the early detection of Alzheimer. Another example application area is illustrated by means of the research which has been carried out by the authors in Chapman et al. (2005). The researches conducted an observational study where they examined the performance of a Naive Bayes classifier, CoCo, at “categorizing patients into 1 of 7 syndromes based on triage chief complaints” (Chapman et al., 2005, p. 446), which they obtained from the Emergency Department of the University of Pittsburgh’s Medical Center. CoCo was trained on more than 10,000 chief complaints that were manually classified with any of the seven syndromic categories. The classifier’s accuracy ranged from 92% to 99% while recall ranged from 30% to 75%.

\(^6\)Two versions of undersampling.
4 Data

This chapter begins with introducing the primary data, i.e. the patient records, used in this study (Section 4.1). After that, the metadata used is presented: (1) an external file which contains information about the patient records (Section 4.2) and (2) a terminology database which was constructed as part of the Detect-HAI project (Section 4.3).

4.1 Patient records

By law, the health-care process must be documented in patient records, which in turn can be regarded as a support and mnemonic for health-care professionals. In terms of automatic surveillance, digital patient records play a crucial part in order to build surveillance systems. The patient records used in the Detect-HAI project as well as in this study were provided by Karolinska University Hospital and used as both, training and test data.

The section begins with describing how the records were obtained as well as which records of those received by Karolinska University Hospital are not included in the final dataset (Section 4.1.2). Subsequently, patient records are described in more detail in Section 4.1.3.

4.1.1 Data collection

The patient records used for this study have been retrieved from Point Prevalence Measurements (PPM), conducted in the spring of 2012. PPMs are carried out during one day, twice a year - spring and fall, at all health care units in Sweden. PPM represents a measure which estimates the occurrence of disease in a population by counting new and existing cases of the disease at one specific time. In total, PPM-records for 120 patients from 70 different clinics at Karolinska University Hospital have been received. All these records deliberately contain HAI according to the PPM results. The medical experts of the Detect-HAI project had access to five month of records for each of these patients. As a result, the experts, unlike the physicians who carried out the PPMs, obtained information on how the progress of the disease and with it the treatment proceeded during the time after the PPMs had been conducted. Thus, they could give a more concise assessment on whether an HAI really occurred in a patient record or not. According to their manual evaluation, see Section 4.2, in fact, only 129 patient records contain HAI, whereas 84 do not.

1This research has been approved by the Regional Ethical Review Board in Stockholm (Etikprövningsnämnden i Stockholm), permission number 2012/1838-31/3.

2https://onlinecourses.science.psu.edu/stat507/02/occurence
4.1.2 Exclusion criteria

The records of those patients, who were hospitalized for less than 48 hours, are not represented in the final dataset in order to minimize the risk of including infections which are not associated to the hospital. The time frame is based on international definitions of HAI, cf. (Kelly and Monson 2012, p. 640), and the incubation period of infections, and is estimated to be less than 48 hours for a multitude of disorders.

4.1.3 Internal build-up

Generally, if a patient is admitted to a clinic, daily patient records (DPRs) are kept. Those records contain an unstructured part for notes, where general treatment and observations are written by nurses while more specific examination and treatment notes are written by physicians. When the patient is discharged, a discharge summary\(^3\) is written by the physician, i.e., a summary of the treatment and also advice of how to care after discharge. Additionally, the patient record may contain structured parts, namely medication, microbiological data and body temperature, where the data is obtained from various hospital databases. Nowadays, patient records are produced, kept and visualized digitally in recordsystems such as TakeCare; cf. (Friberg and Johansson 2011, p. 5 f.). Such environments contain different modules where nurses and physicians enter information as described above which are then saved in the various hospital databases. For the purpose of the Detect-HAI project and this study, the information about the respective patients had to be retrieved from its storage. Based on evaluations of the two medical experts as to which information kept in the digital patient record is valuable for the Detect-HAI project, information from four modules of the TakeCare environment was retrieved: Journalanteckning (Engl.: record notes), Läkemedelsmodul (Engl.: drug module), Mikrobiologiska Svar (Engl.: microbiological result), and Kroppstemperatur (Engl.: body temperature). The data was extracted from the TakeCare environment and merged into a text file by Hideyuki Tanushi. One text file contains the retrieved information of one daily patient record of one patient. The general internal build-up of such a text file is depicted in Figure 4.1a

\(^3\)Alternatively referred to as discharge diagnoses by for instance Lauría and March (2011).
As cited above, various studies which utilized the unstructured part of the patient record, used discharge summaries for building their mainly rule-based systems. Our first attempt was to do likewise. However, based on an ocular analysis by our medical experts, it was found that, in some discharge summaries, no indications were given from which the medical experts could have inferred that an HAI occurred. Rather, information obtained from all (or numerous) daily patient records, which reflect the patient’s stay, gave information about whether an HAI occurred or not.

We thus decided to merge all daily patient records which belong to one hospitalization into one file, which we will call Hospitalization Records (HSR), see Figure 4.1b. For this project, the medical experts define one hospitalization as the stay of a patient at a health facility which is needed for one care process. In case the patient is discharged from one health facility and admitted to another one within 24 hours this is regarded as the same hospitalization. Moreover, also a noted event, which occurred 24 hours after discharge, is included in the hospitalization. Altogether the merging procedure yielded 213 HSRs.

The number of daily patient records belonging to one hospitalization varies significantly, reflecting the number of days the patient has been hospitalized, i.e., from 2 to 144 DPRs for HSRs containing HAI and 3 to 93 DPRs for HSRs not containing HAI. Due to this variation, even the number of tokens per HSR varied. The number of tokens in those HSRs which contain HAI ranged from 172 to 48,150, yielding a total of 1,267,711 tokens. For those 85 HSRs which do not contain HAI the number of tokens for each file varied from 257 to 21,016, yielding a total of 282,197 tokens.
4.2 Manual evaluation sheet

In the course of the project a comprehensive excel file, vri_vardtidlist_ppm, was built by Hideyuki Tanushi and the medical experts Maria Kvist and Elda Sparrelid. The file contains valuable information about each hospitalization. Most important for this work are the start and end dates of each hospitalization, indicating which DPRs that belong to it. In addition, the file comprises the manual evaluations of each hospitalization, i.e., whether it contains:

- an infection (INF), in particular:
  - HAI or
  - a community acquired infection (CAI), i.e., an infection which was not caused in the hospital, or

- no infection (NoINF)

This information was utilized when merging all Daily Patient Records into their respective Hospitalization Record and sorting all these files into the particular class directories as will be described in the following sections.

In addition to the patient records and the aforementioned excel arc which contains information about them, a terminology which comprises infection-specific terms (IST) was utilized throughout the experiments and is described in the subsequent section.

4.3 IST-terminology database

In the course of the Detect-HAI project at DSV, a terminology database which contains infection-specific terms (IST) was built in a semiautomatic approach. Infection-specific terms, such as kateter (Engl.: catheter), ultraljud (Engl.: ultrasound), operation (Engl.: surgery), or feber (Engl.: fever), are expected to be contained in patient records in case an infection occurs. In order to build the terminology database, the medical experts involved in this project supplied a seed set of about 30 infection-specific terms which were based on frequent observations in the above mentioned data and their knowledge about infections. The seed set was then extended by giving each term of it as input to an automatic synonym extractor. The synonym generator deployed was implemented at DSV and is based on random indexing; cf. [Hassel (2004)]. For each input term, a table holding related terms, which could include synonyms or misspellings, was generated as an output by the synonym generator. All proposed terms were then manually analyzed by one medical expert with respect to whether they could be regarded as applicable infection-specific terms or not. All relevant terms were added to the terminology database. The final IST-terminology database comprises a total of 1,045 terms.
5 Method

This chapter elaborates on the method used in this study. In this regard, Section 5.1 introduces the experimental setup by presenting the three classification tasks. Section 5.2 briefly demonstrates the three machine learning algorithms before Section 5.3 presents the preprocessing techniques deployed. The chapter is finalized by the description of the evaluation methods (Section 5.4).

5.1 Experimental setup

The experimental set up comprises three classification tasks, which are depicted in Figure 5.1.

![Overview of the three classification tasks](image)

Figure 5.1: Overview of the three classification tasks

The ultimate aim of all three classification tasks is to reliably filter out all HSRs which contain HAI, by means of approaching a recall of 100% with the highest precision possible. The approaches vary in the way that they assign the HSRs to different classes, resulting in diverging class information when training the classifiers.

This section will first introduce the reader to the binary classification task (Section 5.1.1). Subsequently, the two-step classification (Section 5.1.2) and multi-class classification tasks (Section 5.1.3) are presented.

5.1.1 Binary classification

In the binary classification task, the goal is to train a classifier which can distinguish between HSRs which contain HAI and and those which do not, in
order to assign unseen HSRs into the class HAI or NoHAI. Given the daily patient records, the information which is provided in the excel file (see Section 4.2) and which states which hospitalizations contain HAI and which do not, was utilized. As shown in Figure 5.2a, 129 HRSs contain HAI while 84 do not. The emphasis of this classification lies in the differentiation between those HSRs which contain Hospital Acquired Infections and those which do not. This implies that the class NoHAI not only comprises 61 HSRs which contain no infection at all (NoINF) but also 23 HSRs which contain community-acquired infections (CAI).

Based on this information, all daily patient records were merged into their respective HSRs, which were then sorted into their particular class directories, i.e., HAI or NoHAI. From this directory structure an ARFF-file was generated. The ARFF-format is the required input format to WEKA. Exactly how the class directories were constructed and converted to the ARFF-format will be described in Section 5.3.1.

Once the dataset of 213 HSRs is converted to its ARFF-format, it is divided into a training set which comprises 90% of the data and a test set of 10%. As depicted in Figure 5.3, for all three algorithms, a HAI-NoHAI model is trained and tested using 10-fold crossvalidation which is described in more detail in Section 5.4.1. As a result, the classifier outputs its predictions on the test data, i.e., which of the HSRs that contain HAI and no HAI, respectively.

Binary classification - the only plausible approach?

Throughout the project, test classifications were performed based on the binary class approach which was introduced in the previous paragraph. The results were all in all very encouraging and have been presented in Ehrentraut et al. (2012). Yet, from thorough analysis of this binary apportionment and the initial classification results as well as discussions with the medical experts, it became apparent that only discriminating between HAI and NoHAI might not be the only plausible approach.
Detecting HAI, that is, deciding whether a patient suffers from HAI retrospectively based on the information given in patient records is a difficult task, even when performed manually. This is due to the fact that signs of HAI can sometimes be indistinguishable from symptoms that indicate other diseases or may not be named at all, as stated by the medical experts Maria Kvist and Elda Sparrelid. Similarly (Breathnach, 2009, p. 559) mentions the difficulties when distinguishing between genuine and nosocomial pneumonia.

This fact is reflected in the terminological structure of the HSRs. During the manual analysis of the HSRs, the medical experts found that terms used in HSRs containing HAI or CAI are similar if not the same. Which in fact appears plausible since both represent an infection and could thus be allocated to the same class INF. However, recall from Section 5.1.1 and Figure 5.2a that, in the initial HAI-NoHAI apportionment, HSRs containing HAI and HSRs containing CAI were assigned to different classes, HAI and NoHAI, respectively. Our classification task is based on a bag-of-words approach, i.e. only the absence or presence of a word is considered and not in which order or context it appears; cf. (Manning et al, 2008, p. 269). Thus, term occurrences are utilized in order to discriminate between one class and the other. If we now assume a terminological overlap between HSRs which were assigned to different classes, we would expect the classifier to have difficulties in finding discriminative features. And indeed, when analyzing the results of the binary classification, the tendency of a great number of positives seems to verify this assumption.

These consideration resulted in two additional approaches: the two-step classification, which is described in the following section and the multi-class classification which is described in Section 5.1.3.

5.1.2 Two-step classification

The emphasis of this classification task is to try optimizing classification results, and the recall values for HAI in particular, by incorporating the considerations mentioned in the previous section. As for the binary classification, the goal is to detect all HSRs which contain HAI. The approaches differ with regard to the trained classifier: while the essence of the binary classification is to train one classifier to distinguish between HSRs which contain HAI and those which do not, the core of the two-step classification lies in training two independent classifiers which are applied jointly in order to retrieve HSRs which contain HAI.

The first classifier is trained to distinguish between HSRs which contain infections and those which do not. This is referred to as the first classification step. For this purpose, HSRs containing HAI and CAI were allocated to the same class in the first classification step, INF, while all other HSRs were assigned to class NoINF. As shown in Figure 5.2b, 152 HSRs belong to class INF; 129 of which contain HAI and 23 CAI. The remaining 61 HSRs do not contain an infection and belong to class NoINF. Just as in the binary classification task, all daily patient records were merged into their respective HSRs, which were then sorted into the particular class directories for the first classification step, i.e., INF or NoINF. Then, an ARFF-file was constructed from this directory structure and devided into a training set, comprising 90% of the data, and test set, containg the remaining 10%. As depicted in Figure 5.4, an INF-NoINF-
model is trained and tested using 10-fold crossvalidation which is described in Section 5.4.1. As a result, the first classifier outputs its predictions on the test data, i.e., which of the HSRs that contain INF and no INF.

The second classifier is trained to discriminate between HSRs containing HAI and those containing community-acquired infections (CAI). This is referred to as the second classification step. While the first (INF-NoINF) and second (HAI-CAI) classifier are trained separately, they are applied consecutively on the same test set. This implies that the HAI-CAI classifiers is dependent on the results of the INF-NoINF classifier. In other words, the second classifier can only by applied to the subset of infections which were predicted by the first classifier.

![Figure 5.4: Two-step classification flow](image)

Those HSRs which were predicted as infections were used to build a test set for the second classification step. The subset of HSRs which was used for training during the first classification step and contained an infection was used to build a training set for the second classification task. Just as for the first classification step, the training and test set were built by (1) sorting the INF-HSRs into HAI or CAI folders (by running ProcessPredictions.java) and (2) generating an ARFF-file from this directory structure. As depicted in Figure 5.4, an HAI-CAI-model is then trained and tested. As a result, the second classifier outputs its predictions on the test data, i.e., which of the HSRs that contain HAI and CAI.

Section 2.3.1 described, which measures are used for illustrating the performance of the classifiers. Yet, in order for the first classification step to be reflected in the results of the second classification step, the way how recall is calculated in the second classification step had to be adapted. The calculation had to capture the fact that the set of HSRs containing HAI in the test set of the second classification step represents a subset of HSRs containing HAI in the test set of the first classification step. This subset results from the fact that HSRs which contain HAI but were not predicted in the first classification step are not included in the second classification step. An example: a test set in the first classification step contains 15 HSRs which contain HAI, i.e.,
$HAI_{test1} = 15$. The first classifier, however, finds only 13 of those HSRs. Then, the test set which is build for the second classification step only includes 13 HSRs which contain HAI, i.e., $HAI_{test2} = 13$. If no adaptations were made, the second classification step would take $HAI_{test2}$ as a total of all HSRs containing HAI. Assuming the second classifier predicts 13 HSRs to contain HAI, the recall would be measured as follows:

$$\frac{HAI_{pred}}{HAI_{test2}} = \frac{13}{13} = 100\%$$

This is, however, not representative for the performance with regard to the ‘entire’ dataset, including the first and second classification step. Thus, recall is calculated as follows:

$$\frac{HAI_{pred}}{HAI_{test1}} = \frac{13}{15} = 86.7\%$$

The precision measure in the second classification step remained unchanged. Yet, the F-score had to be recalculated from the precision and the new recall value.

### 5.1.3 Multi-class classification

As the name multi-class implies, this classification task involves several classes. The aim is to assign unseen HSRs into one of the three following classes: HAI, CAI or NoINF. As for the previous two classification tasks, the training and test sets were built by making use of the information about each hospitalization given in the excel file. As depicted in Figure 5.2c, 213 HSRs were used of which 129 contain HAI, 23 contain CAI and 61 contain NoINF. Utilizing this information, all daily patient records were merged into their respective HSRs, which were then sorted into the particular class directories for this classification task, i.e., HAI, CAI or NoINF. The multi-class classification flow resembles the binary classification flow and is demonstrated in Figure 5.5.

Just as for the other two classification tasks, once the dataset of 213 HSRs is converted to its ARFF-format, it is divided into a training set which comprises 90% of the data and a test set of 10%. As depicted in Figure 5.5 for all three algorithms, an HAI-CAI-NoINF model is trained and tested using 10-fold crossvalidation. As a result, the classifier outputs its predictions on the test data, i.e., which of the HSRs that contain HAI, CAI and no NoINF, respectively.

After outlining the built-up of each classification task the following section will describe which learning algorithms were utilized to build the different classifiers.
5.2 Learning algorithms

There is a large number of different learning algorithms and classifier models that could be applied in our task. A detailed discussion of classification models and their assumptions and properties is beyond the scope of this paper. We decided to apply three well-known techniques that have been shown to be very effective for text classification, namely Naïve Bayes (NB), Support Vector Machines (SVM) and decision trees (using C4.5). One important aspect which is common for all supervised learning techniques is feature selection. Both, (Dalal and Zaveri, 2011, p. 39) and (Colas and Brazdil, 2006, p. 174), state that preprocessing, feature selection and parameter tuning have a large impact on performance, more than the actual choice of the classification model.

In the following, a brief overview of the classifiers that were selected for the experiments is given, for Naïve Bayes (Section 5.2.1), SVM (Section 5.2.2) and C4.5 (Section 5.2.3).

5.2.1 Naïve Bayes

Naïve Bayes represents knowledge in form of probabilistic summaries, while making the simplistic assumption that all feature values contained in the documents, are statistically independent from each other (within each class); cf. (Hall, 1999, p. 10 f.).

The posterior probability that a document belongs to class $C_i$, given the feature values $<v_1, v_2, ..., v_n>$ of the document, is calculated as follows:

$$p(C_i|v_1, v_2, ..., v_n) = \frac{p(C_i) \prod_{j=1}^{n} p(v_j|C_i)}{p(v_1, v_2, ..., v_n)}$$

where $p(C_i)$ is the prior probability of the class, $p(v_j|C_i)$ defines the conditional probabilities of features given the document class and $p(v_1, v_2, ..., v_n)$ is a constant which can be computed if one requires that the posterior probabilities of the class sum to one. For more details on Naïve Bayes see Hall (1999).

5.2.2 SVM

Support Vector Machines (SVMs) use the concept of representing the documents which are to be classified as points in a high-dimensional space and finding the hyperplane that separates them. For illustration purposes, Figure 5.6 depicts the notion of SVM in a two-dimensional space where the hyperplane is simply a straight line. In a given classification tasks, a training set contains a number of HSRs which are labeled with the class they belong to, taken the binary classification task, this would be HAI or NoHAI. In a two-dimensional space, these training-HSRs are represented as dots1: blue dots representing HSRs which belong to class HAI and red dots, representing HSRs which belong to class NoHAI, as depicted in Figure 5.6. Now, a Support Vector Machine tries to find the line which best divides these two classes. This concept, in fact, is not unique to SVM. However, the difference between SVM and other classifiers deploying this concept is how the line, or

1The dots in turn represent document vectors
hyperplane, is selected. SVM tries to find the line with the maximum margin, where margin refers to the distance between the line and the nearest data points, cf. (Noble 2006, p. 1566), that is, in the present case, training HSRs on either side of the line, which are represented as yellow dots in Figure 5.6. These 'dots', or example vectors are termed support vectors, since they provide the decision function of an SVM classifier. Unseen HSRs are then classified based on which side of the line they appear.

Using SVM was among others motivated by the statement that “SVM is found to be very effective for 2-class classification problems” (Dalal and Zaveri, 2011, p. 39). For SVM, the SMO implementation which is part of the WEKA environment, has been used. For more details on SVM see Noble (2006).

5.2.3 C4.5

C4.5 is a decision tree classifier. Classifiers of this kind predict the class of an unseen instance by representing knowledge learned throughout training as a decision tree. Decision trees are built by selecting discriminating features based on information gain. For further details on how a decision tree is built during the learning phase, the reader is referred to Tan et al. (2006). Once the decision tree has been constructed, it can be applied to the test set to predict the class labels of previously unseen instances.

Figure 5.7 images a simple decision tree for the binary classification task. Nodes in the tree correspond to features, branches to their associate values observed in the instances and leaves to classes. Recall that in this classification task, all feature values are numeric, in which 1 represents the presence of the feature in a given instance and 0 represents the absence.

To classify an unknown instance based on a learned decision tree, one starts at the root node, operation in the example, which is considered to be the most discriminative feature; cf. see (Witten et al. 2011, p. 307). If the term operation is present in the instance, the branch corresponding to the observed value, i.e., 1, is followed. Upon

2Note that this tree is for illustration only. It does neither depict a hundred percent illustration of the tree built by C4.5 during the classification task nor does it match the complexity of indicators which lead to conclude whether HAI is present in given patient record or not.
reaching the next node, which corresponds to the feature swollen, it is tested whether this term exists in the instance or not, i.e., whether the value is 1 or 0. Depending on the outcome of this test, the tree is processed downwards alongside its branches. Given that the terms swollen, fever and antibiotics exist in the given instance, i.e., are represented by the value 1, the leaf HAI is reached. Here, the process terminates and the class of the leaf is assigned to the instance, cf. [Hall (1999)] and (Witten et al., 2011, p. 64). The crucial aspect in obtaining a good classification model lies in the complexity of the tree, that is, the tree must fit the training data well without overfitting it in order to be general enough to accurately classify the test data. A detailed discussion by means of which methods this can be achieved is given in for example [Tan et al. (2006)].

In the present study, the J48 implementation, which is part of the WEKA environment, was used. Decision tree algorithms have proved their robustness and execution speed and are therefore popular in practice. For more details on the C4.5 decision tree see [Hall (1999)], [Witten et al. 2011] or [Tan et al. (2006)].

5.2.4 The WEKA environment

All three classifiers are part of WEKA, the machine learning environment applied in this study. WEKA is a collection of state-of-the-art machine learning algorithms, data preprocessing and visualization tools as well as common evaluation measures, which was developed at the University of Waikato in New Zealand. Methods included in the environment span over all major data mining problems: regression, classification, clustering, association rule mining and attribute selection. The ARFF-format is the required input format in WEKA. ARFF stands for Attribute-Relation File Format and describes, according to the WEKA developers, “a list of instances sharing a set of attributes” (Bouckaert et al., 2012, p. 161). Exactly how the data at hand is converted into this format is closely described in Section 5.3.1. Weka can be used via three different graphical user interfaces: the Explorer, Experimentor and Knowledge flow. The experimentor interface was explicitly developed for the purpose of comparing different learning techniques and is therefore used within certain tasks of the project. The available commmand-line interface was, however, used for the majority of the tasks. WEKA is available from [www.cs.waikato.ac.nz/ml/weka] and extensive documentation is provided through their wikispaces. For more information on WEKA in the context of data mining, the reader is referred to [Witten et al. (2011)], which is written by the developers of WEKA.

For each algorithm the default parameters were used.

5.3 Preprocessing

The first part of this section describes how DPRs are merged to HSRs and then converted to the required ARFF-format (Section 5.3.1). Section 5.3.2 illustrates, which preprocessing and feature reduction methods are applied as well as how the data is processed in this respect.
5.3.1 Conversion to ARFF-format

The patient records, as they were received by Hideyuki Tamushi, were not directly usable for classification. All records were contained in the folder KnowtatorDataAll, which comprises subfolders for each patient. Each patient folder contains all daily patient records (DPRs) available for that patient, i.e., one text file per DPR. Thus, preprocessing steps were required to (1) merge these DPRs into HSRs and (2) convert these HSRs from their text format into the ARFF-format which is required by WEKA as described in Section 5.2.4.

The general flow of creating an ARFF-file is exemplified for the binary classification and is shown in Figure 5.8:

As motivated in Section 4.1, all DPRs which belong to one hospitalization were merged into one file, which is called HSR. This was done by running the program CreateHospitalizationSimple.java which basically traverses through all DPRs contained in the folder KnowtatorDataAll, merging them into their respective hospitalization file and then sorting the obtained HSRs into the HAI or NoHAI directory. Information about whether the particular HSR should be sorted into the HAI or NoHAI directory was retrieved from the excel file which was introduced in Section 4.2. Sorting the HSRs into this particular directory layout was crucial for the step of actually generating ARFF-files as is explained subsequently.

In WEKA, the TextDirectoryLoader class is available which generates ARFF-files from text documents which are provided in a specific directory layout. The layout in turn depended on the particular classification task. In the binary classification task, the HSRs were sorted into two folders, HAI or NoHAI, representing the two classes, as described above. In the two-step classification task, HSRs were sorted according to whether they contain (1) INF or NoINF and (2), of all the HSRs which were predicted to contain an infection, HAI or CAI. In the multi-class classification task, the HSRs are sorted into created HAI-, CAI- and NoINF-directories. The different directory layouts, specific to each classification task, are illustrated in Figure 5.9.
The respective main directories represent the relations, e.g., HAI-NoHAI, while the subdirectories, e.g., HAI and NoHAI, represent the classes which are used in the different classification tasks. Once the HSRs were sorted into the task-specific layout, the `TextDirectoryLoader` was used in order to create the ARFF-files as shown in Figure 5.8.

As visible in Figure 5.8, the ARFF-file holds (1) the name of the relation, @relation HAI-NoHAI, (2) a block with two features:\(^4\) text, which is of type String and class, which has two possible values enclosed in braces - HAI and NoHAI (3) and the instances, starting after @data. Instances are written one per line, whereupon the values of the feature class are separated by comma at the end of the line. In the present classification task, one instance equates to one document, i.e., one HSR. Thus, since there are 213 HSRs in the data, the ARFF-file contains 213 instances.

As has been described so far, illustrates the flow of creating an ARFF-file without any preprocessing method being deployed. Creating ARFF-files for the remaining preprocessing methods which are described in Section 5.3.2 was achieved by means of the following procedure: In order to generate an ARFF-file for the lemmatized data, the newly generated HAI-NoHAI folder, including its subdirectories HAI and NoHAI, was duplicated and lemmatized using the CST lemmatizer. The ARFF-file was then simply created by running the `TextDirectoryLoader` on the lemmatized directory. To generate an ARFF-file which included infection-specific terms only, an alternative method in the `CreateHospitalizationSimple.java` program was activated, removing all but the IST terms from each daily patient record, creating the HSRs and

---

\(^4\)In Figure 5.8 the term attribute is used. Both terms do, however, refer to the same entity. Attribute is simply the term which is, by convention, used in the WEKA context, Witten et al. (2011), while the term feature is utilized in many other books about machine learning, see e.g. Alpaydin (2010).
sorting those into separate versions of the HAI and NoHAI directories. Creating the ARFF-file from these directories was done in the same way as for the previously described preprocessing methods.

In the ARFF-files described and depicted so far, the text is represented as a String, ergo the feature text String in Figure 5.8. Most classifiers in WEKA, however, cannot process String features. Therefore, all instances had to be converted into numerical features vectors. This was done by using the StringToWordVector filter, see Bouckaert et al. (2012). This yielded an ARFF-file of the structure pictured in Figure 5.10. As visible, the ARFF-file now contains all features present in the feature space. The features can take on values of two kinds: 1, which represents the presence of the feature or 0, which represents the absence. Each instance is then characterized by the values of the set of features. Since each instance only contains a subset of the features which exist in the entire feature space, most values would be zero. To avoid this, the instances are represented as sparse feature vectors: being enclosed in braces, each instance contains the index number of each non-zero feature (index starts at 0) and its value.

The ARFF-files for the stop word removal and TF-IDF 50 techniques were created during this filter step, by giving the ARFF-file which was created for the no preprocessing technique as an input, and choosing additional parameters for stop word removal and tf-idf 50, respectively, when running StringToWordVector from the commandline.

In summary, five ARFF-files were created for the binary classification task, i.e., one ARFF-file for each feature selection technique:

1. No preprocessing: HAI-NoHAI.arff
2. Lemmatization: HAI-NoHAI_lemma.arff
3. Infection-specific terms: HAI-NoHAI_ist.arff
4. Stop word removal: HAI-NoHAI_stop.arff
5. TF-IDF 50: HAI-NoHAI_tfidf50.arff

The ARFF-files for the two-step and multi-class classification were created likewise.

5.3.2 Preprocessing the input data

There are several methods which are designed for preprocessing the input data in order to make it more amenable for learning methods and, as an integral part, improve prediction results. They range from simple preprocessing and feature reduction techniques to more elaborate methods such as feature selection, feature discretization, sampling or wrapper techniques; cf. (Dalal and
 According to for instance Witten et al. (2011), Colas and Brazdil (2006), or Yang and Pedersen (1997), in many cases, the feature space of the input data is high-dimensional, that means, there are far too many features for a learning scheme to handle. This fact marks a major characteristic and difficulty in text classification, making it a non-trivial task for automatic classifiers. In order to enhance the chance of success in classification it is thus desirable to reduce the dimensionality of the data to be processed by the classifier, “while maintaining the original information of the feature sets.” (Shi et al., 2011, p. 131). This is to reduce execution time and improve predictive accuracy. By doing so, irrelevant features which can introduce noise into the data and thus obscure possible relevant feature are filtered out (Doraisamy et al., 2008, p. 333).

Per default, the feature space comprised 1,000 features, i.e., given no further parameter specifications, WEKA chooses the 1,000 most frequent terms based on their term frequency (TF). There are a number of different weighting strategies which are adopted to measure the contribution of the feature (Shi et al., 2011, p. 131). TF refers to the simplest one where the weight of a term is equal to the number of times the term occurs in a document (Manning et al., 2008, p. 117).

In this study, four different preprocessing techniques are applied in order to optimize respectively reduce the feature space: lemmatization, stop word removal, TF-IDF, reduction to infection-specific terms. All techniques operate independent of the learning algorithms by reducing features from the data before learning begins; cf. (Hall, 1999, p. 28). For more examples of different preprocessing techniques see for example Yang and Pedersen (1997), Dalal and Zaveri (2011) and Doraisamy et al. (2008).

Lemmatization

To remove stop words from or lemmatize the input data are frequently used methods when preprocessing data for machine learning; cf. (Dalal and Zaveri, 2011, p. 38). In our study we used the CST lemmatizer\(^5\) in order to perform lemmatization. Lemmatization describes the process of reducing a word to a common base form, normally its dictionary form (lemma); cf. (Jongejan and Dalianis, 2009, p. 145). This is achieved by removing inflectional forms and sometimes derivationally related forms of the word, by means of vocabulary usage and morphological analysis. For instance: am, are, is ⇒ be, or hospitals, hospital’s ⇒ hospital; cf. (Manning et al., 2008, p. 32). For Swedish that is highly inflectional, lemmatization is more important than for English.

Stop word removal

Moreover, we used stop word removal as a preprocessing method. As part of the WEKA environment, we were able to deploy a filter which removed all stop words from the input text at the same time we ran the StringToWordVector. Stop words are terms which are regarded as not conveying any significant semantics to the texts or phrases they appear in and are consequently discarded; cf. (Dragut et al., 2009, p. 1). The filter was configured to use the Swedish stop

\(^5\)http://cst.dk/online/lemmatiser/uk/
list which is available via Snowball⁶ and which comprises 113 words, such as och (Engl.: and), att (Engl.: to) or i (Engl.: in).

TF-IDF

In another approach, the Term Frequency-Inverse Document Frequency (TF-IDF) weight was assigned to all terms which is the “most common and classic method of weighting” (Shi et al., 2011, p. 131) in text mining. TF was defined above. IDF is, according to Manning et al. (2008, p. 117 f.), a mechanism used in combination with TF, to attenuate the effect of words that occur too often in the set of documents as that they could be important in order to discriminate between those. IDF is calculated as follows: \( \text{idf}_t = \log \frac{N}{\text{df}_t} \) where \( N \) is the number of documents in a collection and \( \text{df}_t \) is the document frequency of term \( t \), i.e. the number of documents in the collection that contain \( t \). TF-IDF for a term is calculated by: \( \text{tfidf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t \). Thus, TF-IDF for a \( t \) is highest if \( t \) occurs many times within a small number of documents. We reduced the number of features by keeping the 200, 100, 70 and 50 terms with the highest TF-IDF scores. For more information on TF-IDF and different weighting schemes see Manning et al. (2008).

Infection-specific terms (IST)

All techniques presented so far represent automatic preprocessing methods. Witten et al. (2011) state, however, that the best way to select relevant features is manually, given a deep understanding of the learning problem and what features actually mean. In this regard, the IST-terminology which was presented in Section 4.3 was deployed as an additional feature reduction technique. When applying this method, all terms were removed from the HSRs except for those which occur in the terminology. This techniques was initially motivated by the assumption that infection-specific terms would occur in HSRs which contain HAI while being absent in HSRs which do not contain HAI, thus, marking descriminative features for the classifier. By means of this procedure, the feature space was decreased to 374.

5.4 Performance evaluation

In this last section of Chapter 5, 10-fold cross-validation is presented as the evaluation techniques used (Section 5.4.1). In the following the statistical test which is applied is presented (Section 5.4.2):

5.4.1 10-fold cross-validation

For evaluating each classifier independently, stratified 10-fold cross-validation, one of the best known and most commonly used evaluation techniques was used; cf. Kohavi (1995, p. 1138 f.) and Lavesson (2006, p. 23 ff.). When applied, the dataset of 213 HSRs is divided into 10 folds, the classifier is then trained on 9 folds and tested on the remaining one. The procedure is repeated until all folds

⁶http://snowball.tartarus.org/algorithms/swedish/stop.txt
have been used for testing ones. Stratification ensures that each class, given the different classification tasks, that is HAI and NoHAI (binary classification), INF and NoINF (two-step classification), and HAI, CAI and NoINF (multi-class classification), is properly represented in each fold with respect to the class distribution over the entire dataset. For example, in the binary classification task 129 or about 61% of all HSRs belong to class HAI, thus each subset should consist of roughly 61% HSRs which belong to class HAI; cf. (Lavesson 2006, p. 23). Cross-validation is especially useful if the dataset, as in the present case, is small, as it maximizes the amount of training data. Thus, it is considered to be good method for estimating the true performance; cf. (Witten et al. 2011, p. 305). Moreover, the leave-one-out approach, a special case of the k-fold cross-validation, would have been appropriate for evaluating the performance. The advantage of this approach is that it uses as much data as possible for training by setting $k = N$, the size of the data set which is 213 in the present case. In this approach each test set only contains one record, i.e., HSR. However, as Tan et al. (2006) state, this approach is computationally expensive and was therefore not applied in this study but can be considered for future tasks.

5.4.2 Statistical tests

When comparing the classifiers’ results, statistical testing is necessary in order to verify the significance of the results. In this study, the non-parametric sign test is used. The choice was motivated by the fact that the authors in Japkowicz and Shah (2011) present this statistical test as being simple to calculate and yet appropriate when wanting to compare the performance of multiple classifiers on a single domain. Just like (Japkowicz and Shah 2011, p. 231 ff.) do in their example calculations, the sign test was one-tailed and performed at 5% significance level. The null hypothesis states that the classifiers perform equally well. When applying the sign test to compare classifiers on one domain, multiple trials are made on the domain, i.e., by performing cross-validation. For each preprocessing technique, the performances of the classifiers are compared and statistically tested based on the classifiers’ results on each fold. These results are not shown due to space restrictions. Instead, the average performance measures are shown in the results table for the binary (Table 6.1), two-step (Tables 6.2 and 6.3) and multi-class (Table 6.4) classification.
6 Results

This chapter starts with presenting the results of the three classifiers for each classification task separately: Section 6.1 displays the results of the binary classification, Section 6.2 of the two-step classification and Section 6.3 of the multi-class classification. Subsequently, the results of the three classification tasks are compared to another in Section 6.4. The chapter is finalized by summarizing the results in Section 6.5.

The focus of this study lies on obtaining high recall for HSRs which contain HAI. Thus, only the performance measures of the classifiers for those records are presented. In the course of presenting the results of each classification task, Tables 6.1 to 6.4 illustrate the respective Precision (P), Recall (R) and F1-score for each classifier given the different preprocessing methods.

For each classification task, the results were analyzed with regard to the following aspects:

- comparing the classifiers and their obtained recall values to one another, regarding the question whether any of the classifier outperform another, making it (them) more applicable for our task
- comparing the preprocessing techniques with one another, given the recall values obtained by the different classifiers,
- analyzing the obtained recall values with regard to the F1-scores and in the course of this
- pointing out the combination of classifier and preprocessing method which comes closest to the objective of obtaining high recall with the highest precision possible.

For ensuring readability the combination of classifier and preprocessing method are referred to as follows in the subsequent sections: NB-noprepro - Naïve Bayes + no preprocessing, NB-lemma - Naïve Bayes + lemmatization, NB-stop - Naïve Bayes + stop word removal, NB-ist - Naïve Bayes + infection specific terms and NB-tfidf - Naïve Bayes + TF-IDF 50. The combination of preprocessing techniques and SVM and C4.5, respectively is expressed likewise.

6.1 Binary classification

This section presents the results for the binary classification task.
Table 6.1 depicts the results of Naïve Bayes, SVM and C4.5 for the binary classification, given the different preprocessing methods.

As visual in the table, SVM yields the highest recall for three out of five preprocessing methods: 89.1% (SVM-tfidf), 79.1% (SVM-lemma) and 78.3% (SVM-stop). For the remaining two preprocessing methods C4.5 attains the highest recall: 85.3% (C4.5-noprepro) and 68.2% (C4.5-ist). The following tendencies can be noted when comparing the three classifiers to one another:

- NB yields the lowest recall values for all five preprocessing techniques. These are, compared to SVM, considerably lower, in average 14.2 percentage points, and in four out of five cases significantly lower. Compared to C4.5, the recall values obtained by NB are considerably, yet, in the majority of the cases not significantly lower, in average 12.9 percentage points.

- By contrast, either SVM or C4.5 obtain the highest or second highest recall values for all five preprocessing methods. With an average difference of 4.82 percentage points, their obtained recall values, however, do not differ significantly for any of the preprocessing methods.

- The three highest recall values for the binary classification are 89.1% (SVM-tfidf), 85.3% (C4.5-noprepro) and 82.9% (C4.5-tfidf).

The recall values given the different combinations of classifier and preprocessing method are displayed in Figure 6.1, clearly illustrating the above stated.

---

1. Exception: SVM-terms and to NB-terms are not significantly different.
2. Exception: C4.5-noprepro and NB-noprepro differ significantly.
In order to grasp the effect which the different preprocessing methods have, the recall values are to be compared per preprocessing method by setting the performance of NB-noprepro (60.5%), SVM-noprepro (79.8%) and C4.5-noprepro (85.3%) as a baseline for each classifier. Both NB and SVM obtain their highest recall when TF-IDF 50 is applied, 78.3% and 89.1%. For both classifiers, these recall values are significantly better than the ones obtained by all other preprocessing methods, including the baseline. By contrast, C4.5 yields its highest recall when no preprocessing is applied, 85.3%. Thus, applying TF-IDF 50 leads to a statistically insignificant decrease in recall for C4.5 to 82.9%.

For all three classifiers, applying lemmatization and stop word removal, leads to a drop in recall, albeit not significant. NB does in fact yield its lowest recall when stop words are removed, that is, 58.1%. Applying IST leads to a considerable decrease in recall for SVM and a significantly lower recall for C4.5 compared to the baseline, resulting in the lowest recall values for both classifiers, i.e. 65.1% and 68.2%. By contrast, NB yields its second highest recall when IST is applied, 63.6%. It is interesting to note that both SVM and C4.5 show a considerable, if not yet in all terms significant, drop in recall for SVM/C4.5-ist, while NB shows an increase in recall for just this preprocessing method. The tendency of a general increase in recall when TF-IDF 50 is applied as well as the fact of an overall low recall when IST is applied is visible in Figure 6.1.

After pointing out how recall varies, given the three classifiers and different preprocessing methods, it is important to analyze the respective F1-score for the highest recall values. It is observable that the highest F1-scores are obtained in correlation with the highest recall values, that is F1 = 76.9% for SVM-tfidf, F1 = 78.9% for C4.5-noprepro, and F1 = 72.1% for C4.5-tfidf. Just as for the recall values, none of the F1-scores vary significantly. These correlations point to SVM-tfidf and C4.5-noprepro as coming closest to the objective of obtaining high recall with a reasonable overall performance. SVM-tfidf obtains the highest recall, 89.1%, with a slightly lower overall performance, 76.9%, while C4.5-noprepro shows the overall best performance, obtaining an insignificantly lower recall, R = 85.3% and F1 = 78.9%.

Concluding this section, the findings for the binary classification task can be summarized as follows:
• NB obtains considerably lower recall values for all preprocessing methods than SVM and C4.5 which perform quite similar.

• Compared to the baseline, applying TF-IDF 50 results in a fairly high recall for all three classifiers and also a significantly higher recall for NB and SVM.

• SVM-tfidf and C4.5-noprepro come closest to the objective of obtaining high recall with a reasonable overall performance.

6.2 Two-step classification

This section presents the results for the two-step classification task. As mentioned in Section 5.1.2, the emphasis of this classification task was to try optimizing classification results, and the recall values for HAI in particular, by incorporating the findings presented in Section 5.1.1.

This section starts by presenting the results of the INF-NoINF classifier during the first classification step (Section 6.2.1). Subsequently, the performance results of the two-step classification task as a whole are presented, i.e. the results that are yielded when the INF-NoINF and HAI-CAI classifier are applied consecutively on the test set (Section 6.2.2).

6.2.1 First classification step

In the first classification step, a classifier was trained to predict if HSRs contain INF or NoINF. As described in Section 5.1.2, all HSRs which were predicted to contain an infection were used in order to build a test set for the second classification step. Thus, the performance of the classifier during the first classification step is crucial, since it directly determines the 'quality' of the dataset used in the second classification step. It is therefore inevitable to look at the results of the first classification step before analyzing the results of the two-step classification task in its entirety.

Table 6.2 illustrates the results of Naïve Bayes, SVM and C4.5 for the first classification step, given the different preprocessing methods. In other words, the table depicts the performance results of the INF-NoINF classifier on the test set.

<table>
<thead>
<tr>
<th>Preprocessing</th>
<th>Naïve Bayes</th>
<th>SVM</th>
<th>C4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F₁</td>
</tr>
<tr>
<td>No preprocessing</td>
<td>83.3</td>
<td>59.2</td>
<td>69.2</td>
</tr>
<tr>
<td>Lemmatized</td>
<td>84.5</td>
<td>61.2</td>
<td>71.0</td>
</tr>
<tr>
<td>No stop words</td>
<td>85.7</td>
<td>59.2</td>
<td>70.0</td>
</tr>
<tr>
<td>IST</td>
<td>86.0</td>
<td>68.4</td>
<td>76.2</td>
</tr>
<tr>
<td>TF-IDF 50</td>
<td>77.8</td>
<td>82.9</td>
<td>80.3</td>
</tr>
</tbody>
</table>

Table 6.2: Precision, Recall and F₁-score (in %) for the first classification step
As becomes visible in the table, SVM obtains the highest recall for three preprocessing methods: 90.1% (SVM-lemma), and 87.5% for both SVM-noprepro and SVM-stop. When IST is applied as a preprocessing method, C4.5 yields the highest recall of 78.9%. For TF-IDF-50, SVM and C4.5 obtain the same high recall of 90.8%.

When comparing the classifiers in their performance to one another, the following can be noted:

- Naïve Bayes yields the lowest recall for all five preprocessing methods. For all preprocessing methods but NB-terms, the yielded recall values are significantly lower than those of SVM. The recall values of NB and C4.5 differ significantly for three preprocessing methods: no preprocessing, lemmatization and stop word removal. With regard to the remaining two preprocessing methods, the differences in recall are not significant.
- Either SVM or C4.5 yield the highest recall values for all preprocessing methods. When compared to another, however, their results are in the majority of the cases not significantly different.
- The three highest recall values for the first step of the two-step classification are 90.8% (SVM- and C4.5-tfidf), 90.1% (SVM-lemma) and 87.5% (SVM-noprepro).

From Table 6.2 it appears that all three classifiers obtain their highest recall when TF-IDF 50 is applied: 82.9% (NB-tfidf), 90.8% (SVM-tfidf) and 90.8% (C4.5-tfidf). Compared to the baseline when no preprocessing is applied, the result of NB-tfidf is significantly better than all other preprocessing methods, including the baseline. The result of SVM-tfidf is significantly better than of SVM-terms. Compared to all other preprocessing methods there is, however, no significant difference in the obtained recall values. C4.5-tfidf performs observatively, yet not significantly, better than C4.5-stop and C4.5-noprepro. By contrast, C4.5-tfidf performs significantly better than C4.5-terms and C4.5-lemma.

The results presented above and in Table 6.2 indicate, how well the classifiers perform in predicting HSRs which contain INF, given the different preprocessing methods. Yet, they do neither reveal how many of the predicted INFs actually contain HAI and CAI, nor how many of the predicted INFs in fact belonged to class NoINF. Since the overall aim of the two-step classification task is to filter out all HSRs which contain HAI, it is interesting to analyze, how many HSRs containing HAI are actually 'lost' during the first classification step.

For this purpose, some class statistics of the 10 test folds during the first classification step need to be presented. The test folds contain either 21 or 22 HSRs. Of these HSRs, per test fold, between 11 and 15 HSRs contain HAI as their actual class, while between 1 and 4 contain CAI. Together they add up to 15 or 16 HSRs which contain INF. Thus, per test fold either 6 or 7 HSRs remain which contain NoINF. Figure 6.2 images the percentage of how many HAIas are lost during the first classification step, given the different combinations of classifiers and preprocessing methods.

\[\text{Exception: SVM-lemma and C4.5-lemma differ significantly}\]
\[\text{The percentage depicted is the average of the losses of all 10 folds. Given the number of}\]
34
It becomes clear that, overall, most HAIIs are lost when NB is applied, depending on the preprocessing method between 17.8% (NB-tfidf) and 37.4% (NB-noprepro and NB-stop). SVM performs overall best in regard to not loosing, i.e., identifying most HAI in the test sets. Especially SVM-lemma and SVM-tfidf result in a low loss in HAI, i.e., 7.4% and 8.5%. C4.5-tfidf yields a very similar result, that is only 7.8% of all HAIIs are not predicted.

Figure 6.3 depicts how many CAIs are lost in comparison. Most CAIs are lost when NB is applied, whereat the highest losses are obtained by NB-noprepro and NB-stop, 58.3% as well as NB-terms 61.7%. Only NB-tfidf performs considerably better. The fewest HSRs containing CAIs are lost when in fact NB-tfidf or C4.5-stop are applied, i.e., only 12.5%. For SVM, there are no great differences in the results obtained for the different preprocessing methods. Fewest CAIs are lost when SVM-tfidf is applied, 17.5%, while most are lost when SVM-lemma is deployed, 28.3%. In average the results obtained by C4.5 are quite similar to those of SVM, given the different preprocessing methods. It does strike, however, that C4.5-lemma looses unproportionally many HSRs containing CAI compared to the average of the five processing methods, i.e., 48.3%.

After analyzing the amount of HAIIs and CAIs that were missed throughout the first classification step, it is of further interest, how many HSRs containing NoINF were misclassified as INF. This is shown in Figure 6.4. For four out of five preprocessing methods, no preprocessing, lemmatization, stop word removal and IST, Naïve Bayes clearly misclassifies less NoINFs as INFs, than SVM and C4.5, i.e., between 24.8% (NB-stop) and 29.8% (NB-noprepro). SVM misclassifies in average 16.8% more and C4.5 even 22.9% more NoINF-HSRs as INF. While the percentage of misclassification between those four preprocessing methods does not vary considerably, it is clearly visible that TF-IDF 50 results
in a significantly higher percentage of misclassifications for all three classifiers. Of those three, C4.5-tfidf misclassifies the most NoINF-HSRs as INFs, 73.8%, while SVM-tfidf misclassifies 70.2% and NB-tfidf 59.5%.

As mentioned previously, the first classification step directly determines the quality of the second classification. As shown, a number of HSRs containing HAIs are already lost during this first step which is going to effect the second classifications step, as not all HAIs which were originally contained in the test set can be retrieved anymore. In general, a greater percentage of CAIs than HAIs were lost throughout the first classification step, which is positive to the effect that we aim at retrieving all HSRs containing HAI. The exact percentage varied however considerably, depending on the combination of classifier and preprocessing method used. Applying NB resulted in a high loss of both HAIs and CAIs, for all but the NB-tfidf method when compared to SVM and C4.5. SVM-lemma, C4.5-tfidf and SVM-tfidf resulted in the fewest losses of HAI while at the same time the losses in CAI were nearly twice as high, 7.4% (HAI) and 28.3% (CAI) for SVM-lemma, 7.8% (HAI) and 21.7% for C4.5-tfidf, and 8.5% (HAI) and 17.5% (CAI) for SVM-tfidf.

Moreover, it could be observed that a fairly high number of HSRs which do not contain an infection were misclassified as such. Interestingly, SVM-tfidf and C4.5-tfidf which were pointed out to result in the lowest losses of HAI yield the highest percentage of misclassified NoINFs, 70.2% and 73.8%. Even NB-tfidf misclassifies a large amount of NoINFs, i.e. 59.5%. While only a vague tendency points towards TF-IDF 50 as a preprocessing technique which results in few losses of HAIs and CAIs, it can be clearly noted that TF-IDF 50 results in a considerably larger amount of misclassifications of NoINFs than all other preprocessing methods for all three classifiers. To which extend these findings will influence the second classification step will be analyzed in the next section.

6.2.2 Second classification step

Table 6.3 lists the results of Naïve Bayes, SVM and C4.5 for the second classification step, given the different preprocessing methods. In other words, the table depicts the performance results that are yielded when the INF-NoINF and HAI-CAI classifier are applied consecutively on the test set. First, the results are presented in general before relating them to the findings in the previous section.
As visible in the table, SVM yields the highest recall for three out of five preprocessing methods: 91.0% (SVM-lemma), 87.9% (SVM-noprepro) and 87.2% (SVM-stop). For the remaining two preprocessing methods, C4.5 obtains the highest recall values: 92.2% (C4.5-tfidf) and 71% (C4.5-ist).

The following tendencies can be noted when comparing the three classifiers to one another:

- NB yields the lowest recall values in four out of five preprocessing techniques. These results are, compared to SVM, significantly lower\(^5\). Solely when IST is applied, NB yields a slightly better recall than SVM, 67.1% compared to 66.3% which is, however, not statistically significant. When compared to C4.5, the results of NB-stop and NB-tfidf are significantly lower, for the remaining preprocessing methods no significant difference could be noted.

- Either SVM or C4.5 obtain the highest recall values for all preprocessing methods. For all preprocessing methods where SVM obtains the highest recall, the results are significantly better than those obtained by C4.5. When comparing the recall values of SVM and C4.5 were C4.5 obtained a higher recall none of the values are significant.

- The three highest recall values are: 92.2% (C4.5), 91.0% (SVM-lemma) and 90.6% (SVM-tfidf).

Applying TF-IDF 50 tends to result in higher recall values: Both NB and C4.5 yield their highest recall when TF-IDF 50 is applied, 80.6% and 92.2%. For both classifiers, these recall values are significantly better than those obtained by all other preprocessing methods, including the baseline. SVM only obtains its second highest recall when TF-IDF 50 is applied, 90.6%. However the difference to the highest recall, 91.0% (SVM-lemma) is not statistically significant. Applying lemmatization results in a recall which is about the same as for no preprocessing for all three classifiers, in other words, lemmatization does neither lead to a major decrease nor increase in recall. When removing stop words, the same as for as for lemmatization can be stated. NB yields its lowest recall value when stop words are removed, 58.1%.

Applying IST yields the lowest recall values for SVM and C4.5, 66.3% and 71.0%. Considering SVM, this recall value is significantly lower than the recall values obtained by the classifier given all other preprocessing methods. For

---

\(^5\)Exception: SVM-terms and to NB-terms are not significantly different.
C4.5, this recall value is significantly lower than C4.5-tfidf, but not statistically different compared to the recall values yielded for all other preprocessing methods. NB on the other hand yielded its second highest recall, 67.1%, value when IST was applied. This recall is significantly better than the baseline. However, this recall does not come close to the highest recall values. Thus, the gain in applying NB-ist in order to obtain high recall is negligible.

The above stated becomes clearly visible in Figure 6.5 which illustrates the various recall values for the different combinations of classifier and preprocessing method:

![Figure 6.5: Recall for the two-step classification](image)

When analyzing the three highest recall values in comparison to their F1-scores, SVM-lemma and C4.5-tfidf come closest to the objective of obtaining high recall with a reasonable overall performance, whereat C4.5-tfidf obtains the highest recall with a slightly lower overall performance, $R = 92.2\%$ and $F_1 = 76.6\%$, and SVM-lemma shows the overall best performance, $F_1 = 82.4\%$, obtaining an insignificantly lower recall, 91.0%.

The following can be recorded for the two-step classification:

- NB yields the lowest recall for the majority of the preprocessing methods whereas either SVM or C4.5 obtain the highest recall values for all preprocessing methods. SVM performs significantly better than NB and C4.5 in three out of five preprocessing methods.

- Compared to the baseline, applying TF-IDF results in a higher recall for all three classifier as well as a significantly higher recall for NB and C4.5.

- SVM-lemma and C4.5-tfidf come closest to the objective of obtaining high recall with a reasonable overall performance.

At this point it is interesting to compare the first and second classification step to one another. It is noticeable that, while the recall values remain similar, the precision is considerably higher during the first classification step than during the second classification step. For Naïve Bayes, the precision is, for all five preprocessing methods in average, around 7 percentage points higher while the recall differs in average by only 1.06 percentage points. For SVM, the precision is in average 10.28 percentage points higher, while again recall differs...
by only 1.27 percentage points. For C4.5 the tendencies are less explicit. While the precision is 8.82 percentage points higher, recall differs by 5.76 percentage points in average. It is further interesting to compare and discuss the results of the first classification step to the results of the binary and multi-classification task in a similar manner. However, the results of the multi-class classification have to be presented first before doing so in Section 6.4.

6.3 Multi-class classification

This section presents the results for the multi-class classification.

<table>
<thead>
<tr>
<th>Naïve Bayes</th>
<th>SVM</th>
<th>C4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>R</td>
<td>F₁</td>
</tr>
<tr>
<td>77.8</td>
<td>59.7</td>
<td>67.5</td>
</tr>
<tr>
<td>76.6</td>
<td>86.0</td>
<td>81.0</td>
</tr>
<tr>
<td>74.1</td>
<td>77.5</td>
<td>75.8</td>
</tr>
<tr>
<td>Lemmatized</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>77.8</td>
<td>59.7</td>
<td>67.5</td>
</tr>
<tr>
<td>75.7</td>
<td>89.1</td>
<td>81.9</td>
</tr>
<tr>
<td>72.0</td>
<td>79.8</td>
<td>75.7</td>
</tr>
<tr>
<td>No stop words</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>78.4</td>
<td>58.9</td>
<td>67.3</td>
</tr>
<tr>
<td>73.2</td>
<td>84.5</td>
<td>78.4</td>
</tr>
<tr>
<td>72.6</td>
<td>76.0</td>
<td>74.2</td>
</tr>
<tr>
<td>IST</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>78.3</td>
<td>64.3</td>
<td>70.6</td>
</tr>
<tr>
<td>68.7</td>
<td>61.2</td>
<td>64.8</td>
</tr>
<tr>
<td>67.7</td>
<td>65.1</td>
<td>66.4</td>
</tr>
<tr>
<td>TF-IDF 50</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>67.7</td>
<td>81.4</td>
<td>73.9</td>
</tr>
<tr>
<td>67.0</td>
<td>91.5</td>
<td>77.4</td>
</tr>
<tr>
<td>65.3</td>
<td>87.6</td>
<td>74.8</td>
</tr>
</tbody>
</table>

Table 6.4: Precision, Recall and F₁-score (in %) for the multi-class classification

Table 6.4 summarizes the results of Naïve Bayes, SVM and C4.5 for the multi-class classification, given the different preprocessing and feature selection methods.

As becomes visible in Table 6.4 SVM yields the highest recall for four out of five preprocessing methods: 91.5% (SVM-tfidf), 89.1% (SVM-lemma), 86.0% (SVM-noprepro) and 84.5% (SVM-stop). For the remaining preprocessing method, C4.5 obtains the highest recall: 65.1% (C4.5-ist).

- NB yields the lowest recall values in four out of five preprocessing techniques. Compared to SVM, the results obtained by NB are for most preprocessing methods significantly lower. Solely when IST is applied, NB yields a slightly better recall than SVM, 64.3% compared to 61.2% which is, however, not statistically significant. When compared to C4.5, the NB obtains significantly lower results when no preprocessing, lemmatization and stop word removal are applied as preprocessing techniques. For the remaining two preprocessing the results are not significantly different.

- Either SVM or C4.5 obtain the highest recall values for all preprocessing methods. Except when lemmatization or stop word removal are applied, none of the results are significantly different.

- The three highest recall values are: 91.5% (SVM-tfidf), 89.1% (SVM-lemma) and 87.6% (C4.5-tfidf).

All three classifiers yield their highest recall when TF-IDF 50 is applied as a preprocessing method: 81.4% (NB-tfidf), 91.5% (SVM-tfidf) and 87.6% (C4.5-tfidf). When compared to the respective baseline, i.e., 59.7% (NB-noprepro),
86.0% (SVM-noprepro) and 77.5% (C4.5-noprepro), the recall obtained by NB-tfidf is significantly better, while the recall values of SVM and C4.5 increased, but do not differ significantly. For SVM and C4.5, applying IST results in the lowest recall compared to all other preprocessing methods, 61.2% and 65.1%, respectively. Compared to the respective baselines SVM-ist yields a significantly lower recall while C4.5-ist yields a considerably, yet not significantly, lower recall. When applied to NB, IST yields in an increase in recall. However it is not significantly better than the recall obtained by the NB baseline. When lemmatization and stop word removal are applied as a feature selection method, the obtained recall values of all three classifiers are about the same as for the baseline, being slightly lower or higher, yet not differing significantly.

Figure 6.6 clearly visualizes the above stated.

![Figure 6.6: Recall for the multi-class classification](image)

When analyzing the highest recall values in comparison to their F1-scores, SVM-lemma and SVM-tfidf come closest to the objective of obtaining high recall with a reasonable overall performance. SVM-tfidf obtains the highest recall with a slightly lower overall performance, R = 91.5% and F1 = 77.4%, and SVM-lemma shows the overall best performance, F1 = 81.9%, while obtaining an insignificantly lower recall, 89.1%.

The results of this classification task can be summarized as follows:

- NB yields the lowest recall for the majority of the preprocessing methods whereas either SVM or C4.5 obtain the highest recall values for all preprocessing methods.
- Compared to the baseline, applying TF-IDF results in a higher recall for all three classifier and even a significantly higher recall for NB.
- SVM-lemma and SVM-tfidf come closest to the objective of obtaining high recall with a reasonable overall performance.

### 6.4 Comparison of the three classification tasks

After analyzing the results of the different classification tasks separately, this section will focus on comparing the results to another.
As repeatedly stated, the aim was to obtain high recall with the highest precision possible. The combinations of classifier and preprocessing method which came closest to our objective for each classification task were named at the end of Sections 6.1, 6.2 and 6.3. The conclusions were drawn from the fact that they either obtained the highest recall with a slightly lower overall performance or the best performance, i.e., highest F1-score, with a insignificantly lower recall. Since we do, as mentioned in Section 2.3.2, weigh recall higher than precision, it is valuable to compare the recall values of all three classification tasks in relation to their according F2-scores, in order to point out the combination of classifier and preprocessing method which comes closest to our objective when comparing all three classification tasks. Tables 6.5, 6.6 and 6.7 depict the recall and F2-values obtained during the binary, two-step and multi-class classification.

<table>
<thead>
<tr>
<th></th>
<th>NB</th>
<th>SVM</th>
<th>C4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>F2</td>
<td>R</td>
</tr>
<tr>
<td>Nopropro</td>
<td>60.5</td>
<td>63.2</td>
<td>79.8</td>
</tr>
<tr>
<td>Lemma</td>
<td>59.7</td>
<td>62.6</td>
<td>79.1</td>
</tr>
<tr>
<td>Stop</td>
<td>58.1</td>
<td>61.2</td>
<td>78.3</td>
</tr>
<tr>
<td>IST</td>
<td>63.6</td>
<td>66.0</td>
<td>65.1</td>
</tr>
<tr>
<td>TFIDF</td>
<td>78.3</td>
<td>75.7</td>
<td>89.1</td>
</tr>
</tbody>
</table>

Table 6.5: Recall and F2-scores for the binary classification task

<table>
<thead>
<tr>
<th></th>
<th>NB</th>
<th>SVM</th>
<th>C4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>F2</td>
<td>R</td>
</tr>
<tr>
<td>Nopropro</td>
<td>59.6</td>
<td>62.6</td>
<td>87.9</td>
</tr>
<tr>
<td>Lemma</td>
<td>60.2</td>
<td>63.1</td>
<td>91.0</td>
</tr>
<tr>
<td>Stop</td>
<td>58.1</td>
<td>61.4</td>
<td>87.2</td>
</tr>
<tr>
<td>IST</td>
<td>67.1</td>
<td>69.3</td>
<td>66.3</td>
</tr>
<tr>
<td>TFIDF</td>
<td>80.6</td>
<td>77.4</td>
<td>90.6</td>
</tr>
</tbody>
</table>

Table 6.6: Recall and F2-scores for the two-step classification task

<table>
<thead>
<tr>
<th></th>
<th>NB</th>
<th>SVM</th>
<th>C4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>F2</td>
<td>R</td>
</tr>
<tr>
<td>Nopropro</td>
<td>59.7</td>
<td>62.6</td>
<td>86.0</td>
</tr>
<tr>
<td>Lemma</td>
<td>59.7</td>
<td>62.6</td>
<td>89.1</td>
</tr>
<tr>
<td>Stop</td>
<td>58.9</td>
<td>62.0</td>
<td>84.5</td>
</tr>
<tr>
<td>IST</td>
<td>64.3</td>
<td>66.7</td>
<td>61.2</td>
</tr>
<tr>
<td>TFIDF</td>
<td>81.4</td>
<td>78.2</td>
<td>91.5</td>
</tr>
</tbody>
</table>

Table 6.7: Recall and F2-scores for the multi-class classification task
When comparing the performance of the three classifiers during the different classification tasks, the following can be stated:

- The recall values obtained by NB are almost exactly the same for all but the TF-IDF 50 and IST preprocessing methods, i.e., during all three classification tasks, NB-noprepro, NB-lemma and NB-stop yield an average recall of 59.4%. By contrast, a slightly more distinct variation in recall could be noted for TF-IDF 50 and IST. NB-ist is lowest during the binary classification, 63.6%, and highest during the two-step classification, 67.1%. From Sections 6.1, 6.2 and 6.3 it can be derived that TF-IDF yields the highest recall values for NB in all three classification tasks. Thereby, NB-tfidf is lowest during the binary classification task, 78.3%, and highest during the multi-class classification, 81.4%. It strikes that, for NB-noprepro to NB-stop, the $F_2$-scores are insignificantly higher than the obtained recall, in average 2.8 percentage points. By contrast, NB-tfidf yields the highest recall, yet the $F_2$-scores drop considerably during all three classification tasks, that is, by three percentage points in average. However, the $F_2$-scores obtained by NB-tfidf are still significantly higher than those obtained by all other preprocessing methods, in average 77.1% compared to 63.6%. The best result of NB-tfidf is yielded during the multi-class classification with a recall of 81.4% and $F_2$-score of 78.2%.

- The recall values obtained by SVM-noprepro, SVM-lemma, SVM-stop and SVM-ist during the binary classification are considerably lower than those yielded during the two-step and multi-class classification, in average 75.6% (binary) compared to 83.1% (two-step) and 80.2% (multi-class). This gain could be a result of the distinction made between HAI and CAI-class. By contrast, the results obtained for SVM-tfidf are about the same during all three classification tasks, whereat, the lowest recall is obtained during the binary classification, 89.1%, and the highest during the multi-class classification, 91.5%. Yet, the values do not differ significantly. Since the recall values obtained by SVM-noprepro, SVM-lemma, SVM-stop and SVM-ist during the two-step and multi-class classification are higher than those obtained during the binary classification task, the gain in recall obtained by SVM-tfidf for those two classification tasks is less significant. In all three classification tasks, SVM-tfidf yields the highest (binary and multi-class) or second highest (two-step) recall. This gain is, however, offset by substantial loss of precision. It becomes visible by means of the $F_2$-score which, compared to all other preprocessing methods, barely increases (binary classification), respectively remains almost the same (two-step and multi-class classification). The payoff becomes especially clear, when comparing the results obtained by SVM-tfidf and SVM-lemma during the multi-class classification. While SVM-tfidf obtains a higher recall, 91.5% compared to 89.1%, SVM-lemma still yields the best overall performance, 86.1% compared to 85.3%. The best result is obtained by SVM-lemma during the two-step classification, that is 91.0% recall and 87.4% $F_2$-score.

- The results yielded by C4.5 are not as explicit. Except for the binary classification, C4.5-tfidf yields the highest recall. The price is, however,
paid in terms of loss in precision. Still the overall performance remains higher than for all other preprocessing methods. Noticable for the performance of C4.5 is the high recall and F\textsubscript{2}-score, obtained during the binary classification, i.e., 85.3% recall with a 82.6% overall performance. Yet, the best result is yielded by C4.5-tfidf during the two-step classification with a recall of 92.2% and an overall performance of 85.2%.

When comparing the classifiers to one another, the following can be noted for the different classification tasks. For each classification task,

- SVM yields the highest recall values for all preprocessing methods for all three classifiers in the majority of cases. Compared to NB the results are significantly better in most instances, while they tend to be better compared to C4.5, yet not significantly in all cases. At the same time, SVM yields comparatively higher F\textsubscript{2}-scores in almost all cases.
- NB yields clearly lower recall and F\textsubscript{2}-scores than SVM and C4.5 for all but the IST preprocessing method.
- When compared to the baseline

1. applying TF-IDF 50 resulted in higher recall values in the majority of cases for all three classifiers\textsuperscript{6}. Mostly, these results were even significantly better than those obtained by all other preprocessing methods, including the baseline. However, this gain in recall is offset by a substantial loss in precision for all three classifiers in all three classification tasks. In other words, the precision lies between 63.7% (C4.5-binary) and 67.7% (NB-multi). This clearly contrast all other preprocessing methods (for all three classifiers in all three classification tasks), which in general obtain lower recall values than TF-IDF 50 but do yield precision values which are in all cases higher than 70%. Now, if we recall the observation made in Section 6.2.2 stating that applying TF-IDF 50, compared to all other preprocessing methods, resulted in a greater number of HSRs containing NoINF being incorrectly classified as INF, it becomes clear how the low precision is triggered by the high number of false positives.
2. No considerable increase in recall and F\textsubscript{2} was obtained by lemmatization or stop word removal and
3. applying IST proved to generate the lowest recall and overall performance for SVM and C4.5 in all three classification tasks.

- For the two-step and multi-class classification the recall values obtained by SVM and C4.5 for no preprocessing, lemmatization and stop word removal are conspicuously higher than those obtained by NB, in average 9.3 percentage points for SVM and 7.5% percentage points for C4.5. This could be seen as a result of the distinction made between HAI and CAI-class. As a result, even though lemmatization does not yield any considerable increase in recall, as pointed out above, the obtained

\textsuperscript{6}The only exceptions were noted for C4.5 in the binary classification and SVM in the two-step classification.
recall values are quite high. By contrast, no such change in recall can
by noted when either IST, all recall values are equally low for the three
classification tasks, i.e. around 64%, or TF-IDF 50, all recall values are
similarly high, i.e. around 90%, are applied. Yet, the gain achieved by
means of SVM-tfidf and C4.5-tfidf compared to the other preprocessing
methods is less significant.

With regard to the highest recall values, the three classification task yield
quite similar results. The highest recall of all three classification tasks is obtained
by C4.5-tfidf, 92.2%, during the two-step classification. Yet, it becomes clear
that neither the difference to the second highest recall, 91.5% (SVM-tfidf,
multi-class classification), nor the difference to the third highest recall, 91.0%
(SVM-lemma, two-step classification), is significant. Moreover, it becomes clear
that, for each classification task, the highest recall value is in fact obtained
when TF-IDF 50 is applied as a preprocessing method. However, as pointed
out multiple times, the price of high recall when applying TF-IDF 50 is paid in
terms of loss in precision.

When comparing the different classifiers and especially the two-step and
multi-class classification based on their F₂-scores, it becomes apparent that
SVM yields high F₂-scores not only for SVM-tfidf. For both classification tasks,
the F₂-scores of SVM-noprepro, 84.6% (two-step) and 83.8% (multi-class),
SVM-lemma, 87.4% (two-step) and 86.1 (multi-class), SVM-stop, 83.5% (two-
step) and 82.0% (multi-class), and SVM-tfidf, 84.0% (two-step) and 85.3%
(multi-class) are in average around 84.6%. This is significantly higher than
the average obtained by NB, 66.2%, and considerably higher than the average
obtained by C4.5, 77.2%. At the same time, the F₂-scores yielded by SVM-
noprepro, SVM-lemma, SVM-stop and SVM-tfidf during these two classification
tasks do not vary significantly.

To conclude this part, when pointing out the combination of classifier and
preprocessing method which comes closest to the objective of obtaining high
recall with the highest precision possible of all three classification tasks, it is in
fact SVM-lemma during the two-step classification task which comes closest
with a recall of 91.0% and F₂-score of 87.4%.

In section 6.2.2, the results of the first classification step were compared
to those of the second classification step, showing that, while recall remains
similar, precision is considerably higher during the first classification step than
during the second classification step. Similar tendencies can be found when
comparing the performance results of the first classification step to the results of
the binary classification and multi-class classification. Compared to the results
obtained during the binary classification, for Naïve Bayes, the precision during
the first classification step is, for all five preprocessing methods, in average
7.98 (Naïve Bayes), 7.96 (SVM) and 7.82 (C4.5) percentage points higher while
recall differs by 2.14 (Naïve Bayes), 8.04 (SVM) and 5.72 (C4.5) percentage
points. The results for C4.5 are not as explicit as for NB and SVM. Compared
to the results obtained during the multi-class classification, the precision is, in
average, 7.46 (Naïve Bayes), 9.18 (SVM) and 8.68 (C4.5) percentage points
higher while recall differs by 1.38 (Naïve Bayes), 3.86 (SVM) and 5.42 (C4.5)
percentage points. Again, the results are not as explicit for C4.5 as for NB and
SVM.
The fact that the precision is considerably higher during the first classification step than during the binary and multi-class classification as well as second classification step, for all three classifiers and all preprocessing methods, indicates that the classifiers do profit appreciably from differentiating between HSRs which contain an infection and those which do not, that is, they seem to find more discriminating features and thus yield better performance results. This in turn can be seen as confirming the assumption given by the medical experts, stating that HSRs containing an infection differ with regard to the vocabulary used in the medical records compared to those which do not. The fact that precision is lower during the multi-class classification respectively drops during the second classification step shows that discriminating HAI from CAI is a more difficult task which can be explained by a shared vocabulary.

In summary, the results of our approaches show that discriminating between HSRs containing and infection and those which do not helps to increase precision considerably by lowering the number of false positives. Even recall is by trend higher when only discriminating between HSRs which contain an infection and those which do not. This is especially noticeable when comparing the average recall of SVM during the first classification step which is 8.04 percentage points higher than during the binary classification. Discriminating between HAI and CAI is more difficult and lowers precision results. This may even explain the fact that in the End the results obtained during the two-step and multi-class classification are only marginally better than those yielded during the binary classification task.

6.5 Summary of results

The results of the different classification tasks show that a recall of above 90% is possible while keeping a reasonable $F_2$-score. The best performance result was yielded by SVM-lemma during the binary classification task, yielding a recall of 91% with an $F_2$-score of 87.4%.

When comparing the performance of the three classifiers throughout the three classification tasks, SVM and C4.5 performed in general better than NB, indicating a greater applicability of these two classifiers for our task. SVM and C4.5 led to fairly similar results during the binary classification task. However, during the two-step and multi-class classification, especially SVM seems to have benefit from the approach of differentiating between HAI and CAI which becomes visible in conspicuously better $F_2$-scores.

The results presented in the previous sections have clearly shown that applying TF-IDF 50 results in a definite increase in recall for all three classifiers in all three classification task. For that reason it is worthwhile to look at the features, which the classifiers base their decision upon. Interestingly, terms such as och (Engl.: and), det (Engl.: it) or hon (Engl.: she), i.e., stop words, are among the 50 top features. Intuitively, we would not expect the classifier to consider such terms in order to detect patient records which contain HAI, but rather terms like operation (Engl.: surgery), feber (Engl.: fever) or kad (Engl.: abbreviation for catheter), i.e., terms which are considered to be infection-specific by the medical experts. This result, as well as the fact that applying IST yielded fairly low results, leads to conclude that infection-specific terms, at least
in the present machine-learning approach, do not improve performance when trying to retrieve patient records which contain HAI by means of classification. Furthermore, the results show that applying TF-IDF 50 limits the classification to a comparably low precision of around 65%. As a result, when pointing out the combination of classifier which obtained high recall with the highest precision possible it is in fact not a combination of a classifier and TF-IDF 50 which come closest to the objective but SVM-lemma as pointed out previously.

As stated with regard to the performance of the classifiers, the two-step and multi-class classification tend to yield better results, albeit only marginally. The first classification step, however, yields a considerably higher precision and slightly higher recall than the binary and multi-class classification as well as the second classification step. This pinpoints the fact that discriminating between HSRs which contain and infection and those which do not, does indeed have a positive impact on performance results. Discriminating between HAI and CAI, on the other hand, seems more difficult. This can be seen to effect the final result, i.e., that the two-step and multi-class classification yield only marginally better results.
This paper focuses on deploying three machine learning algorithms to hospitalization records. By means of applying different preprocessing methods, it was tried to increase the recall values. Three different classification tasks were designed in order to address the possibilities of initially assigning the HSRs to different classes. Against initial expectations, the results of the three machine learning algorithms were all in all very encouraging. Especially the drastic feature reduction by only taking the top 50 features according to their TF-IDF scores yielded most promising recall values throughout all classification tasks. At the same time, applying TF-IDF 50 resulted in comparatively low precision which led to the result that SVM-lemma came closest to the objective of obtaining high recall with the highest precision possible during the two-step classification task, i.e., yielding a recall of 91.0% and $F_2$-score of 87.4%.

We are well aware of the fact that our dataset was small and that the differences in the results of Naïve Bayes, SVM and C4.5 are marginal and not in all cases significantly different. Yet, we are convinced that the results reveal the potential of applying machine learning techniques to patient records, including the structured as well as unstructured parts. This is further motivated by the fact that, so far, we have neither tuned parameters of the different classifiers nor used particularly elaborate preprocessing and feature reduction methods. Future research will thus have to focus upon improving the scores by, for instance, using (1) more elaborate feature selection, (2) wrapper techniques for feature reduction which are optimized on a specific learning algorithm and therefore, according to [Hall 1999], yield better results, or (3) tune parameters.

Furthermore, the medical experts involved will, in the course of the project, value 292 additional HSRs from the rheumatic clinic at Karolinska University Hospital. Thus, we will be able to train the classifiers on about twice as much data as we did now, leading us to expect an improvement in performance. In addition, we aim at training the classifiers on a more realistic dataset, i.e., a dataset which is less balanced than our current one with regard to the number of patient records containing HAI and not containing HAI, respectively.

Finally, the overall goal will continue to be obtaining high recall (approaching 100%) with the highest precision possible for hospitalization records. This will enable us to implement a system which can screen all hospitalization records, and filter out all HSRs which contain HAI. This would reduce the workload for hospital staff tremendously as they only need to analyze those HSRs which were preselected by the system.


51