Centre for Health Informatics and Multiprofessional Education

at

University College London

Learning from patient safety incidents: can data mining help?

by

Carl Reynolds

Project Supervisor

Paul Taylor

Dissertation submitted in partial fulfilment of the

Masters of Science in Health Informatics

University College London

December 2013

I hereby declare that the work presented in this thesis is my own.

Carl Reynolds

Abstract

Background To reduce preventable harm to patients health care professionals report unexpected events that harm, or nearly harm, patients. In England and Wales, the National Health Service (NHS) provides standardized patient safety incident report forms for this purpose which are collated nationally by the National Reporting and Learning System (NRLS). To date over seven million patient safety incidents have been reported to the NRLS. The magnitude of this dataset presents a major challenge for the NRLS analytic team. Two key analytic tasks are:

1. To group similar incidents in order to find common modifiable causes and inform prevention strategies

2. To classify the severity of incidents occurring in order to prioritize and target remedial efforts

Data Mining(DM), a process that includes the use of machine learning algorithms, is emerging as a useful analysis technique in a diverse range of endeavours that involve large datasets. DM techniques are not yet routinely used in operational patient safety systems but they have shown promise as an analytic tool in a variety of patient safety research settings.

It is not known whether DM could help the analysis work of the NRLS by, for example, offering efficiency gains, supporting the existing work of analysts, and permitting new insights into this large database. This is examined for a subset of NRLS patient safety incidents that relate to computer use by testing data exploration and auditing tools, the Lingo clustering algorithm, and Naive Bayes (NB) and Stochastic Gradient Descent (SGD) incident severity classifiers.

Methods A database extraction, transform, and load (ETL) approach was used as a precursor to performing cluster analysis and building an incident severity classifier.

Incidents reported as occurring between 1st January 2002 and 1st March 2012 and classified as concerning computer systems were extracted from the NRLS database.

Data were cleaned and selected fields (incident free text description and severity of incident) were converted to csv and xml data formats for subsequent analysis using Apache Solr, Scikit-learn and NLTK (csv), and Carrot2 (xml). Preprocessing techniques including stemming, tokenization, tagging, and filtering.

Extracted data were audited using Python Brewery and validated by searching for known patterns of interest using Grep, Google Refine, and Apache Solr. Data were loaded into NLTK for lexical analysis, Apache Solr to search for strings of interest for validation, Carrot2 platform to perform cluster analysis, and Scikit-learn in order to construct severity classifiers.

Lingo (a clustering algorithm based on singular value decomposition) was employed within the Carrot2 platform to perform cluster analysis on selected data. NB and SGD incident severity classifiers were tuned using a grid search strategy and evaluated using cross validation.

Results Between 1st January 2002 and 1st March 2012 7273 incidents were classified by NPSA staff as belonging to "Infrastructure (including staffing, facilities, environment)" (incident category level 1) and "IT / telecommunications failure / overload" (incident category level 2) categories. Incidents reported to have caused no harm were the most common (n = 5982). Incidents causing death (n = 7) or severe harm (n = 62) were less common. The detail provided by reporters when describing incidents varied considerably. The median number of words in the free text incident description field (IN07) was 25 and the range spanned 1-738 (this was a mandatory field).

Evidence of poor systems reliability and problems not being fixed promptly were identified as themes by manual search using Grep, Solr and NLTK. Being unable to carry out a task because of computer systems failure, and problems relating to hospital bleep system failure were identified as themes using the lingo

clustering algorithm.

Optimised NB and SGD incident severity classifiers performed similarly at predicting incident severity class from free text incident descriptions. NB classifier: precision = 0.76, recall = 0.83, f1-score = 0.77. SGD classifier: precision = 0.78, recall = 0.84, f1-score = 0.77.

Conclusion DM can offer a valuable additional technique for the patient safety analyst.

With the increasing digitisation of health care demonstration of utility may be more easily obtained when reporting, expert analysis, and action are more closely integrated into tighter feedback loops. For example a classifier used to predict prescription error might be constructed and tied to a prevention intervention with the goal of reducing the rate of a specific measurable harm occurring.

For the subset of patient safety incidents relating to computer use considered, the application of DM methods suggests that the NRLS system does not result in the timely resolution of safety issues. For safety incidents due to computer problems, and possibly other types of safety incident, lessons might be learned from more general approaches to, and cultures of, systems improvement. In particular, the more open and action focussed approach to improving quality present in bug reporting systems found in the open source software community has intuitive appeal.

Key words Health Information Systems, Patient Safety, Artificial Intelligence, Risk Management, Data Mining

Contents

Acknowledgements

I would like to thank my supervisor Dr Paul Taylor for his patient criticism and guidance. I am grateful to Sir Liam Donaldson and the National Patient Safety Agency for supporting and encouraging me through the early stages of this work. I owe a large debt to the open source software community for making, and documenting, the tools used in this thesis. Without the open source community this project would have been impossible. Finally, I am truly thankful to Ross Jones, my friend and business partner, who has been a rich source of ideas, guidance, and practical advice.

List of Tables

List of Figures

Part I

Background

"capturing and recording information on adverse events, and analysing them in the right way is an essential step to reducing risk to patients..."

Building a safer NHS for patients, Department of Health Report 2001

Chapter 1

Introduction

1.1 Overview

I am a medical doctor and an amateur software developer who has recently returned to clinical work from a secondment with the National Patient Safety Agency (NPSA). I have chosen to approach the question posed by the title of this thesis primarily from within the organizational context of the NPSA. To keep the project manageable I analyse only the specific subset of patient safety incidents that relate to computer problems and limit the data mining approaches considered.

The thesis is set out in four main parts, in part I I characterize the problem I will address and describe aspects of the NPSA's work, patient safety, and machine learning. In part II I describe and justify the data mining methods used. In part III I document the results I have obtained and explain their significance. In part IV I discuss my results in the context of the question posed by the thesis and set out my conclusions.

1.2 Problem statement

Over one million incidents are reported to the National Reporting and Learning Service (NRLS) per year, many more than would be humanly possible for the staff to manually review. Therefore in NRLS practice, routine analysis, and learning, is limited in the main to incidents which are reported to have caused serious

harm or death. A consequence of this is that there exists potentially significant unknown patterns, and learning, in the large number of incidents that it is not practicable to review centrally.

It is not known whether data mining can facilitate learning from patient safety incidents. Answering this is a hard problem. In order to proceed I will limit this thesis to consideration of the specific subset of patient safety incidents concerning computer problems. I will also limit the datamining approaches used to testing data exploration and auditing tools, the Lingo clustering algorithm, and Naive Bayes (NB) and Stochastic Gradient Descent (SGD) severity classifiers. Specifically, I will consider the following two questions in relation to patient safety incidents concerning computer problems:

1. Can free text incident descriptions be analysed in an automated fashion to generate meaningful groupings (clusters)?

2. Can the severity of harm arising from a patient safety incident be predicted using free text incident descriptions (using a classifier)?

In current practice similar incidents are grouped together in order to find common modifiable causes and inform prevention strategies. Taxonomy construction and application to incident data facilitates human understanding of the data. However, matching and taxonomy construction is time consuming and necessarily limited in scope and granularity by the preconceived categories selected to be in the taxonomy. In virtue of being a machine process unsupervised machine learning in the form of clustering is less time consuming and can be more dynamic. Further, it may provide new insights into the the incident data and suggest new categories.

Classifying the severity of incidents occurring is important to prioritize and target remedial efforts. Incident severity is included in the initial incident report but misclassification is a known problem.[3][4] An automated classifier may help to flag incidents that are misclassified and assist in central review of incident severity classification.

1.3 The NPSA

The NPSA was formed in 2001 following the publication of two landmark reports by the then Chief Medical Officer, Professor Sir Liam Donaldson. An organization with a memory[5] and Building a safer NHS[6] set out the need for greater organizational learning from safety incidents to make the NHS safer for patients.[7]

1.3.1 Patient safety and incident reporting

Recognising that a 'blame and punish' culture can inhibit reporting of safety incidents the Department of Health and the NPSA placed emphasis on the role of systems, rather than individuals acting alone, in safety incidents. Drawing inspiration from risk management in aviation, where a safety culture and incident reporting are the norm, NHS employees are now actively encouraged to report safety incidents.

What is a patient safety incident? A Patient safety incident, also called an adverse incident, occurs whenever something unexpected happens as part of medical care that harms, or nearly harms, a patient.

How do health care professionals report safety incidents? Healthcare professionals report safety incidents to the NRLS using a standardized incident reporting form. This form, called an IR1 form, is usually paper based but may also be electronic.[8] The form serves as a means for the healthcare professional to inform his or her institution when an incident has occurred.

What is contained on an incident report form? Incident reports record the following information:

1. The facts of the incident (including a free text description)
2. The perception of possible consequences (the potential or actual harm)
3. The perception of how the incident came to arise (the cause or causative factors)

What happens to incident report forms? Incident report forms are reviewed, and investigated further if necessary, by the local clinical governance team. All incident forms are then submitted electronically to the NRLS for national level analysis.

1.3.2 The National Reporting and Learning Service (NRLS)

The NRLS is the division of the NPSA concerned with the analysis of reports of patient safety incidents and safety information from other sources. On the basis of this analysis the NRLS develops and issues safety alerts to NHS organizations such as acute hospital trusts.

Safety alerts aim to reduce preventable harm to patients by raising awareness of threats to patient safety and recommending steps to reduce the risk of these threats. Alerts are typically issued is response to the identification of specific under-recognized threats to patient safety. These are threats identified through the incident reporting system that have proven to cause death or serious harm, are likely to recur, and are at least theoretically amenable to preventative measures or changes of practice.

For example, when the NPSA became aware that there were several patient deaths a year in the United Kingdom due to the accidental administration of potassium chloride solutions, it established that the main causes of accidental administration resulted from the drug being stored in the same place, and having similar packaging to, other commonly used drugs such as frusemide and normal saline. Hence, the incident was potentially preventable by changing how potassium chloride is stored and packaged. The NPSA then issued a safety alert regarding potassium chloride solutions which included specific recommendations regarding the storage, packaging, and handling of potassium chloride solutions and the rates of accidental potassium chloride administration fell.

A key premise of the NPSA's work is that incident reporting, the NRLS, and alerts can improve patient safety by allowing learning from incidents and near misses[9]. This model of incident reporting and learning has been successful in other high risk industries such as aviation where with time the number of incidents has increased but the total number of incidents causing death or serious

harm has fallen.

Over one million incidents are reported to the NRLS per year, many more than would be humanly possible for the staff to review. Therefore routine analysis, and learning, is limited in the main to incidents which are reported to have caused serious harm or death. A consequence of this is that there are potentially additional, as yet undiscovered, patterns and learning in the large number of incidents that are not analysed centrally.

1.4 Data mining

1.4.1 What is data mining?

Data Mining (DM) is the analysis of (often large) observational data sets to find unexpected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.[10] In order to achieve understandable and useful insights, DM requires input from experts with domain knowledge.

Historically, data mining has been conceived of as a step in the Knowledge Discovery in Databases (KDD) process.[11] That is the discovery of knowledge from a database involving the selection of data, preprocessing, transforming, mining to extract patterns and relationships, and then interpreting and assessing the discovered structure.

More recently, the term DM has been used to refer to the KDD process itself and I use the term this way. Incidentally, use of the term Data Science has recently emerged to refer to this process where the data being manipulated is large, gigabytes or more at the time of writing (Big Data).

DM draws heavily on statistical techniques from machine learning which are about building programs or models with tunable parameters that adjust automatically and improve their behaviour by adapting to previously seen data.[12]

DM is non-linear, iterative, and intimately related to the selection of data, data preprocessing, and data transformation. Therefore, in this thesis I use DM to refer to the whole process, from obtaining data to arriving at valuable interpretations of it. I also use the term to refer to both descriptive and predictive

analytic methods.

1.4.2 Techniques used in data mining

DM may be broken up into three stages that mirror the extraction, transformation, and load (ETL) pattern (see section 3.1 also):

1. Data acquisition (extraction)
2. Data processing and analysis (transformation)
3. Evaluation, interpretation, and tuning (loading)

Data cleaning and audit

The first step of data processing in DM is typically data cleaning and audit. This includes removing duplicates and reconciling spelling variants and acronyms as well as auditing the data. Typical audit tasks would include identifying the types of data field present and how often they are completed within the data sample.

Exploring data

Analysis of a large textual data sample may begin with inpection of a manageable subsample or a search of the data for topics of interest which might be present.

Techniques are discussed in more depth in chapter 3. At root, quantitative analysis techniques used in data mining are statistical in nature. They may be used either to describe, and gain new insight into, existing data, or, to build models from existing data to make predictions for new data. The advent of widely accessible computing power and data has led to a significant growth in the number of people carrying out these types of data analysis. In turn, the number of tools available has increased proportionally and applications outside of traditional business analysis and science have become widespread.

The two specific machine learning techniques I will consider in more depth in this thesis are:

- Clustering
- Classification

Clustering

Clustering is an unsupervised machine learning technique in which similar samples in a dataset are automatically placed together into distinct groups, or clusters, by an algorithm.

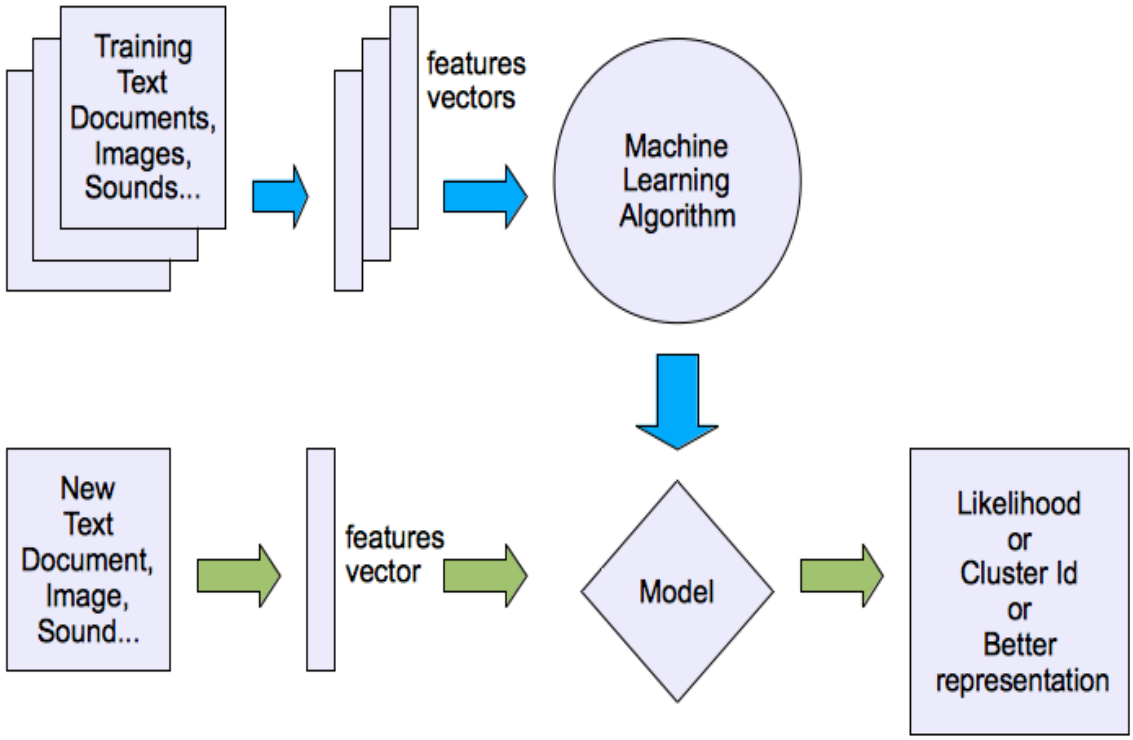

Figure 1.1: Unsupervised learning overview. An unlabelled data set is used to build a model that best summarizes regularities found in the data. The two main techniques in unsupervised machine learning are dimensionality reduction and clustering.[1]

Some common applications of clustering algorithm include[1]:

- Building customer profiles for market analysis
- Grouping related web news (e.g. Google News) and websearch results
- Grouping related stock quotes for investment portfolio management
- Can be used as a preprocessing step for recommender systems
- Can be used to build a code book of prototype samples for unsupervised feature extraction for supervised learning algorithms

The other main unsupervised learning technique, which is not considered in depth here, is dimensionality reduction. In dimensionality reduction the task is to derive a new set of artificial features that is smaller than the original feature set while retaining most of the variance of the original data. This can be useful for allowing visualisation of high dimensional datasets and as a preprocessing step in computationally intensive supervised learning methods.

Classification

In classification a supervised learning algorithm is used to build a predictive model (a classifier) from a labelled dataset. The classifier operates on new unlabelled data to predict the label. For example, a classifier may be built to label new emails as spam, normal or priority mail using a collection of old emails which have been manually labelled by a human. Common applications of classifiers include:

Task	**Predicted outcomes (labels)**
E-mail classification	Spam, normal, priority mail
Language identification in text documents	en, es, de, fr, ja, zh, ar, ru...
News articles categorization	Business, technology, sport...
Sentiment analysis in customer feedback	Negative, neutral, positive
Face verification in pictures	Same / different person
Speaker verification in voice recording	Same / different person

Table 1.1: Example applications of classifiers.[1]

Examples in the public domain

There are many freely available open source software implementations of solutions to machine learning problems available online. Many machine learning libraries provide code examples to perform 'standard' or reference classification tasks in machine learning. For example, predicting the newsgroup a message belongs to

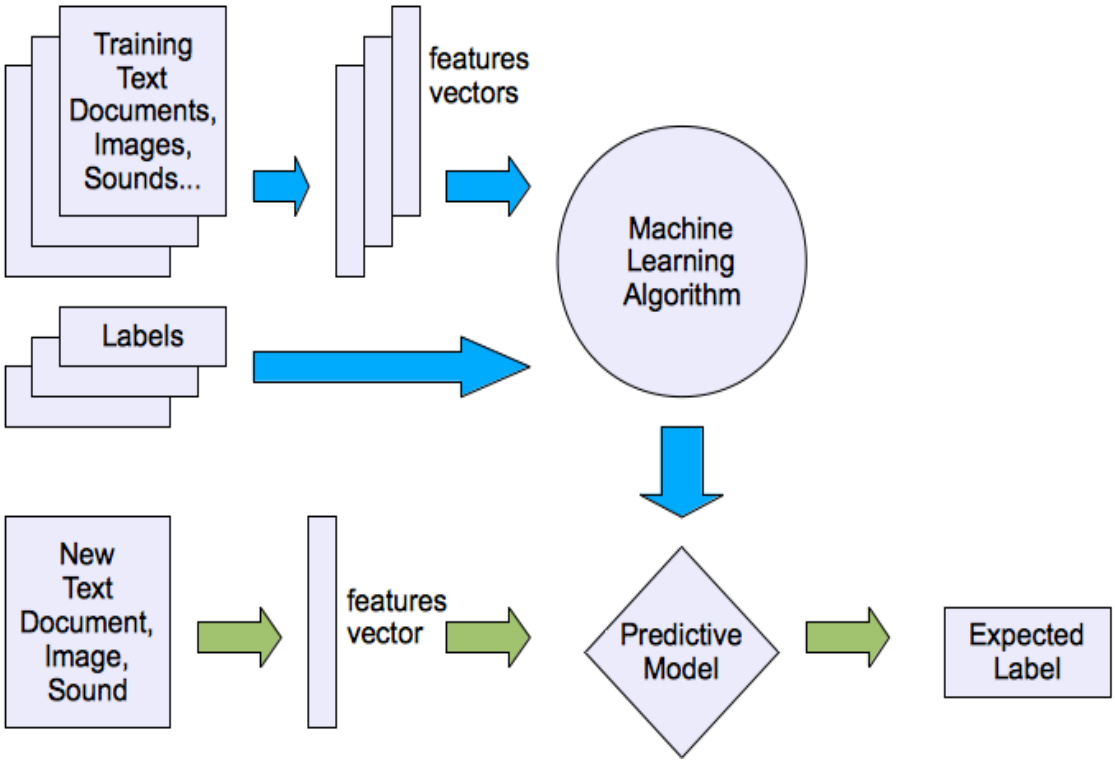

Figure 1.2: Supervised learning overview. A labelled data set is used to train a predictive model that can then predict labels for new unlabelled data. When the labels are categorical variables the task is called classification. When the labels are continuous variables the task is called regression.[1]

	Actual class (observation)	
Predicted class (expectation)	true positive (**tp**)	false positive (**fp**)
	false negative (**fn**)	true negative (**tn**)

Table 1.2: To evaluate a binary classifier label predictions made by the classifier (positive or negative) are compared with the known label (true or false)

from the message body using the reference 20 Newsgroups data set.[13]

Other code examples are readily available through popular question-and-answer sites such as StackExchange (www.stackexchange.com) and the machine learning competition platform Kaggle (www.kaggle.com).

Evaluation of machine learning efforts

Unsupervised machine learning techniques such as clustering are evaluated in practice by the usefulness of the insights generated to the organization or individual who invests in it.[14][15]

Supervised machine learning techniques permit the application of quantified performance measures such as Accuracy, Precision, and Recall.

The accuracy of a binary classifier is the number of correctly identified cases (positives which are true, negatives which are false) divided by the total number of cases. The precision of a classifier is calculated by dividing the number of true positives identified (by the classifier) by the total number of identified positives (true positive + false positives). It is also called the positive predictive value. The recall of a classifier is number of true positives identified divided by the total number of true positives.

In other words:

- $Accuracy = \frac{tp+tn}{tp+fn+fp+fn}$
- $Precision = \frac{tp}{tp+fp}$
- $Recall = \frac{tp}{tp+fn}$

A further commonly used performance measure is the F-score. This is the harmonic mean of precision and recall:

- $F\text{-}score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

1.5 Patient safety incidents relating to clinical information systems

In order to make this thesis more manageable I have decided to limit patient safety incidents considered to the subset of those incidents caused by computer problems. I have chosen to select incidents relating to computer problems for two main reasons:

1. I am a doctor in the NHS and have personal experience of safety issues relating to NHS clinical information systems
2. As health care becomes increasingly digitized safety issues relating to clinical information systems are becoming more important

Safety in clinical information systems has been identified as a major issue both nationally and internationally during the incident sampling period (2002-2012). In 2004, a high level safety review, carried out by the NPSA, found that the National Programme for Information Technology (NPfIT) were not addressing safety in a structured pro-active manner.[16] More recently, the Institute of Medicine published a report titled "Health IT and Patient Safety: Building Safer Systems for Better Care"[17] which recommended that Health IT related adverse events be routinely recorded and analysed.

1.6 What would data mining helping look like?

I will consider how data mining might help the NRLS in general and in the specific case of patient safety incidents relating to clinical information systems.

1.6.1 Data mining helping in general

Data mining may help in one of two ways.

1. It may support current existing analysis practices to be carried out more efficiently by the NRLS, allowing more incidents to be analysed.

2. It may improve effectiveness by providing novel insights into the data which are only practicably achievable through data mining.

Goals in patient safety reporting and learning systems, such as the NRLS include:[18]

- Identifying new and previously unexpected hazards
- Discovering trends
- Uncovering common contributing factors
- Prioritizing areas for remedial efforts
- Informing strategies to decrease adverse events and patient harm

Typically, the first analysis task carried out in traditional incident analysis, in order to achieve the goals above, is classification according to a patient safety incident taxonomy.[18 19]

However, analysis does not, or should not, stop here because while classification of incidents into a predefined taxonomy does involve a type of analysis it greatly constrains the insights that can be obtained from the data when used alone.[20]

In this thesis I will attempt to address both points practically through the consideration of the specific case of patient safety incidents relating to clinical information systems, by carrying out analysis, and theoretically, by synthesising what others have done.

1.6.2 Data mining helping in the specific case of patient safety incidents relating to clinical information systems

Considering patient safety incidents relating to clinical information systems and the example data mining techniques of classification and clustering, data mining might help as follows:

1. Data cleaning and audit methods might help the organisation to identify and monitor data quality issues

2. Tools to explore the data might help to identify novel themes and support analysis of incidents

3. Cluster analysis of free text incident descriptions may support human analysis of incidents

4. Classifying severity of harm arising from a patient safety incident using free text incident descriptions to reduce misclassification and save time

The current classification of incidents which designates the incidents considered as relating to clinical information systems does not offer further granularity beyond this. Cluster analysis may reveal more detail about the types of thing these incidents are to do with, possibly suggesting useful subcategories. This would be helpful for understanding these incidents and coordinating remedial efforts.

It is known that patient safety incidents are occasionally incorrectly classified according to severity of harm by healthcare professionals reporting them. This is particularly unfortunate because the reporters severity of harm classification is used to prioritize which incidents patient safety expert analysts review centrally which means this misclassification may go unrecognised.

In a 900-bed NHS Acute Trust Hospital Williams et al asked 40 healthcare professionals (10 doctors, 10 nurses, 10 pharmacists and 10 pharmacy technicians) to complete a self-administered questionnaire on their perception of the severity of harm in provided example medication incidents and found wide variation between and within professional groups.[21] Ong et al found that for a sample of 113 previously unreviewed incidents those graded as being the most serious incidents, by expert analysis, were misclassified by the reporter in 34.4% of cases and not classified in 3.1% of cases.[4]

If it were possible to build a classifier that accurately predicted severity of harm for computer incidents then it could flag incidents where reporter and computer classification disagreed for review which could reduce the rate of misclassification. This would enhance existing analyst practice and boost NRLS overall efficiency.

Chapter 2

Review of the literature

2.0.3 Patient safety incidents in Health IT

Evidence of harm

The potential for health IT to support high quality and safe care is well recognized.[22][23] So, however, is the potential for health IT to introduce new types of error and have unintended consequences.[24][25][26]

Authors have reported health IT deployment causing harm, and even death.[27] The poor design and usability of some health IT systems has been observed to impact negatively on patient safety, and there have been calls to monitor adverse events due to Electronic Medical Record (EMR) systems.[28][29][30]

Many authors have used qualitative methods to investigate harm occurring to patients that is associated with the use of health IT. Ash et al[25] administered a structured telephone survey of 176 US hospitals that had computerized physician order entry (CPOE) systems and found that adverse unintended consequences with safety implications were commonly reported by informants. Interestingly, these safety issues seemed to persist throughout the lifespan of the CPOE systems considered, suggesting that in current practice these safety issues are intractable for some reason or another.

Han et al[27] unexpectedly found a statistically significant increase in paediatric mortality after introduction of CPOE at an academic tertiary-care level children's hospital. CPOE was introduced at month 13 of an 18 month study period. 1942 children referred and admitted for specialist care during the study period formed

the sample. After adjustment for relevant mortality co-variables multivariate analysis revealed that CPOE was independently associated with an increased odds of mortality (odds ratio: 3.28; 95% confidence interval: (1.94 - 5.55). The authors speculate that this unexpected mortality may be due to impaired communication, delays in the administration of time sensitive medications, and reduced 'face time' with patients resulting from introduction of the CPOE system. However, the mortality-CPOE association discovered can not be used to infer causality, the single research setting considered may not generalize, and the study period after CPOE introduction was short. It is possible that if excess mortality were due to the CPOE introduction this might be a transient 'bedding in' phenomenon.

Scott et al[30] demonstrated a relationship between health IT user interface design and safety outcomes is a laboratory setting using a within participant design randomized controlled trial and a cohort of 24 junior doctors. Participants performed simulated prescribing tasks with a prototype electronic prescribing system and received modal safety alerts, non-modal safety alerts, or no safety alerts. The primary outcome was prescribing error and error rates varied significantly between experimental arms with modal alerts preventing the most errors. A major limitation of the study is that it occurred in an artificial setting and was limited to simulated clinical scenarios. That this was necessary because it is not currently feasible to carry out such studies "in the field" is interesting. It seems highly likely that some software designs will be associated with more or less unintended harm occurring to patients but in routine clinical practice it is not possible to test this or to directly improve the design of the software used.

Response to date

To date, safety has remained a relatively neglected issue in health IT, it was largely absent as a consideration from the early stages of United Kingdom's national health IT programme[16] and did not feature in the recent US health IT stimulus measures[31]. It has been observed that current market forces are not adequately addressing the potential risks associated with use of health IT.[17]

Barriers to improving patient safety in health IT include user disengagement due to a lack of vendor responsiveness,[32] contractual barriers, such as

non-disclosure and confidentiality clauses, and the absence of a public central repository, or linkages among localised repositories to collect, analyse, and act on safety incidents relating to health IT.[17]

2.0.4 What techniques are used to analyse patient safety incidents?

Traditional methods

Historically, case note review has been the principle method of detecting, and analysing, patient safety events. The seminal paper in this area was a review of 30,195 randomly selected hospital records from 51 New York hospitals which identified 1133 patients (3.7 percent) with disabling injuries caused by medical treatment. The authors classified, and subclassifed, incidents by type into five main categories using a standardized adverse event form. The categories were:[33]

1. Performance
2. Prevention
3. Diagnostic
4. Drug treatment
5. System

Internationally, an initial first step of human domain-expert classification of incidents according to a taxonomy is the dominant approach adopted in patient safety incident reporting and learning systems.[18] In the literature this classification then forms the basis of further within category analysis such as qualitative analysis of free text description for a particular category of patient safety incidents to produce a finer taxonomy for classification.[34] Alternatively, initial classification is augmented with targeted free text searches for key words to identify reports of interest. These reports are then categorized and subcategorized by theme. Examples of this approach include Panesar et al's analysis of the free text of all-cause mortality in trauma and orthopaedic surgery[35] and Arnot and Smith's analysis of incidents involving neuromuscular blockade.[36]

Data mining techniques in patient safety

Classifiers have been widely applied to patient safety in a research setting, with mixed results, and are not yet widely used in routine clinical practice.[23] I could not find an example of clustering being used in the patient safety literature and found only one instance of the related unsupervised machine learning technique, dimensionality reduction, being used.[37]

Ong et al used binary classification of incident category to identify incidents as to do with handover or not, and to do with patient identification or not, using Support Vector Machine and Naive Bayes classifiers. They trained the classifiers using the free text of 600 patient safety incident reports that had been labelled as do with handover or not (300 true, 300 false) and tested on an independent test set containing 372 incident reports (248 true, 124 false). Impressive results for this task were achieved using an SVM: accuracy = 97.98%, precision = 0.98, recall = 0.98, F-measure = 0.96. However, the validity of this is clearly very limited by the highly selective nature of the data used for testing and this makes the task appear artificial. It is not clear that identifying incidents to do with handover from a sample of incidents with a true:false ratio of 2:1 for being to do with handover or not is a convincing demonstration of the potential utility of automated incident classification.[42]

The identification of new types of adverse drug events has been a hot area of research, possibly due to the large financial stakes in this area. A recent review of studies of the automated detection of adverse drug events in the literature found that data mining techniques can be a useful tool for detecting novel adverse events in large datasets.[38]

Bentham el al. applied unsupervised and supervised Dimensionality reduction methods on a Vector Space Model with weighting followed by Anomaly detection methods to cluster incidents reported to the NPSA into novel meaningful groups. However the usefulness of this was questionable since they failed to find 'actionable' results. That is they did not identify any new types of incident with common causal factors that were amenable to prevention.[37]

2.0.5 Previous analysis of patient safety incidents due to computer problems

Outside of the health setting 'computer problems' due to failure of software are often reported and tracked publicly in a very detailed fashion using specialised software 'bug' reporting and tracking systems. A software bug is an error, flaw, failure, or fault in a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended ways[2].

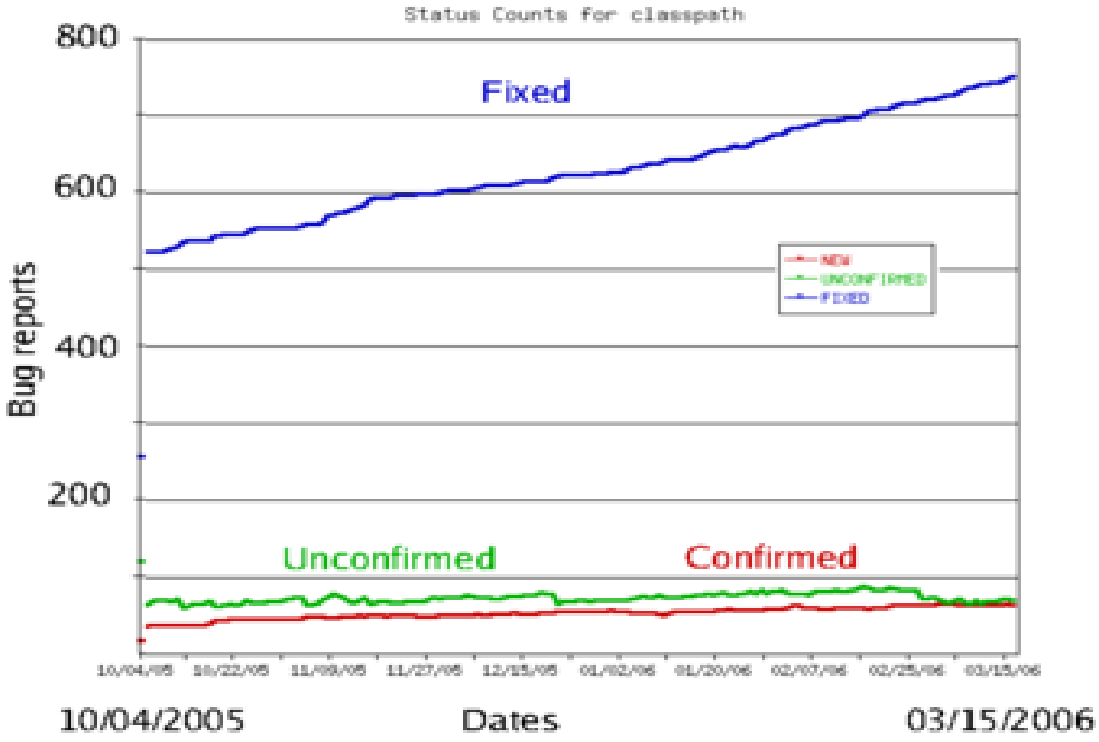

Figure 2.1: The typical bug history (GNU Classpath project data). A new bug submitted by the user is unconfirmed. Once it has been reproduced by a developer, it is a confirmed bug. The confirmed bugs are later fixed. Bugs belonging to other categories (unreproducible, will not be fixed, etc.) are usually in the minority.[2]

In the health setting public software bug tracking is uncommon but the subset of computer problems which are perceived to impact on patient safety are captured in patient safety reporting and this has been analysed by Magrabi et al.

Magrabi et al analysed 111 patient safety incidents related to computer use

across one Australian state to identify 'natural categories' for classification.[39] They identified 32 types of computer use problem which they grouped into four main categories:

1. Information input (31%)
2. Transfer (20%)
3. Output (20%)
4. General technical (24%)

Over 50% of problems were machine related and 45% were attributed to human–computer interaction. They found that delays in initiating and completing clinical tasks and the need to redo work were a common result of computer use problems.[39]

A follow up study of 678 incident reports related to computer use obtained from the US Manufacturer and User Facility Device Experience (MAUDE) database resulted in the addition of four new categories relating to software use:[40]

1. Software functionality
2. Software system configuration
3. Software interface with devices
4. Network configuration

Under-reporting is an issue known to affect all patient safety reporting and learning systems[32,41] and this limits the usefulness of these studies which relied on users reporting incidents. It is possible to capture some types of computer incident, such as unscheduled downtime, in an automated fashion but this was not undertaken in either study.

2.0.6 Data mining evaluation criteria

As discussed previously, evaluation methods in machine learning include objective assessments such as the calculation of a classifier's Accuracy, Precision, Recall, and F-score.

In practice, data mining is evaluated by the usefulness of the insights generated to the organization or individual who invests in it. In the world of business 'usefulness' here often means return on investment, and may be quantified objectively on a balance sheet. Within the context of the NPSA patient safety data mining's 'usefulness' refers to the extent to which it is perceived as a helpful tool to the expert patient safety analyst.[10 37]

As such unless the outcomes of the data mining process are objectively compared with existing processes, or the data mining process is tied to an intervention which affects a measurable outcome, then objective demonstration of the value of data mining is hard and evaluation is limited to the subjective assessment by domain experts of whether or not value has been added.[42 4 37]

Part II

Method

"You can know the name of a bird in all the languages of the world, but when you're finished, you'll know absolutely nothing whatever about the bird... So let's look at the bird and see what it's doing – that's what counts. I learned very early the difference between knowing the name of something and knowing something."

Richard Feynman (1918 - 1988)

Chapter 3

Data analysis

3.1 Overview

The overall process of data analysis is not linear in DM and it requires experimentation to achieve useful insights. However, the process can be usefully considered as comprising three main steps: data extraction, transformation, and loading for analysis.

In this case the subset of patient safety reports relating to computer problems extracted from the NRLS database using a SAS query were cleaned and quality checked before being explored, transformed, and loaded for further analysis, see figure 3.1.

After extraction data were cleaned and selected fields (incident free text description and severity of incident) were converted to csv and xml data formats for subsequent analysis using Apache Solr, Scikit-learn and NLTK (csv), and Carrot2 (xml). Preprocessing techniques including stemming, tokenization, tagging, and filtering were applied.

Data were audited using Python Brewery and validated by searching for known patterns of interest using grep, Google-refine, and Apache Solr. Data were loaded into NLTK for lexical analysis, Apache Solr to search for strings of interest for validation, the Carrot2 platform for cluster analysis, and Scikit-learn in order to construct a classifier.

Lingo (a clustering algorithm based on the Singular value decomposition) was employed within the Carrot2 platform to perform cluster analysis on selected data

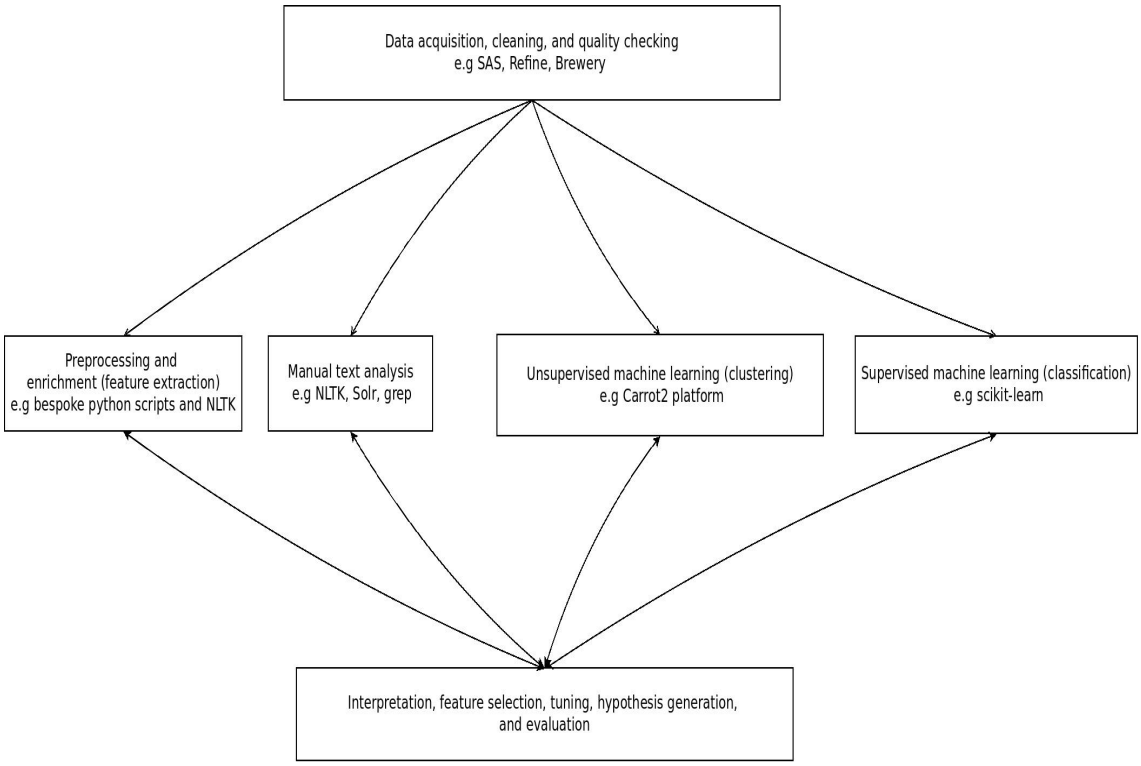

Figure 3.1: Overview of the data mining process with examples of tools used. In addition external corpora may be used to enrich the analysis, and a visualisation step is usually performed.

for the example problems considered. Naive Bayes (NB) and stochastic gradient descent (SGD) incident severity classifiers were tuned using a grid search strategy and evaluated using cross validation.

3.2 Data extraction

3.2.1 Ethics and data storage statement

This work is an exercise in service development and as such formal ethical approval was not required. Data used have had patient identifiable information removed but are theoretically at risk of deanonymization and so were encrypted and stored in a physically secure location.

3.2.2 The "computer problem" extract

SAS Query of NRLS database

All incidents reported as occurring between 1^{st} January 2002 and 1^{st} March 2012 and classified as "Infrastructure (including staffing, facilities, environment)" (incident category level 1) and "IT / telecommunications failure / overload" (incident category level 2), were extracted from the NRLS database using a SAS® Enterprise Guide 4.3 add-on for Microsoft Office Excel® on the 20^{th} of March 2012.

3.3 Data cleaning and audit

Ensuring data quality is an essential prerequisite to analysis. Data cleaning and audit methods might help the organisation to identify and monitor data quality issues (see section 1.6.2).

The source data for "computer problem" extract are incident reports which are completed by individual health care professionals and submitted electronically. Duplicate report submission and submission of reports with missing data is possible and it was therefore important to deduplicate the data and establish its quality. Google-refine[43] (now Open-refine) is a power tool for rapidly cleaning and transforming data and keeping a log of changes made. I chose to use it

because it has an easy to use interface and I'm familiar with it. Python Brewery (now Python Bubbles) is a framework for data processing and data quality measurement which makes it easy to rapidly perform tasks such as calculating percentage field completeness for fields in a data table. I chose to use it because I shared a room with the author at EuroPython 2012, he explained the project to me and it seemed to fit with the task.

3.3.1 Deduplication and reconciliation

Google-refine[43] was used to remove duplicates by faceting on the incident description free text field (IN07). Reconciliation was not carried out for time reasons although Google-refine[43] does support this. This would have been helpful for correcting spellings and handling acronyms and synonyms.

3.3.2 Python Brewery

Google-refine[43] was used to establish median field length and number of words per field using the Google Refine Expression Language (GREL). The Python Brewery[44] module was used to calculate field completeness for the data set (see section B.1.11 for source code and 4.1.1 for results).

3.4 Exploring the data

Tools to explore the data might help to identify novel themes and support analysis of incidents (see section 1.6.2).

Grep, Apache Solr, and NLTK, were used to rapidly search for the presence of words suspected to be key words on the basis of the literature review and prior experience (see section 2.0.5 for background and 4.2 for results). Returned incident descriptions were then briefly reviewed in order to get a feel for the types of descriptions contained in the dataset. Finally the Carrot2 platform was used to perform unsupervised clustering of the data.

3.4.1 Grep

GNU Grep version 2.10, a command-line utility that searches a file for, and returns lines containing, a user specified pattern, was used to search for the presence of incident reports containing words suspected to be key words in computer problems such as “crashed”, “locked”, “froze” etc. The tool was chosen because the author is familiar with it and it fit well with the task.

3.4.2 Apache Solr

Apache Solr 3.6.0[45], an open source search platform, was configured to run locally and used to search incidents for key words (described in section 2.0.5) around themes identified from experience including:

- Software bugs
- Poor reliability
- Ongoing problems

The platform was chosen because of its powerful full text search capabilities as well as its standard open interface which permitted results to be returned in an xml format that could be fed into the Carrot2 clustering tool used in later analysis.

3.4.3 NLTK

NLTK (version 2.0b9), is a computational linguistic tool for Python. The tool was chosen because the author has some prior knowledge of Python, because it is very well documented, and because it has a strong and supportive online community. NLTK was used to characterise and explore the text corpus formed by the free text incident descriptions relating to computer problems and to investigate interesting language snippets found.

Lexical analysis

Specifically, the name of a text file is passed as an argument to a Python script that reads a string from the text file and divides, or tokenizes, it into a list

of substrings. Exactly how depends on the tokenizer used. I used wordpunct tokenize, the standard tokenizer for words and punctuation provided in the NLTK library. As an example, tokenizing the string "Hello, world." with wordpunct tokenize would result in the following list of substrings:

```
['Hello', ',', 'world', '.']
```

After tokenization the words were all converted to lower case to create a vocabulary and calculate lexical diversity (the number of unique words / the number of words). A set of collocations for the text is then built. A collocation is a sequence of words that occur together unusually often. Thus, 'red wine' is a collocation, whereas 'the wine' is not. Collocations are resistant to substitution, for example, "maroon wine" would sound very odd. They tend to be specific to a given text and so can convey useful information.

A cumulative frequency distribution plot of the 50 most common words and a text dispersion plot of words of interest, such as those known to be associated with computer failure like "crash", was made for interest. A text dispersion plot is a plot of the frequency of occurrences of a word over time (for commented source code see section B.1.1, for code output see section 4.2.3).

Named entity recognition

Named entity recognition is used to identify persons, companies, and locations, and discover relationships between them. This was carried out for the text in the following way. Using NLTK, free text descriptions of incident reports were tokenized to sentences and then words, tagged with a POS (part of speech) tagger, and 'chunked' into noun phrases (see figure 3.2). 'Chunking' means extracting a list of 'chunks', in this case noun phrases, from the text (for commented source code see section B.1.2).

Investigation of language snippets of interest

Where snippets of interest, for example "Computers unable", which was identified as a cluster using the Carrot2 clustering platform, were found (see figure 4.10) this was investigated further by review of incidents containing the phrase, identified

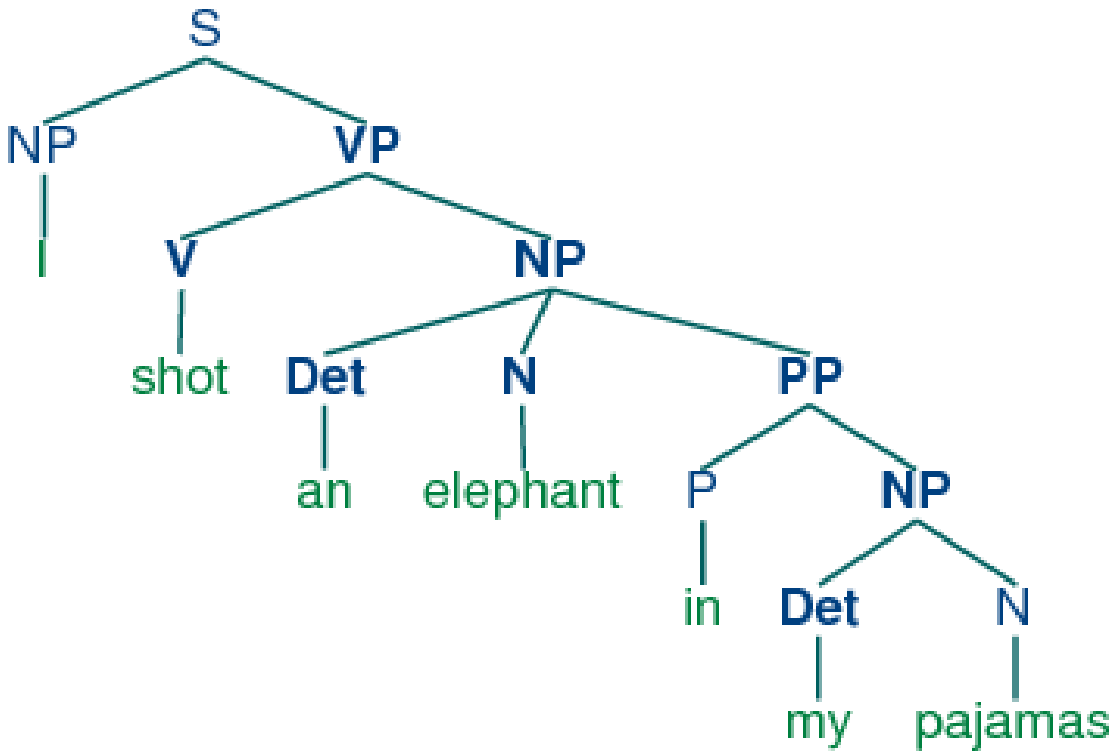

Figure 3.2: Example of part of speech tagging in NLTK for an ambiguous phrase. NNP = proper noun, V = verb, Det = determiner, N = noun, P = preposition, VP = verb phrase, NP = noun phrase.

using grep and/or the Solr search engine and the use of NLTK to, for example, create a frequency distribution plot of the top 50 words which follow "unable to" in the text (for commented source code see section B.1.3 and for results see section 4.2.3).

3.5 Lingo

Clustering is an unsupervised machine learning technique (see section 1.4.2) that might support human analysis of large numbers of patient safety incidents (see section 1.6.2)

The lingo algorithm was chosen because it seems to map well to the domain problem considered. It is typically used for search engine result clustering so for example a search for apache would return semantically usefully labelled clusters of results e.g "Apache Indians, "Apache helicopters", "Apache Software Foundation", "Apache Licence", "Apache Wars" etc. The group of incidents considered in this dissertation are in a sense "search results" to be summarised since they were obtained by searching the NPSA database for all incidents classified as being to with computer problems. The algorithm was also applied to search results for searches within the sample, produced using the Solr search engine, for strings of interest.

The algorithm is based on Singular Value Decomposition(SVD) and uses a Vector Space Model(VSM) with each document d being represented as a *document vector* $[\omega_{t_0}, \omega_{t_1}, \ldots \omega_{t_\Omega}]$ where $t_0, t_1, \ldots t_\Omega$ is a set of words and ω_{t_i} expresses the weight (importance) of term t_i to document d and each term is assumed to be an independent dimension. Collections of *document vectors* form a *term-document matrix*, with the value of each element depending on the strength of association with the respective document. Singular value decomposition, a way of factorizing the matrix, is then used to find clusters.

There are three phases to clustering with the lingo algorithm:

1. Cluster discovery (unsupervised phrase extraction)

2. Candidate label discovery (cluster-label-induction)

3. Cluster-label matching (cluster-content allocation)

The phrase extraction phase discover phrases and single terms that could potentially explain the verbal meanings of SVD-derived abstract concepts using a modified semantic hierarchical clustering (SHOC) algorithm.

The cluster-label-induction phase identifies the abstract concepts that best describe the input document collection and uses frequent phrases to construct a human-readable representation of these concepts (the cluster labels).

Cluster labels are required to:

- appear in the input snippet at least a specified number of times
- not cross sentence boundaries
- be as complete as possible
- neither begin nor end with a stop word

Finally, in the cluster-content allocation phase input documents are matched against cluster labels, if the similarity exceeds a predefined threshold then the document is allocated to the cluster.

To use the Carrot2 platform it was necessary to convert the .csv file into an appropriately formatted .xml file. This was achieved using a python script that Ross Jones helped me to write (for commented source code see section B.1.4 and for results see section 4.3).

3.6 Scikit-learn

Classification is a supervised machine learning technique (see section 1.4.2) that might help to ensure the accuracy of the classification of the severity of patient safety incidents (see section 1.6.2).

Scikit-learn is is an open source machine learning library for the Python programming language. It supports supervised text classification tasks using Naive Bayes (NB) and Stochastic Gradient Descent (SGD) algorithms. I used it because Python is my preferred programming language and because Scikit-learn is a very well documented project with a large and very supportive community.

3.6.1 Classification task

The example classification task chosen was to classify the degree of harm of a clinical incident based on its description.

The training data set was composed of incident description and degree of harm fields. Each incident has an incident description and a human-assigned degree of harm category.

Categories used are:

- Death
- Severe
- Moderate
- Low
- No Harm

3.6.2 Adapting the 20 Newsgroups data set code examples

The 20 Newsgroups data set (see section 1.4.2 also) is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. It is a very popular data set for testing text classification applications and there existed many examples for scikit-learn. The standard classification task considered is to predict the newsgroup that the document belongs to on the basis of the document contents. This is roughly analogous to predicting severity based on document contents so I decided to adapt these examples for my purposes by converting my patient safety data set into a similar format to the 20 Newsgroups data set and editing the code as necessary.

Splitting the patient safety dataset into training and testing subsets

To avoid the problem of overfitting it was necessary to split the original data set into training and testing subsets. This is achieved using a Python script (incidentsplit.py) and with the csv and numpy libraries to read the original csv and split it into a training dataset (2/3) and a testing dataset (1/3) in a randomised fashion (for commented source code see section B.1.5).

Extracting incidents and placing them in a folder structure

Messages in the 20 newsgroups data set are arranged in 20 folders which are labelled according to the name of the newsgroup the message was posted to. To prepare my dataset for analysis it was necessary to extract incidents and place them into folders labelled according to the severity of the incident. This was achieved using a Python script (incidentextract.py) that reads a csv file and writes the free text incident description into a folder that matches its severity category (for commented source code see section B.1.6).

Selecting a classifier

Different classifiers were evaluated by adapting an existing 20 newsgroup scikit-learn Python classifier comparison script to use the computer incident data (for commented source code see section B.1.7). As part of this process the effect of tuning classifier parameters, weightings, balancing the data set, and simplifying the task to a binary (death OR severe) vs not(death OR severe) classification were examined. Two classifiers, MultinomialNB and SGDClassifier appeared to perform particularly well and were selected for further optimization. To experiment with balancing the training data set all deaths and severe incidents (N=51) were used as positives and matched with a random selection of 51 incidents from the pool of all incidents that weren't death or severe incidents (for commented source code see section B.1.8).

Classification

Training and test datasets are loaded into scikit-learn from a folder container which contains two subfolders 'Training' and 'Testing'. Within each subfolder there are five further subfolders 'Death', 'Severe', Moderate', 'Low', 'No Harm' containing the appropriate free text incidents as individual files (for commented source code see section B.1.9).

Countvectorizer converts the collection of text documents into a matrix of token counts. The default setting of word tokenisation is used.

Term occurrences (X_counts) fails to take account of document size. Longer documents may have higher average count values than shorter ones even though

they talk about the same topic. To avoid this the number of occurrences in any document is divided by the total number of words in that document to give the Term Frequency (tf) using the function TfidfTransformer.

The MultinomialNB function is used to implement a multinomial Naive Bayes classifier which is suitable for classification tasks involving data with discrete features (i.e the word counts we have generated for text classification). SGDClassifier, uses a Stochastic Gradient Descent algorithm which is an efficient learning algorithm for large text data sets and supports sample weighting. SGDClassifier was also chosen because it has performed well in recent public online text classification challenges hosted by the machine learning challenge website kaggle.com (see section 1.4.2 also).

Parameters for MultinomialNB and SGDClassifier were optimized using a the scikit learn grid search and pipe libraries which allow automated parameter tuning and evaluation to optimise algorithm performance (for commented source code see section B.1.10 and for results see section 4.4).

Classifier performance was measured by cross validation. The dataset is split into training and testing partitions in a ratio of 2:1. The model is trained on 2/3rds of the labelled data, the training set, and tested against the remaining 1/3rd of the data, the testing set. Partitions are created by random sampling of the dataset. In this way training the classifier on the testing dataset and the resultant problem of overfitting are avoided.

Part III

Results

"Whereof one cannot speak, thereof one must be silent."

Ludwig Wittgenstein (1889 – 1951)

Chapter 4

Results

Of over seven million incident reports, 7273 were classified by NPSA staff as belonging to "Infrastructure (including staffing, facilities, environment)" (incident category level 1) and "IT / telecommunications failure / overload" (incident category level 2) categories. These were selected to form the "computer problem" extract (since it was felt that computer problems would fall under this classification category).

4.1 Descriptive statistics

4.1.1 Data quality

The NRLS "computer problem" extract contains 7273 rows of data. Each row represents a clinical incident report. Each column represents one of 32 fields (See tables 4.3 and 4.4) which are extracted from the incident report form and cleaned to remove sensitive data by the NRLS. The dataset contained 159 duplicates (Table 4.1) which were removed before further analysis was carried out. The completeness of fields varied considerably and some fields were derived entirely from other fields, for example, the age range field is populated by the patient age field.

The median number of words in the free text incident description field (IN07) was 35 and the range spanned 1-730, this was a mandatory field (see section 3.3).

Is the description of the incident a duplicate?	
False	7114
True	159

Table 4.1: Incident reports in the computer problem dataset with identical incident descriptions (duplicates)

4.1.2 Sample breakdown

The NRLS incident reporting form has 5 categories of patient harm:

1. no harm
2. low harm
3. moderate harm
4. severe harm
5. death

The computer problem sample is similar to the totality of incident reports submitted to the NRLS in that no harm and low harm incidents are reported more frequently than more serious incidents such as severe harm and death. See tables 4.1.2, 4.1.2, 4.1.2, and 4.1.2 for details.

4.2 Data exploration

Exploring the data with Grep, Apache and Solr (see section 3.4) confirmed the prescence of two major themes suspected to be present, namely, poor systems reliability and problems not being fixed promptly.

4.2.1 Grep

Greps for terms the author thought might be interesting such as “crash”, “freeze”, “locked”, and “unable”, returned many records and gave a flavour of how healthcare professionals report computer problems.

Spelling variant	N
Anaes	1
Anaesetic	1
anaeshthetist	1
anaestetist	1
anaesth	1
Anaesthatised	1
Anaesthatist	1
anaesthesia	1
ANAESTHESIST	1
anaesthetic	39
Anaesthetic	11
ANAESTHETIC	3
anaesthetics	4
Anaesthetics	2
anaesthetied	1
anaesthetise	1
anaesthetised	5
anaesthetising	1
ANAESTHETIST	6
anaesthetist	54
Anaesthetist	21
anaesthetist4	1
anaesthetists	3
anaesthtic	1
anaethestised	1
anaethetised	1
anaethetist	3
Anaethetist	1
anaethetists	1

Table 4.2: There were 31 spelling variants of anaethetist and related terms in the computer problem data set. Acronyms were also commonplace. Without reconciliation spelling variants and acronyms can be detrimental to machine learning tasks such as clustering and classifying

Field name	**Field completeness(%)**
Unique Incident Id	100.0
Age At Time Of Incident Date	30.0
Incident Received By Npsa	100.0
Type Of Device	6.0
Patent Age At Time Of Incident	30.0
Date Record Exported To Nrls	100.0
Date Of Incident	99.9
Location (lvl1)	100.0
Location (lvl2)	99.7
Location (lvl3)	55.6
Location - Free Text	7.5
Incident Category - Lvl1	100.0
Incident Category - Lvl2	100.0
Incident Category - Free Text	0.0
Free Text Description Of What Happened	100.0

Table 4.3: Field names in the dataset and percentage completeness for the computer problem data set. Percentage field completeness is calculated by dividing the number of non-blank fields for a given field name by the number of incident reports, after removing duplicates, and multiplying by 100.

Field name	**Field completeness(%)**
Actions Preventing Re-occurrence	38.0
Apparent Causes	21.5
Med Process	1.6
Med Error Category	1.5
Approved Name (drug 1)	0.1
Proprietary Name (drug 1)	0.0
Patient Age Range	30.3
Patient Sex	55.5
Specialty - Lvl 1	100.0
Specialty - Lvl 2	63.3
Speciality - Free Text	18.8
Degree Of Harm (severity)	100.0
Paediatric Care	29.9
Source Of Notification	100.0
Care Setting Of Occurrence	100.0
Reason Exclusion	0.0

Table 4.4: Field names in the dataset and percentage completeness. Percentage field completeness is calculated by dividing the number of non-blank fields for a given field name by the number of incident reports, after removing duplicates, and multiplying by 100.

Harm	NPSA definition
No Harm	A situation where no harm occurred: either a 'prevented patient safety incident' or a 'no harm patient safety incident'.
Low	Any unexpected or unintended incident which required extra observation or minor treatment and caused minimal harm, to one or more persons.
Moderate	Any unexpected or unintended incident which resulted in further treatment, possible surgical intervention, cancelling of treatment or transfer to another area and which caused short-term harm, to one or more persons.
Severe	Any unexpected or unintended incident which caused permanent or long-term harm, to one or more persons.
Death	Any unexpected or unintended incident which caused the death of one or more persons.

Table 4.5: NPSA classification of degree of harm caused by incidents.

What was the degree of harm?	N
No Harm	5982
Low	792
Moderate	268
Severe	65
Death	7

Table 4.6: Degree of harm caused by incidents relating to computer problems.

Specialty	**N**
Medical specialties	1104
Diagnostic services	1039
Not applicable	995
Obstetrics and gynaecology	695
Surgical specialities	673
Other	600
Unknown	526
Primary care / Community	508
Accident and Emergency	415
Other specialities	137
Mental health	123
PTS (Patient Transport Service)	98
Learning disabilities	78
Anaesthesia Pain Management and Critical Care	62
Dentistry - General and Community	60
Children's Specialities	1

Table 4.7: Speciality involved in computer incidents, based on level one data.

Speciality	**N**
Acute / general hospital	5054
Ambulance service	121
Community and general dental service	34
Community nursing, medical and therapy service (incl. community hospital)	1614
General practice	92
Learning disabilities service	17
Mental health service	182

Table 4.8: Location of the computer incidents, based on care setting data.

Figure 4.1: Grep results for search on term “crash”.

Figure 4.2: Grep results for search on term “freeze”.

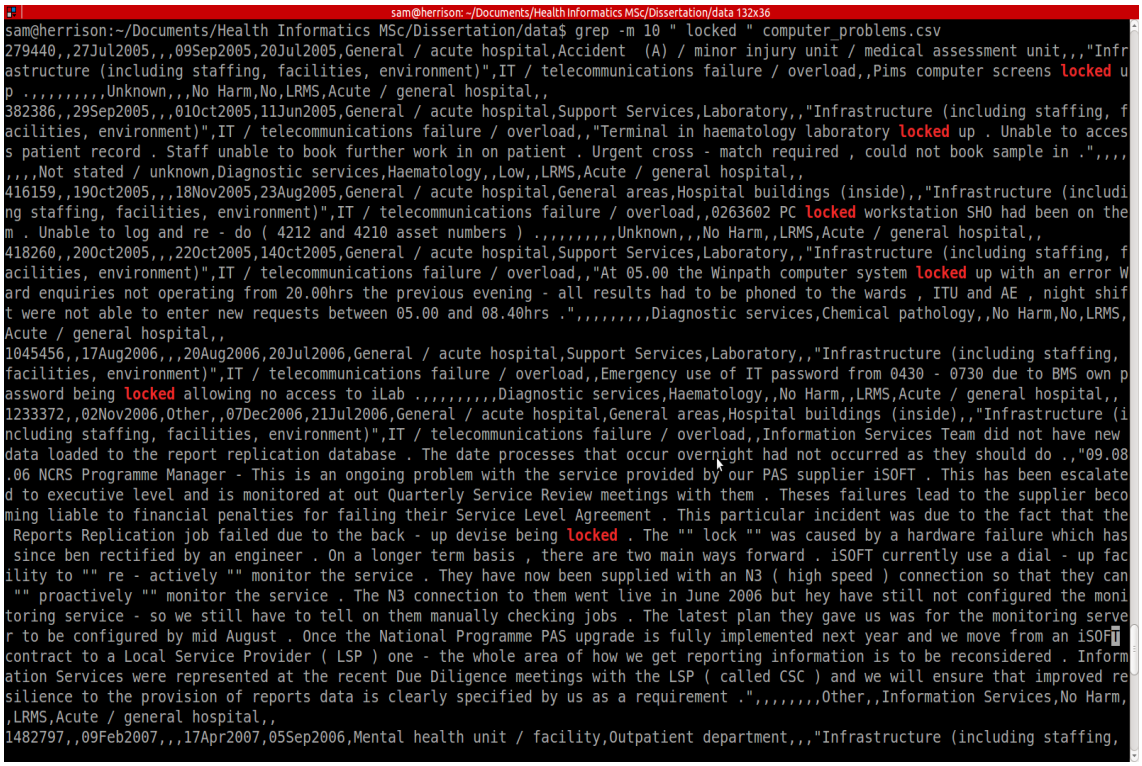

Figure 4.3: Grep results for search on term “locked”.

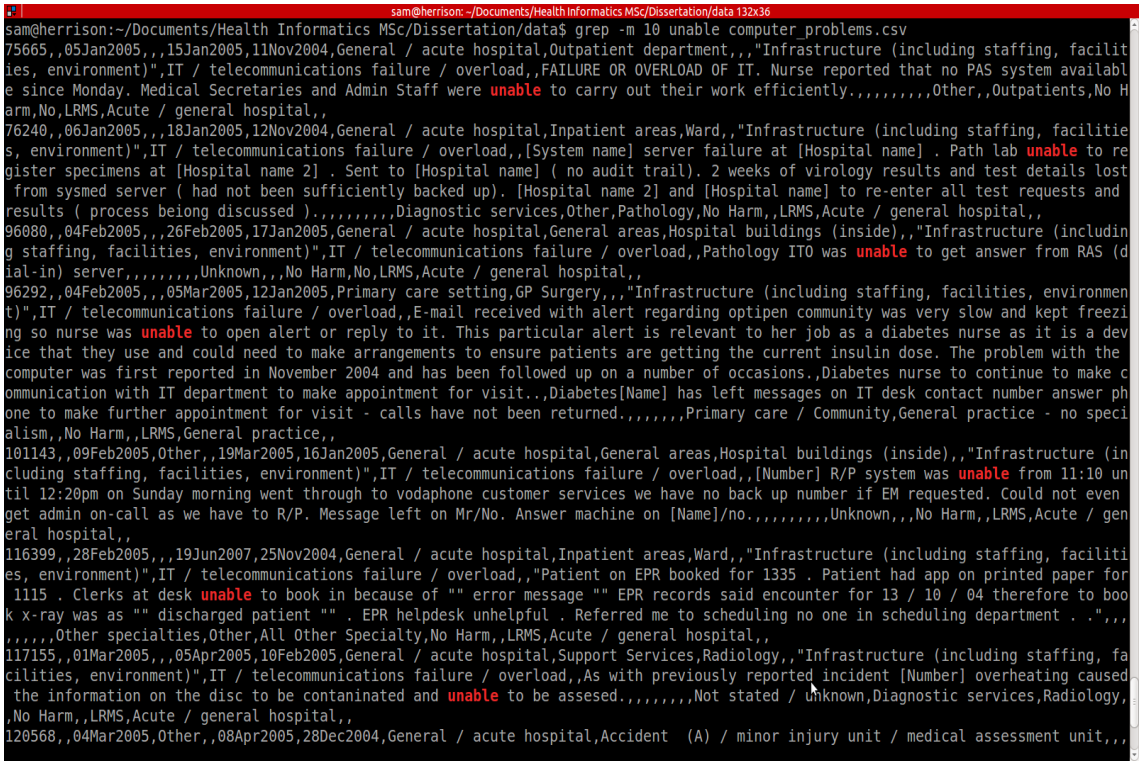

Figure 4.4: Grep results for search on term “unable”.

4.2.2 Solr

Poor system reliability Poor system reliability was identified as a theme and terms associated with poor reliability such as "crashed" were combined in a search using Solr.

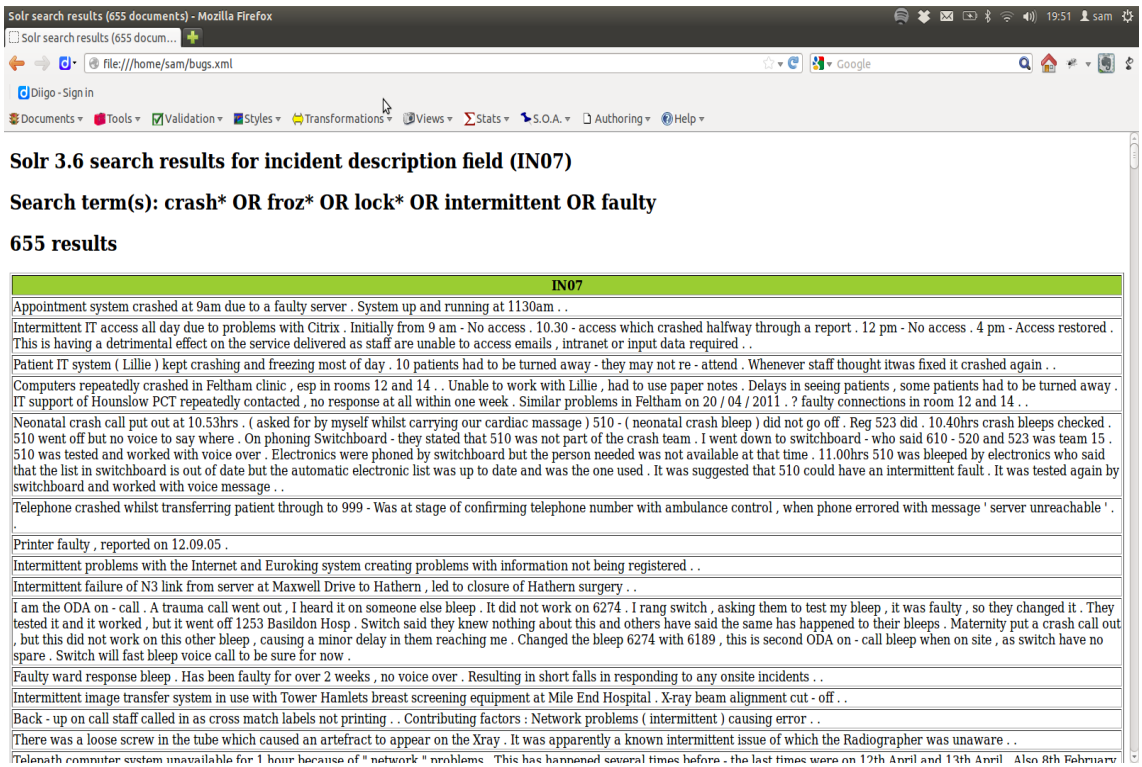

Figure 4.5: Search results for search of the computer problems data set for terms associated with poor system reliability.

Problems not fixed promptly Example incident reports identified by Solr using search on terms which might indicate ongoing computer problems.

- "Yet again 400 + results , some a year old , have appeared today for signing off . This is a significant risk which has been highlighted repeatedly and I am not aware that any action has been taken / or even that a clear explanation has been given for why it occurs . (Examples of 4 sample patients listed on form .) ."
- "Computerised ' Carevue' system for charting vital sighs and ventilation on Nicu and Picu freezing over past 24hrs . ICT phoned x2 and team in Vienna investigating . Was rebooted overnight . At 06.15 the carevue

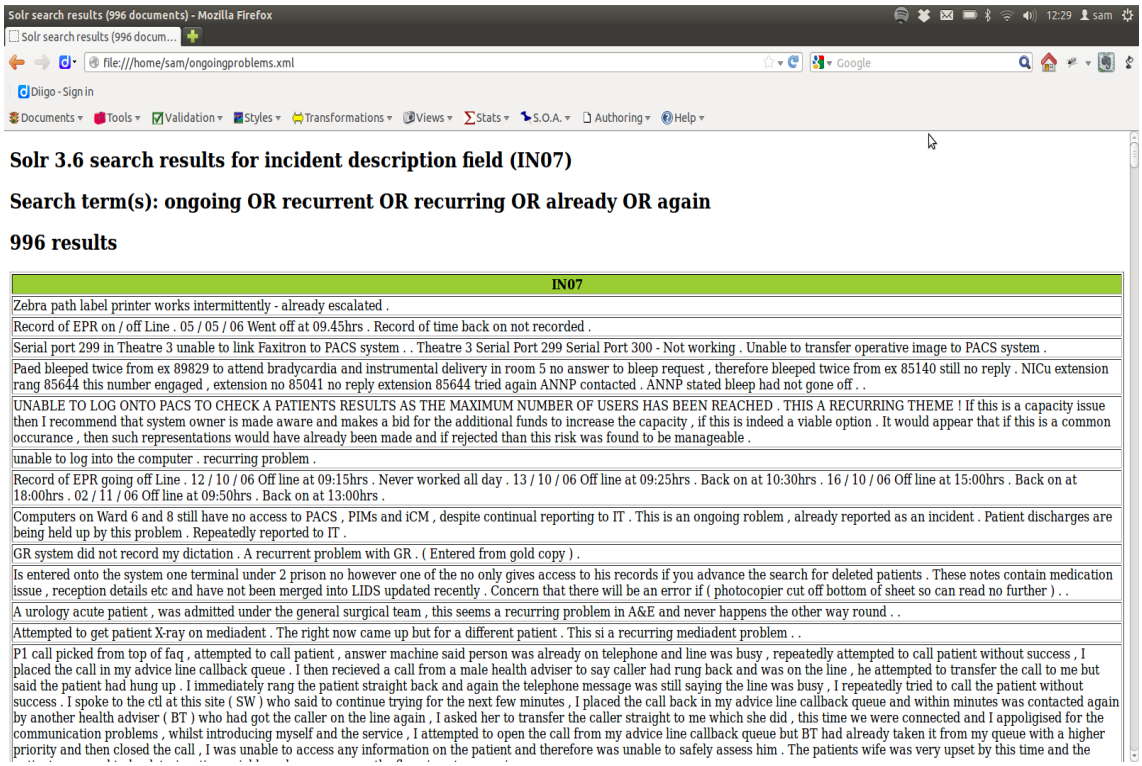

Figure 4.6: Searching the computer problems dataset for terms which might indicate an ongoing problem such as “ongoing OR recurrent OR recurring OR already OR again” yielded 996 results.

system again froze , this time completely , not allowing any entering or viewing of screens . This has an impact on patient safety as the day staff cannot review the patients and accuratly assess their clinical stability and plan treatment accordinglyly . Also , 2 new patients to NICU overnight , who clinical information cannot be accessed . ."

4.2.3 NLTK

Basic lexial analysis was carried out using NLTK (see section 3.4.3). This proved most useful for drilling down and exploring snippets relating to a theme identified by the lingo clustering algorithm, namely of users reporting being unable to do things because of computer system failure (see section 4.2.3).

Basic lexical statistics

There were 408657 words in the text. 14018 words formed the vocabulary. The lexical diversity (number of words in text/vocabulary) was 29 (see section 3.4.3).

Collocations

A collocation is a sequence of words that occur together unusually often. Thus, "red wine" is a collocation, whereas "the wine" is not. Collocations are resistant to substitution, for example, "maroon wine" would sound very odd. They tend to be specific to a given text and so can convey useful information.[46]

Collocations identified in "computer problem" extract were:

- Staff Name
- cardiac arrest
- computer system
- Health Advisor
- warm transfer
- staff member
- Contributing factors

- labour ward
- switch board
- health advisor
- call back
- Staff member
- blood results
- NHS Direct
- several times
- help desk
- ambulance control
- File Closed
- ambulance service
- Staff name

The top 10 trigrams, filtering for (removing) those occurring less than 10 times, were:

- First Advice Queue
- Abbott Diagnostic where
- regarding poor connectivity
- ongoing issues regarding
- serial no C160045
- instrument serial no
- issues regarding poor
- contributing factors
- front sheet did

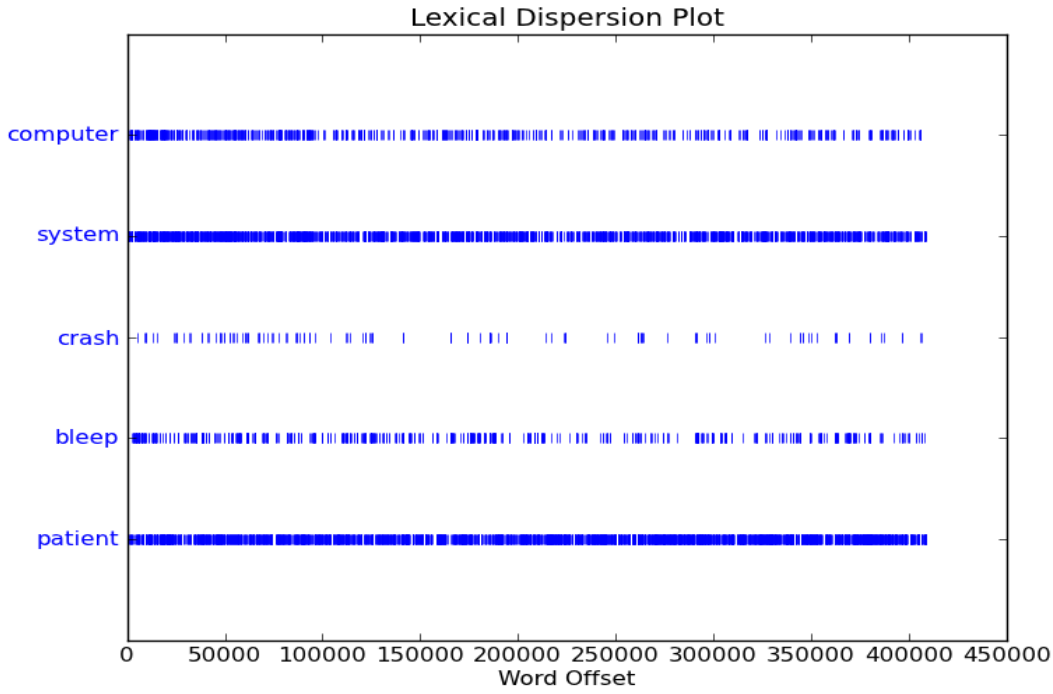

Figure 4.7: Each stripe represents an instance of a word and each row represents the entire text. If present, trends in the frequency of the use of words over time can be visualized in this way (the incident descriptions forming the text have been time ordered).

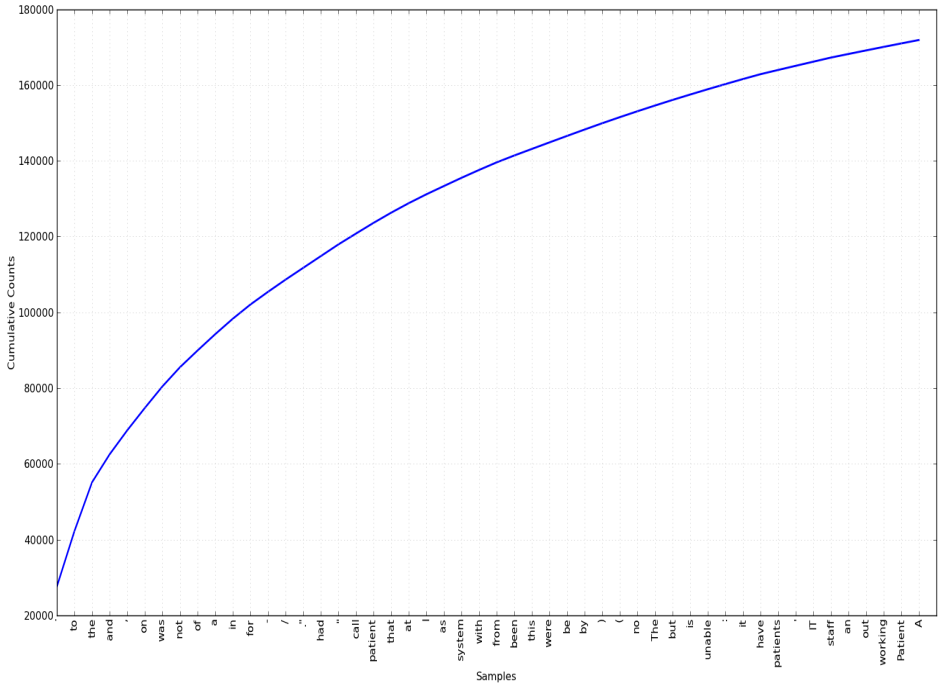

Figure 4.8: Cumulative frequency plot for the 50 most frequently used words in incident reports about computer problems.

Unable to snippets

A subset of poor system reliability was identified as to do with reporters being unable to perform particular tasks due to computer system failure using the lingo clustering algorithm. Using NLTK it was established that the words "unable to" were followed by another word in 1398 instances. 256 unique words made up these instances.

Top 10 words to follow the expression "unable to" in the data set:

```
[('access', 318),
('contact', 108),
('get', 84),
('print', 71),
('log', 68),
('transfer', 48),
('use', 47),
('view', 47),
```

```
('be', 43),
('hear', 42)]
```

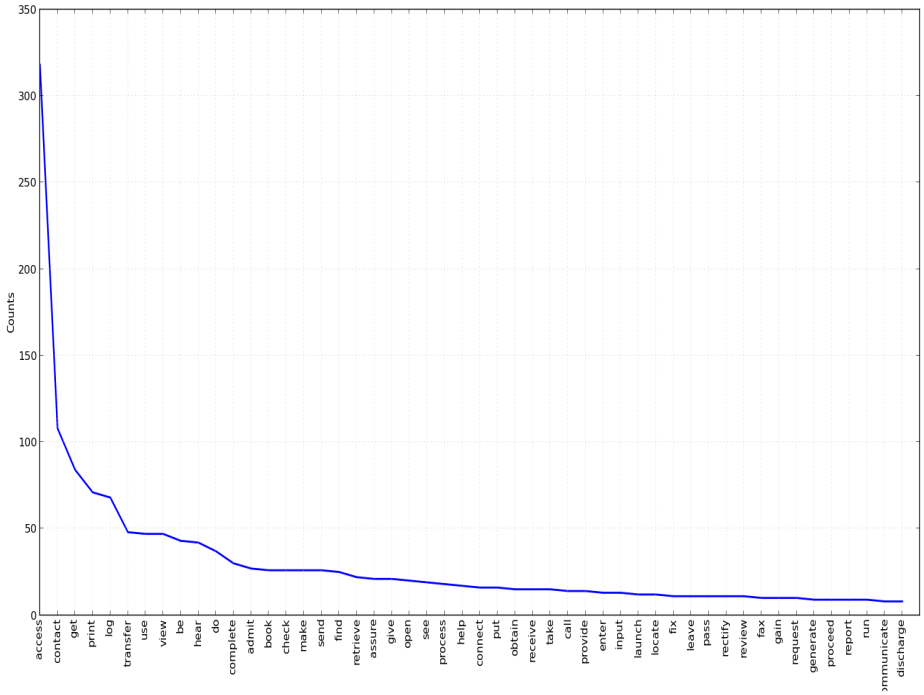

Figure 4.9: Frequency distribution plot of the 50 most common words to follow the words “unable to” in the text.

Named entity recognition

Parts of speech (POS) tagging was of very limited value in this case because the NPSA (usually) removes names of persons, companies, and locations, as part of their data cleaning.

4.3 Lingo clustering algorithm

Using the lingo clustering algorithm (see section 3.5) incidents are grouped without human supervision. The two most reliable discreet clusters identified were to do with bleep problems and being unable to access systems because of system failure (see section 4.3). Examples of incidents grouped under the bleep problem cluster selected to show the difficulty of the task are shown below (see section 4.3.1).

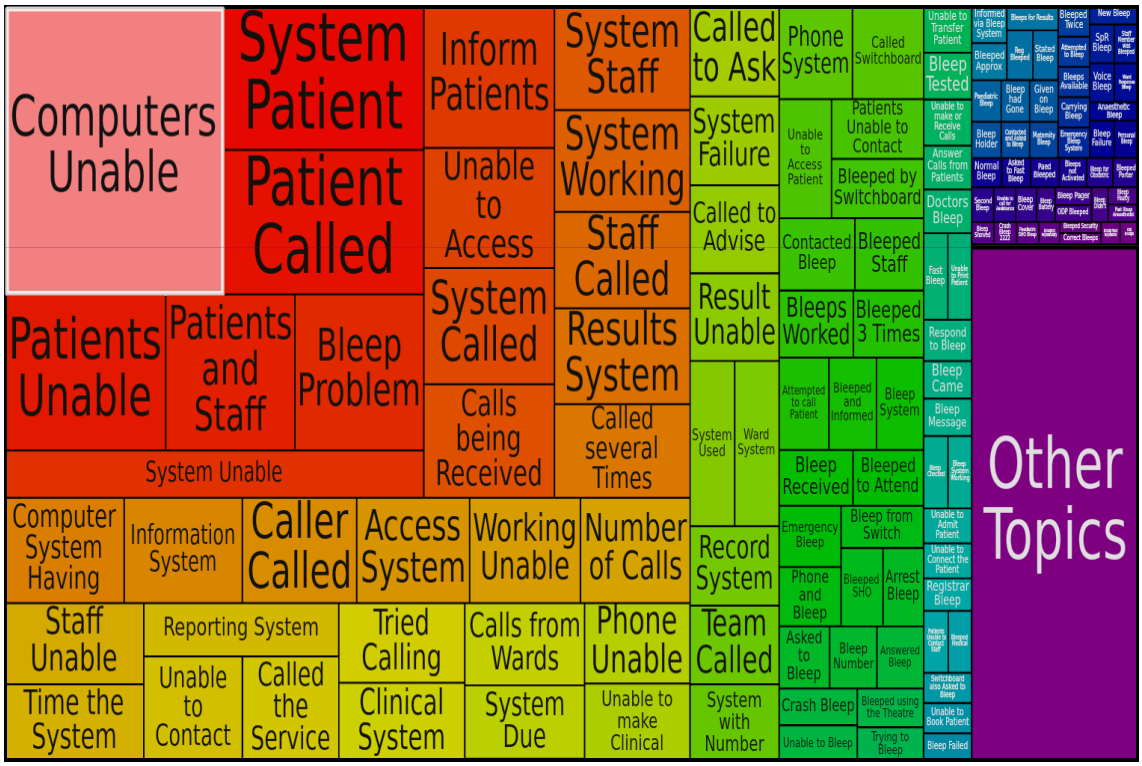

Figure 4.10: Automatically generated clusters and labels with the lingo algorithm using the Carrot2 platform. Box size is proportional to the number of incidents within the cluster. Colour is arbitrary.

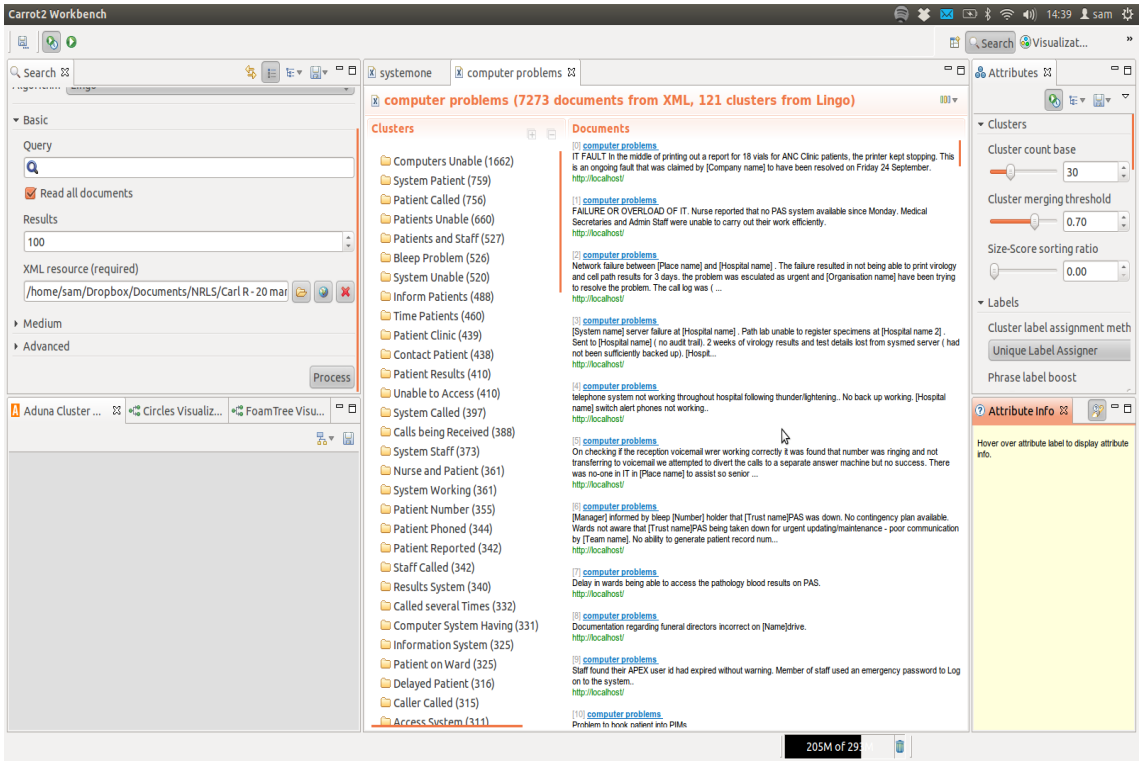

Figure 4.11: Overview of the lingo clusters and setup screen.

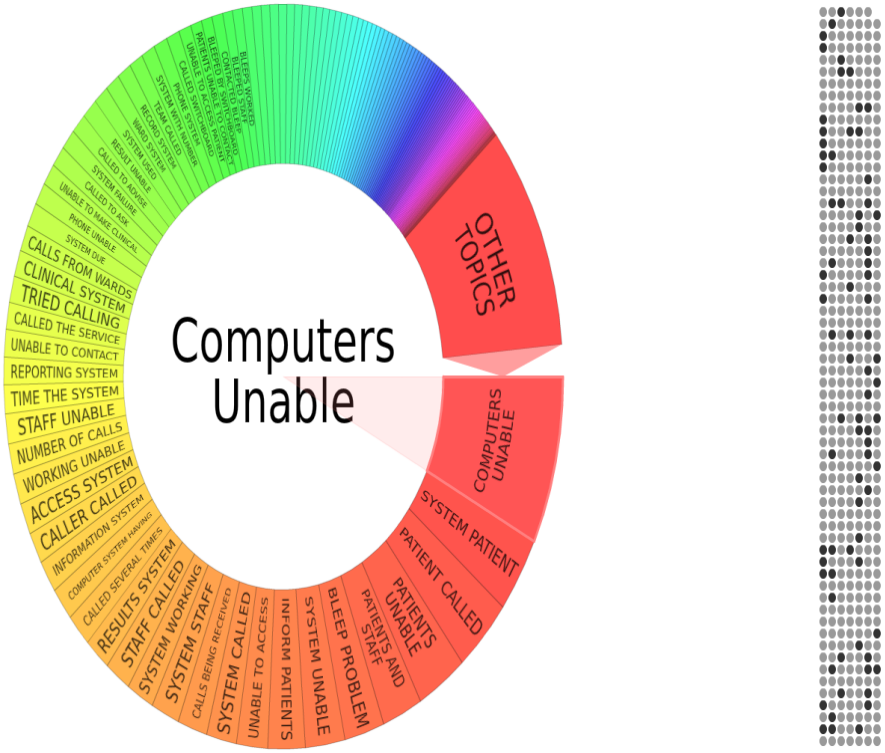

Figure 4.12: Automatically generated clusters and labels with the lingo algorithm using the Carrot2 platform. Box size is proportional to the number of incidents within the cluster, circle visualisation. Colour is arbitrary.

Cluster label	Number of documents
Computers Unable	1662
System Patient	759
Patient Called	756
Patients Unable	660
Patients and Staff	527
Bleep Problem	526
System Unable	520
Inform Patients	488
Time Patients	460
Patient Clinic	439

Table 4.9: Automatically generated cluster labels for the ten largest clusters identified by the lingo algorithm

4.3.1 Bleep problems

Bleep problems were identified as a large cluster with a high reliability score using lingo. However, these exerpts show that whlile some of the incidents identified are clearly to do with bleep problems others are not.

- Bleep problem: "ODP bleeped 3 times giving no reply, rang main theatres who said ODP was in theatre 17. The bleep system does not work in this theatre, which is widely known, expect the ODP was unaware of this."
- Computer problem but not to do with bleep system: "[Manager] informed by bleep [Number] holder that [Trust name]PAS was down. No contingency plan available. Wards not aware that [Trust name]PAS being taken down for urgent updating/maintenance - poor communication by [Team name]. No ability to generate patient record num..."
- Unclear if bleep sytem problem or not: Pt daughter was feeding her. When a staff nurse was passing pt's daughter said that her mother was not well, having difficulty breathing. Went to see pt - breathing shallow. Oxygen

Cluster label	Cluster reliability score
Bleep Problem	21.8
Unable to Access	12.4
System Staff	11.0
System Working	10.3
Calls being Received	9.6
Patients and Staff	9.3
Staff Called	7.9
Working Unable	7.6
Inform Patients	7.4
Called several Times	7.4

Table 4.10: Automatically generated cluster labels for the ten highest reliability scores identified by the lingo algorithm. The higher the reliability score the higher the reliability of the cluster content.

> was started. Bleeped Dr twice - no reply. Co-Ord informed. Co-Ord came immediately - card...

4.4 Classification with Scikit-learn

Performance was not even across the severity classes but reasonable overall classifier accuracy, precision, and recall were achieved with the Stochastic Gradient Descent classifier performing marginally better than the Natural Bayes classifier (see section 3.6).

Classifier selection

```
"""
#Sample of initial results with Naive Bayes and SGDClassifier classifiers
#from classifierselection.py output
"""
```

Cluster label	Number of documents	Cluster reliability score
Bleeps Worked	131	47.6
Bleeped 3 Times	127	44.5
Bleeped and Informed	117	27.8
Bleep System	115	50.2
Bleep Received	112	48.1
Bleeped to Attend	104	28.1
Emergency Bleep	103	49.8
Phone and Bleep	99	51.5
Bleep from Switch	91	41.8
Bleeped SHO	86	57.8

Table 4.11: Higher cluster reliability scores are achieved by applying lingo algorithm to incidents containing the word 'bleep'.

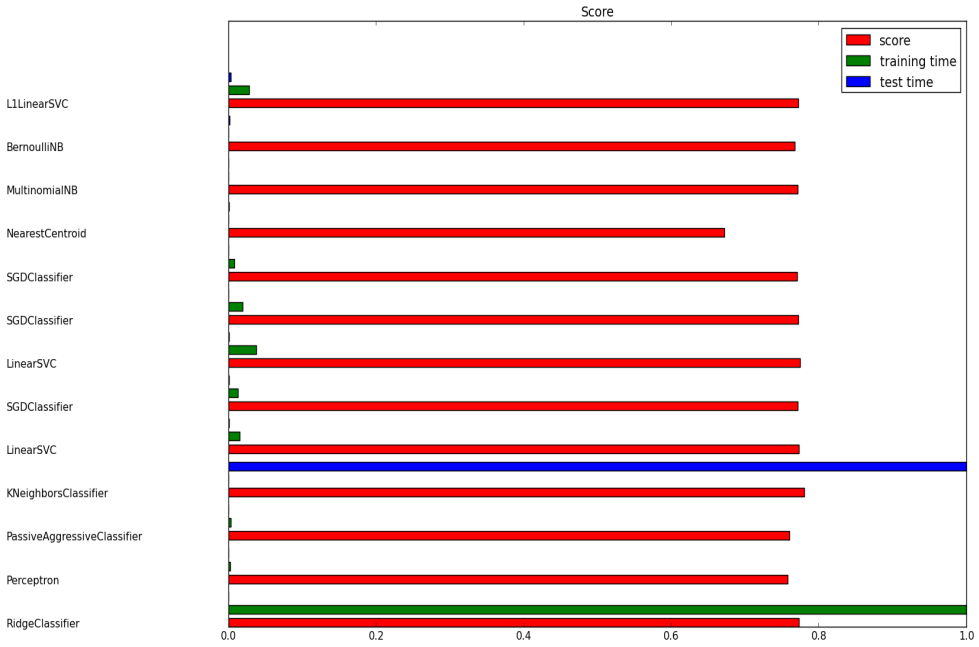

Figure 4.13: Accuracy comparison for different supervised machine learning algorithms and settings generated by classifierselection.py

```
================================================================================
Naive Bayes

--------------------------------------------------------------------------------
Training:
MultinomialNB(alpha=0.01, class_prior=None, fit_prior=True)

classification report:
             precision    recall  f1-score   support

      Death       0.00      0.00      0.00         1
        Low       0.51      0.06      0.11       288
   Moderate       0.00      0.00      0.00        93
     NoHarm       0.84      0.99      0.91      2021
     Severe       0.00      0.00      0.00        22

avg / total       0.76      0.83      0.77      2425



--------------------------------------------------------------------------------


================================================================================
SGDClassifier

--------------------------------------------------------------------------------
Training:
SGDClassifier(alpha=0.0001, class_weight=None, epsilon=0.1, eta0=0.0,
       fit_intercept=True, l1_ratio=0.15, learning_rate=optimal,
       loss=hinge, n_iter=50, n_jobs=1, penalty=elasticnet, power_t=0.5,
       random_state=None, rho=None, shuffle=False, verbose=0,
       warm_start=False)

classification report:
```

	precision	recall	f1-score	support
Death	0.00	0.00	0.00	1
Low	0.65	0.06	0.11	288
Moderate	0.00	0.00	0.00	93
NoHarm	0.84	0.99	0.91	2021
Severe	0.00	0.00	0.00	22
avg / total	0.78	0.84	0.77	2425

==========

```
#gridsearch.py output

4847 documents
5 categories

Performing grid search...
pipeline: ['vect', 'tfidf', 'clf']
parameters:
{'clf__alpha': (1e-05, 1e-06),
 'clf__n_iter': (10, 50, 80),
 'clf__penalty': ('l2', 'elasticnet'),
 'tfidf__norm': ('l1', 'l2'),
 'tfidf__use_idf': (True, False),
 'vect__max_df': (0.5, 0.75, 1.0),
 'vect__max_features': (None, 5000, 10000, 50000),
 'vect__ngram_range': ((1, 1), (1, 2))}
done in 3681.902s

Best score: 0.842
Best parameters set:
```

```
^^Iclf__alpha: 1e-05
^^Iclf__n_iter: 80
^^Iclf__penalty: 'l2'
^^Itfidf__norm: 'l1'
^^Itfidf__use_idf: True
^^Ivect__max_df: 1.0
^^Ivect__max_features: None
^^Ivect__ngram_range: (1, 2)
```

```
#incidenclassify.py output for SGDClassifier with
#gridsearch optimised parameters
```

	precision	recall	f1-score	support
Death	0.00	0.00	0.00	1
Low	0.87	0.05	0.09	288
Moderate	0.00	0.00	0.00	93
NoHarm	0.84	1.00	0.91	2021
Severe	0.00	0.00	0.00	22
avg / total	0.80	0.84	0.77	2425

```
Confusion matrix showing poor performance of
classification to smaller classes for SGDClassifier
```

```
[[                        (predicted)
 [                 Death Low  Mod NoHarm Severe
 [           Death     0    0    0     1     0]
 [           Low       0   13    0   275     0]
 [ (actual)  Moderate  0    1    0    92     0]
 [           NoHarm    0    1    0  2020     0]
 [           Severe    0    0    0    22     0]]
```

Chapter 5

Discussion

5.1 Results

Google-refine and Python Brewery facilitated a rapid and useful assessment of data quality. Clustering reports with the lingo algorithm and identifying bigrams and trigrams yielded interesting results which could then rapidly be investigated further with Grep, the Apache Solr search engine and regular expressions in Python NLTK.

For example, problems concerning bleep system failure and problems around being unable to access information were both identified with clustering and NLTK (see sections 4.3, 4.3.1, and 4.2.3) and have been identified as useful categories in the literature.[40] Ongoing problems with health IT were also identified as a theme which may be useful to help build the case for rethinking the management models used to manage these systems. This resonates with the observations regarding ongoing issues and failure of the market to ensure safe health IT discussed in the literature review (see sections 2.0.3 and).

The clusters resulting from the application of the lingo algorithm were impressive but by no means perfect. The selected excerpts from incidents in the most reliable cluster, bleep problems, demonstrates the mixed success of the approach. While bleep problems were automatically identified as a meaningful category and the majority of incidents in the cluster did appear to be primarily about problems with the bleep system, some only mention the bleep system in passing and for some it is unclear what role the bleep system plays in the incident (see section

4.3.1).

Overall, the performance achieved by the classifiers built and tested to classify severity of harm from free text incident descriptions in this thesis was disappointing (see section 4.4). With optimal parameters the two most successful algorithms, the Natural Bayes and Stochastic Gradient Descent incident severity classifiers, performed similarly. Natural Bayes classifier: precision = 0.76, recall = 0.83, f1-score = 0.77. Stochastic Gradient Descent classifier: precision = 0.78, recall = 0.84, f1-score = 0.77. Performance was much worse for the prediction of the class labels 'Death' and 'Severe'. This is probably a function of how uncommon these incidents were in the dataset, either there were an insufficiently large number of incidents in these classes to train an accurate classifier and/or the incidents in these classes were insufficiently different from the incidents in other classes.

The disappointing performance achieved may be because of a relative lack of technical expertise of the author for the task, the sample size, or because the task is impossible. The experience of others would suggest the task is not impossible.[4]

Training in, and use of, modern tools such as Google-refine, Python Brewery, Carrot2, Apache Solr, and NLTK, would be likely to improve efficiency and augment analysis capacity beyond that achieved with current tools at the NPSA. In particular, addressing data quality issues using google-refine and python brewery is a relatively straightforward task that could substantially improve the quality of subsequent analysis. The Apache Solr search engine could also be especially useful since the interface and behaviour is familiar to those who have used google search and it is much faster than the Excel SAS plugin currently used and would permit a more interactive interrogation of the database.

Beyond potentially freeing up analysis time it is unclear whether the data mining methods tested could assist the discovery of new useful knowledge.

A major limitation of the overall taxonomy-classification approach is the tendency for incidents, once classified, to be reduced to being no more than members of their category, losing the richness provided by the detail of the report.[20] This is potentially especially troublesome when categories are very crude. For example, some authors have reduced computer software safety issues into the following

categories:[40]

1. software functionality
2. software system configuration
3. software interface with devices
4. network configuration

The categories may be apt but they make a sorry comparison with the routine incident (bug) reporting found online for open source software projects (see section 2.0.5) where it is the norm to provide a very detailed description of what occurred and the circumstances in which the event is reproducible. These information reports contain actionable information and classification is limited to a severity assessment and status (fixed yet or not fixed). Possibly, this contrast points to the value of community and peer production fostered by openness and engagement but also to the importance of tying more closely incident reports to positive action, and suggests the need for a technological and cultural shift.

Of note the severity assessments seen in bug reporting systems tend to have fewer than the five categories seen in NRLS patient safety reporting. The optimum number of categories may well be fewer than five and this should be investigated further.

It has been observed that much effort in health care is devoted to defining the incidents that should be reported and devising classification systems to capture them. Classification itself will not necessarily produce useful safety information, indexing should be used as a tool in analysis; the classification system it represents should not be the sole analysis carried out.[20 47] The more dynamic interaction with a large body of data permitted by data mining may help to safeguard against this being the case.

5.2 Limitations

The major limitation of this work is that while the author has the advantage of domain expertise in medicine and patient safety he is far from being a professional

statistician or computer programmer and, acting alone, no doubt vulnerable to methodological oversights and errors.

The data sample was obtained by a category search and it is well recognized that a proportion of incidents will be incorrectly classified.[48][34] However, searching by category did avoid the nuances of searching by spelling variant within a database for which spellings have not been reconciled.

Under-reporting is an issue known to affect all patient safety reporting and learning systems and is related partly to the lack of routine feedback given to those who report.[32][41] With respect to computer related problems undereporting may be especially common where safety incidents occur because of systems which are inadequate but do not malfunction.[49]

The quality of reports varies hugely with fields being frequently left empty or insufficiently detailed (see section 4.1.1). For example here are the incident descriptions which contained less that four 'tokens' (a token is a word or punctuation).

```
See above .
Computer crashed .
As above .
System failure .
test datix .
Unknown .
Computer fault .
```

It may also be that is some instances the person reporting the incident lacks an understanding of what happened resulting in an inadvertently misleading report.[34] Possibly a lack of understanding, may also be responsible for the infrequency with which the fields "Apparent Causes" and "Actions Preventing Reoccurence" were completed (see section 4.1.1).

5.3 Evaluation of data mining

5.3.1 Data mining evaluation criteria

Traditionally, evaluation methods include objective assessments such as the calculation a classifier's Accuracy, Precision, Recall, and F-score.

In practice, data mining is evaluated by the usefulness of the insights generated to the organization or individual who invests in it. In the world of business 'usefulness' or return on investment may be quantified objectively on a balance sheet.[10 37]

Unless the outcomes of the data mining process are objectively compared with existing processes, or the data mining process is tied to an intervention which affects a measurable outcome, then objective demonstration of the value of data mining is hard and depends on the subjective assessment of whether value has been added by domain experts.[42 4 37]

5.4 Future applications

Health IT has been recognized as an important source of patient safety incidents and there have been calls for a national EMR adverse event monitoring system in America.[50] The case for iteration and continuous improvement and monitoring of health IT systems, and the relationship between poor usability and safety issues, has been well made.[51 52 28 29] With appropriate data capture and analysis health IT safety issues should be particularly amenable to data mining techniques coupled with software and hardware fixes.

When the move to electronic medical records is achieved, records could be routinely monitored to detect and act on safety incidents in near real time.[23] Richer data capture may permit more useful classifiers which are tied to appropriate trigger responses.[53 54]

This may be to prevent a harm directly, or indirectly, to assist in prioritization of incidents, and encourage reporting by giving feedback. For example, a user may be informed of the risk present by the system when he or she is performing a task classified as high risk.

For example, the constellation of a junior doctor prescribing something they've never prescribed before, to a frail elderly patient with complex comorbidities, at two in the morning may be a particularly error prone situation. Classifiers may allow automated identification of non-trivial high risk occasions which would permit controlled experimentation around linking the classifier to an effective intervention (one that prevents harm).

Another application would be a free text severity classifier which, trained effectively with a sufficiently large dataset, could be employed to inform action, and provide feedback to a user when they submit a report, encouraging reporting and reducing the rate of incorrect classification.[48 34 4 32 4]

Part IV

Conclusion

"Knowing is not enough; we must apply. Willing is not enough; we must do." Johann Wolfgang von Goethe (1749-1832)

Chapter 6

Conclusion

6.1 Can data mining help us to learn from patient safety incidents?

6.1.1 Yes for problem topic discovery

Data mining can demonstrably help with problem discovery in a non-trivial way. Some of the candidate clusters and cluster-labels generated by the lingo algorithm are meaningful and useful. The general purpose tools and techniques of data mining have the potential to allow appropriately trained analysts to carry out their role more efficiently, and may improve novel incident type detection.

6.1.2 Probably for specific classification tasks

There is ample evidence from the literature that accurate predictive models can be built for classification purposes. In some instances, such as free text severity classification, these methods could have immediate utility.

Other applications, such as classifying circumstances and acting to prevent harm in an automated fashion, or facilitating strategies to prevent harm, using electronic medical records, will be important, and helpful, applications of data mining in the future.

With the increasing digitisation of health care demonstration of utility may be more easily obtained when reporting, expert analysis, and action are more closely integrated into tighter feedback loops. For example a classifier used to predict

prescription error might be constructed and tied to a prevention intervention with the goal of reducing the rate of a specific measurable harm occurring.

For the subset of patient safety incidents relating to computer use considered, the application of DM methods suggests that the NRLS system does not result in the timely resolution of safety issues. For safety incidents due to computer problems, and possibly other types of safety incident, lessons might be learned from a more general approaches to, and cultures of, systems improvement. In particular, the more open and action focussed approach to improving quality present in bug reporting systems found in the open source software community has intuitive appeal.

Part V

Appendix

Glossary

Accuracy Accuracy is number of true positives plus false negatives all divided by the number of true and false positives and negatives.. 25, 34, 80, 87

Anomaly detection Anomaly detection refers to the application of statistical and machine learning techniques to detect outliers. An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.. 32, 87

Big Data 'Big' is a moving target, and depends on available computing power, but at present refers to data which is least gigabytes big.. 20, 87

CfH Connecting for Health. 87

Chunking Chunking is recovering phrases constructed by the part of-speech tags. An example would be recovering all of the noun phrases from a text.. 87

Collocation Collocations are words that occur more frequently together than would be predicted by chance. A good example is "red wine" rather than say "maroon wine".. 87

Computer is a programmable machine that receives input, stores and manipulates data, and provides output in a useful format. 87

Cross-validation Cross-validation, sometimes called rotation estimation, is a technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. One round of cross-validation involves partitioning

a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). 87

Data Mining is the analysis of (often large) observational data sets to find unexpected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner. 87

Data Science Data Science is the practice of deriving valuable insights from dat comprises Data Engineering, Scientific Method, Math, Statistics, Advanced Computing, Visualization, and Domain Expertise. 20, 87

Dimensionality reduction Dimensionality reduction is the process of reducing the number of variables under consideration in order to facilitate analysis. It can be divided into feature selection and feature extraction.. 32, 87

DM Data Mining. 20, 21, 87

Domain Knowledge Domain Knowledge is specialist knowledge about a particular field of activity. For example, a patient safety expert would have specialist knowledge of safety issues in health care.. 87

EMR Electronic Medical Record. 29, 87

F-score Also called F-measure, this is the harmonic mean of precision and recall: $F\text{-}score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$. 25, 34, 80, 87

Knowledge Discovery in Databases The process of discovering knowledge from a database involving the selection of data, preprocessing, transforming, mining to extract patterns and relationships, and then interpreting and assessing the discovered structures. 20, 87

Naive Bayes A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions about the features (variables) under consideration.. 32, 87

NPfIT National Programme for Information Technology. 26, 87

NPSA National Patient Safety Agency. 16, 18, 19, 26, 87

NRLS National Reporting and Learning Service. 16, 18, 19, 26–28, 87

Overfitting Overfitting occurs when a statistical model describes random error or noise instead of the underlying relationship.. 87

Patient safety incident Any unintended or unexpected incident which could have or did lead to harm for one or more patients receiving NHS-funded care (this definition incorporates all terms such as adverse incidents, adverse events and near misses). 18, 87

Pointwise Mutual Information Pointwise Mutual Information is a very simple information-theoretic measure that, when computed between two words x and y, "compares the probability of observing x and y together (the joint probability) with the probabilities of observing x and y independently (chance)".. 87

POS Part-of-speech. 87

Precision Precision is calculated by dividing the number of true positives by the total number of identified positives (true positive + false positives). It is also called the positive predictive value.. 25, 34, 80, 87

Recall Recall is calculated by dividing the number of true positives by the total number of positives (true positive + false negatives). It is also called the sensitivity.. 25, 34, 80, 87

Singular Value Decomposition Singlular value decomposition a technique in linear algebra for factorizing a real or complex matrix. 45, 87

Stop words Stop words are words that are removed (filtered) prior to or after natural language processing to improve analysis.. 87

Support Vector Machine A support vector machine is a supervised learning algorithm constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks.. 32, 87

Term Frequency Inverse Document Frequency The term frequency inverse document frequency (Tf–idf) is the product of two statistics and is used to give each word a weighting score. Term frequency weighting, which is related to the frequency of the appearance of a word in a particular incident description. The second is the IDF or inverse document frequency weighting, which is related to the number of descriptions in which a word appears. The net effect is to scale down the weighting of unimportant words such as "the".. 87

Tokenizers Tokenizers break up a stream of text into words, phrases, symbols, or other meaningful elements, called tokens. The process is called tokenization.. 87

Vector Space Model In a vector space model each dimension in the vector space corresponds to a particular word, and the position of an entry in a particular dimension is a function of the number of times the corresponding word appears in that piece of text.. 32, 45, 87

Appendix A

Tools used

- SAS Excel plugin
- Python with NLTK, Brewery, Scikit-learn, and Matplotlib modules
- Carrot2
- RapidMiner with the text processing and named entity recognition plugins
- Apache Solr
- Google Refine
- Kile LaTeXeditor
- Jabref reference manager
- LaTeXwith bibtex, pythontex and several other packages
- Vim
- Ubuntu Linux

Appendix B

Source code

B.1 Python code

B.1.1 lexicalanalysis.py

```
#Adapted from code examples in S. Bird, E. Klein, and E. Loper.
#Natural language processing with Python. O'Reilly Media, 2009
#Takes a text file and tokenizes it words, converts to lower case,
#filters stop words, builds vocab for text, calculates lexical diversity,
#builds collocation, builds frequency distribution of most common words,
#builds example dispersion plot of words of interest
#(manually entered below in this script), displays results

#lexical_analysis.py


import nltk
from nltk.corpus import stopwords
from sys import argv

script, inputfilename = argv #takes whatever filename you pass in

print inputfilename
```

```
raw = open("%s" % inputfilename).read()


#loads incident descriptions identified as being due to computer problems

tokens = nltk.wordpunct_tokenize(raw) #tokenizes free text

#tokens = [nltk.PorterStemmer().stem(t) for t in tokens]
# uncomment to stem tokens

#tokens = [nltk.WordNetLemmatizer().lemmatize(t) for t in tokens]
# uncomment to lemmatize tokens

text = nltk.Text(tokens) #defines text

words = [w.lower() for w in text]
#defines words and makes all words lower case

filtered_words = [w for w in words if w not in stopwords.words('english')]
#removes commonly occurring words ("stop words")

vocab = sorted(set(words)) #defines vocabulary

def lexical_diversity(text): #calculate lexical diversity
    return len(text) / len(set(words))

print "the number of words in the text is %d" % len(text)

print "the number of words in the vocabulary is %d" % len(vocab)

print "lexical diversity is %d" % lexical_diversity(text)
```

```
#prints lexical diversity

text.collocations() #builds collocations

fdist = nltk.FreqDist(ch.lower() for ch in filtered_words if ch.isalpha())

fdist.plot(50, cumulative=True)
#prints a cumulative frequency distribution
#of the 50 most commonly used words in the text

text.dispersion_plot(["computer", "system", "crash", "bleep", "patient"])
#example dispersion plot using arbitary search terms
```

B.1.2 mrchunk.py

```
#takes a raw text file and extracts noun phrases to
#(hopefully) discover interesting relationships
#mrchunk.py

import nltk
import nltk.tag
from nltk.tokenize import word_tokenize, wordpunct_tokenize, sent_tokenize
from sys import argv


script, inputfilename = argv

raw = open("%s" % inputfilename).read()

sentences = sent_tokenize(raw)

wordsenttoke = [word_tokenize(t) for t in sent_tokenize(raw)]
```

```
chunks = nltk.tag.batch_pos_tag(wordsenttoke)

patterns =  "NP:{<DT>?<JJ>*<NN>}"
#DT = determiner e.g the (optional)
#JJ = adjective e.g big (can have any number of adjectives)
#NN = noun e.g dog

NPChunker = nltk.RegexpParser(patterns)

nplist = []


def sub_leaves(tree, node):
    return [t.leaves() for t in tree.subtrees
    (lambda s: s.node == node)]

#a tree traversal function for extracting NP chunks in the parsed tree
def chunk(patterns):
    for sent in chunks:
        tree = NPChunker.parse(sent)
        for subtree in tree.subtrees():
            if subtree.node == 'NP':
                print subtree
                nplist.append(subtree)
chunk(patterns)

print len(nplist)

ne_chunks = nltk.batch_ne_chunk(nplist)

print ne_chunks #print all chunks
```

```
#print chunks tagged as people
for i in range(len(ne_chunks)):
    tree = ne_chunks[i]
    print 'PERSON'
    print sub_leaves(tree, 'PERSON')

#print chunks tagged as orgs
for i in range(len(ne_chunks)):
     tree = ne_chunks[i]
     print 'ORGANIZATION'
     print sub_leaves(tree, 'ORGANIZATION')

#print chunks tagged as
#geo-political entities
for i in range(len(ne_chunks)):
     tree = ne_chunks[i]
     print 'GPE'
     print sub_leaves(tree, 'GPE')

#fdist = nltk.FreqDist(nplist)
#fdist.plot(50, cumulative=True)
```

B.1.3 unableanalysis.py

```
#takes a text file and returns
#a frequency distribution plot of
#the 50 most common words to follow
#the expression ``unable to''
#unable_analysis.py

import nltk

raw = open('computers.csv').read()
```

```
tokens = nltk.wordpunct_tokenize(raw)

text = nltk.Text(tokens)

text.concordance("unable", width=40, lines=100)

text.findall("<unable><to><.*><.*>")

words = [w.lower() for w in text]

c = nltk.ConcordanceIndex(text.tokens)

unableset = [text.tokens[offset+2] for offset in c.offsets('unable')]

print len(unableset)

words = [w.lower() for w in unableset]

vocab = sorted(set(words))

print len(vocab)

fdist = nltk.FreqDist(unableset)

fdist.plot(50, cumulative=False)

fdist.items()[:10] #prints a list of the top 10

print vocab
```

B.1.4 csv2xml.py

```
#selects free text incident description column from .csv and
#creates a Carrot2 .xml input file
#csv2xml.py

# -*- coding: utf-8 -*-
import sys,csv
from xml.dom.minidom import Document

def convert(input_file, output_file, header, query_text):
    print 'Reading: ', input_file
    print 'Writing: ', input_file
    print 'Data in: ', header

    doc = Document(  )
    results = doc.createElement("searchresult")
    doc.appendChild(results)
    query = doc.createElement("query")
    query.appendChild( doc.createTextNode(query_text) )
    results.appendChild( query )

    count = 0
    checked_header = False

    with open(input_file, 'Ur') as f:
        reader = csv.DictReader(f)
        for row_dict in reader:
            if not checked_header and header not in row_dict:
                print ("Unable to locate column named %s, check case and "
                       "header row in CSV" % header)
                sys.exit(1)
            checked_header = True
```

```
            results.appendChild( create_doc_element( doc, query_text,
                                                     row_dict[header], count ) )
            count = count + 1

    with open(output_file, "wb") as f:
        f.write( doc.toprettyxml(indent="  ", encoding="UTF-8") )

def create_doc_element(doc, title_text, text, id):
    n = doc.createElement('document')
    n.setAttribute("id", str(id))

    title = doc.createElement('title')
    title.appendChild( doc.createTextNode( title_text ) )

    url = doc.createElement('url')
    url.appendChild( doc.createTextNode( "http://localhost/" ) )

    snippet = doc.createElement('snippet')
    snippet.appendChild( doc.createTextNode( text ) )


    n.appendChild( title )
    n.appendChild( url )
    n.appendChild( snippet )
    return n

"""  <document id="0">
    <title>default</title>
    <url>http://www.globe.com.ph/</url>
    <snippet>
      Provides mobile communications (GSM) including
```

```
      GenTXT, handyphones, wireline services, an
      broadband Internet services.
    </snippet>
  </document>
  <document id="1">
    <title>Skate Shoes by Globe | Time For Change</title>
    <url>http://www.globeshoes.com/</url>
    <snippet>
      Skaters, surfers, and showboarders
      designing in their own style.
    </snippet>
  </document>

  ...

</searchresult>
"""

if __name__ == '__main__':
    if len(sys.argv) != 4:
        print """
Usage:
    python csv_to_xml.py <input file> <col name> <query text>

  i.e.
    python csv_to_xml.py input.xml IN07 "computer failures"

  You need to specify the input file and the column name, and don't
  forget to trim any junk before the header fields. Also don't forget to
  quote query text if it has spaces"""
        sys.exit(1)
```

```
    convert( sys.argv[1], sys.argv[1] + '.xml', sys.argv[2], sys.argv[3])
```

B.1.5 incidentsplit.py

```
#takes an input csv and splits it into a training.csv and testing.csv.
#we do this to avoid overfitting when training classifiers.
#incidentsplit.py

import os
import csv

csvfile = csv.reader(open('computer_problems_IN07_and_category.csv', 'rb'))

trainingf = open('training.csv', 'wb')
testingf = open('testing.csv', 'wb')

trainingw = csv.writer(trainingf)
testingw = csv.writer(testingf)

#categories
#categories = ['train', 'test']

#set up a random index to split the files in 2:1 ratio
import numpy as np
from numpy.random import RandomState

n_samples = 7273 #number of rows in csv sample file

indices = np.arange(n_samples)
RandomState(42).shuffle(indices)
split = (n_samples * 2) / 3

#iterate through input file, split it, and write to two new files
```

```
for row in csvfile:
        if csvfile.line_num in indices[:split]:
            #category = 'train'
            trainingw.writerow(row)
        elif csvfile.line_num in indices[split:]:
            #category = 'test'
            testingw.writerow(row)

trainingf.close()
testingf.close()
```

B.1.6 incidentextract.py

```
#takes a csv and prints incident desc from row to pre-made folder
#matching severity of incident
#incidentextract.py

import os
import csv

#set up category directories
#dirname = 'category'
#if not os.path.exists(dirname):
#        os.makedirs(dirname)

#open csv file
csvfile = csv.reader(open('training.csv', 'rb')) #manually set this presently

#categories
#categories = ['Death', 'Severe', 'Moderate', 'Low', 'No Harm']

#set up row
i = 0
```

```
for row in csvfile:
        if row[1] == 'Death':
            category = 'Death'
            #set incident report name
            incident = category + str(i)
            print incident
            #set where will write incidents to
            completeName = os.path.abspath('./%s/%s' % (category, incident))
            #open file to write incident to
            f = open(completeName, 'w')
            #write to file
            f.write(row[0])
            #close file
            f.close()
            i = i + 1

        elif row[1] == 'Severe':
            category = 'Severe'
            incident = category + str(i)
            print incident
            completeName = os.path.abspath('./%s/%s' % (category, incident))
            f = open(completeName, 'w')
            f.write(row[0])
            f.close()
            i = i + 1

        elif row[1] == 'Moderate':
            category = 'Moderate'
            incident = category + str(i)
            print incident
            completeName = os.path.abspath('./%s/%s' % (category, incident))
```

```
            f = open(completeName, 'w')
            f.write(row[0])
            f.close()
            i = i + 1

        elif row[1] == 'Low':
            category = 'Low'
            incident = category + str(i)
            print incident
            completeName = os.path.abspath('./%s/%s' % (category, incident))
            f = open(completeName, 'w')
            f.write(row[0])
            f.close()
            i = i + 1

        elif row[1] == 'No Harm':
            category = 'NoHarm'
            incident = category + str(i)
            print incident
            completeName = os.path.abspath('./%s/%s' % (category, incident))
            f = open(completeName, 'w')
            f.write(row[0])
            f.close()
            i = i + 1
```

B.1.7 classifierselection.py

```
#20 newsgroup example adapted to use computer problem data
#classifierselection.py

"""
==========================================================
Classification of text documents using sparse features
```

```
=========================================================

This is an example showing how scikit-learn can be used to classify documents
by topics using a bag-of-words approach. This example uses a scipy.sparse
matrix to store the features and demonstrates various classifiers that can
efficiently handle sparse matrices.

The dataset used in this example is the 20 newsgroups dataset. It will be
automatically downloaded, then cached.

The bar plot indicates the accuracy, training time (normalized) and test time
(normalized) of each classifier.

"""

# Author: Peter Prettenhofer <peter.prettenhofer@gmail.com>
#         Olivier Grisel <olivier.grisel@ensta.org>
#         Mathieu Blondel <mathieu@mblondel.org>
#         Lars Buitinck <L.J.Buitinck@uva.nl>
# License: Simplified BSD

from __future__ import print_function

import logging
import numpy as np
from optparse import OptionParser
import sys
from time import time
import pylab as pl
import sklearn #so can use own data

from sklearn.datasets import fetch_20newsgroups
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import HashingVectorizer
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.linear_model import RidgeClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import Perceptron
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.naive_bayes import BernoulliNB, MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import NearestCentroid
from sklearn.utils.extmath import density
from sklearn import metrics


# Display progress logs on stdout
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(levelname)s %(message)s')


# parse commandline arguments
op = OptionParser()
op.add_option("--report",
              action="store_true", dest="print_report",
              help="Print a detailed classification report.")
op.add_option("--chi2_select",
              action="store", type="int", dest="select_chi2",
              help="Select some number of features using a chi-squared test")
op.add_option("--confusion_matrix",
              action="store_true", dest="print_cm",
              help="Print the confusion matrix.")
op.add_option("--top10",
```

```
                  action="store_true", dest="print_top10",
                  help="Print ten most discriminative terms per class"
                       " for every classifier.")
op.add_option("--all_categories",
                  action="store_true", dest="all_categories",
                  help="Whether to use all categories or not.")
op.add_option("--use_hashing",
                  action="store_true",
                  help="Use a hashing vectorizer.")
op.add_option("--n_features",
                  action="store", type=int, default=2 ** 16,
                  help="n_features when using the hashing vectorizer.")


(opts, args) = op.parse_args()
if len(args) > 0:
    op.error("this script takes no arguments.")
    sys.exit(1)

print(__doc__)
op.print_help()
print()


################################################################################
# Load some categories from the training set
if opts.all_categories:
    categories = None
else:
    categories = [
        'no harm',
        'low harm',
```

```
        'moderate harm',
        'severe harm',
        'death',
    ]

print("Loading patient safety incident dataset for categories:")
print(categories if categories else "all")

#data_train = fetch_20newsgroups(subset='train', categories=categories,
#                                shuffle=True, random_state=42)

#data_test = fetch_20newsgroups(subset='test', categories=categories,
#shuffle=True, random_state=42)

data_train = sklearn.datasets.load_files("training1balanced")
data_test = sklearn.datasets.load_files("testing1")


print('data loaded')

categories = data_train.target_names    # for case categories == None


def size_mb(docs):
    return sum(len(s.encode('utf-8')) for s in docs) / 1e6

data_train_size_mb = size_mb(data_train.data)
data_test_size_mb = size_mb(data_test.data)

print("%d documents - %0.3fMB (training set)" % (
    len(data_train.data), data_train_size_mb))
print("%d documents - %0.3fMB (training set)" % (
```

```
    len(data_test.data), data_test_size_mb))
print("%d categories" % len(categories))
print()

# split a training set and a test set
y_train, y_test = data_train.target, data_test.target

print("Extracting features from the training dataset using a sparse vectorizer
t0 = time()
if opts.use_hashing:
    vectorizer = HashingVectorizer(stop_words='english', non_negative=True,
                                   n_features=opts.n_features)
    X_train = vectorizer.transform(data_train.data)
else:
    vectorizer = TfidfVectorizer(sublinear_tf=True, max_df=0.5,
                                 stop_words='english')
    X_train = vectorizer.fit_transform(data_train.data)
duration = time() - t0
print("done in %fs at %0.3fMB/s" % (duration, data_train_size_mb / duration))
print("n_samples: %d, n_features: %d" % X_train.shape)
print()

print("Extracting features from the test dataset using the same vectorizer")
t0 = time()
X_test = vectorizer.transform(data_test.data)
duration = time() - t0
print("done in %fs at %0.3fMB/s" % (duration, data_test_size_mb / duration))
print("n_samples: %d, n_features: %d" % X_test.shape)
print()

if opts.select_chi2:
    print("Extracting %d best features by a chi-squared test" %
```

```
              opts.select_chi2)
    t0 = time()
    ch2 = SelectKBest(chi2, k=opts.select_chi2)
    X_train = ch2.fit_transform(X_train, y_train)
    X_test = ch2.transform(X_test)
    print("done in %fs" % (time() - t0))
    print()


def trim(s):
    """Trim string to fit on terminal (assuming 80-column display)"""
    return s if len(s) <= 80 else s[:77] + "..."


# mapping from integer feature name to original token string
if opts.use_hashing:
    feature_names = None
else:
    feature_names = np.asarray(vectorizer.get_feature_names())


###############################################################################
# Benchmark classifiers
def benchmark(clf):
    print('_' * 80)
    print("Training: ")
    print(clf)
    t0 = time()
    clf.fit(X_train, y_train)
    train_time = time() - t0
    print("train time: %0.3fs" % train_time)
```

```
    t0 = time()
    pred = clf.predict(X_test)
    test_time = time() - t0
    print("test time:  %0.3fs" % test_time)

    score = metrics.f1_score(y_test, pred)
    print("f1-score:   %0.3f" % score)

    if hasattr(clf, 'coef_'):
#         print("dimensionality: %d" % clf.coef_.shape[1])
        print("density: %f" % density(clf.coef_))

        if opts.print_top10 and feature_names is not None:
            print("top 10 keywords per class:")
            for i, category in enumerate(categories):
                top10 = np.argsort(clf.coef_[i])[-10:]
                print(trim("%s: %s"
                      % (category, " ".join(feature_names[top10]))))
        print()

    if opts.print_report:
        print("classification report:")
        print(metrics.classification_report(y_test, pred,
                                            target_names=categories))

    if opts.print_cm:
        print("confusion matrix:")
        print(metrics.confusion_matrix(y_test, pred))

    print()
    clf_descr = str(clf).split('(')[0]
    return clf_descr, score, train_time, test_time
```

```
results = []
for clf, name in (
        (RidgeClassifier(tol=1e-2, solver="lsqr"), "Ridge Classifier"),
        (Perceptron(n_iter=50), "Perceptron"),
        (PassiveAggressiveClassifier(n_iter=50), "Passive-Aggressive"),
        (KNeighborsClassifier(n_neighbors=10), "kNN")):
    print('=' * 80)
    print(name)
    results.append(benchmark(clf))

for penalty in ["l2", "l1"]:
    print('=' * 80)
    print("%s penalty" % penalty.upper())
    # Train Liblinear model
    results.append(benchmark(LinearSVC(loss='l2', penalty=penalty,
                                       dual=False, tol=1e-3)))

    # Train SGD model
    results.append(benchmark(SGDClassifier(alpha=.0001, n_iter=50,
                                           penalty=penalty)))

# Train SGD with Elastic Net penalty
print('=' * 80)
print("Elastic-Net penalty")
results.append(benchmark(SGDClassifier(alpha=.0001, n_iter=50,
                                       penalty="elasticnet")))

# Train NearestCentroid without threshold
print('=' * 80)
print("NearestCentroid (aka Rocchio classifier)")
```

```
results.append(benchmark(NearestCentroid()))

# Train sparse Naive Bayes classifiers
print('=' * 80)
print("Naive Bayes")
results.append(benchmark(MultinomialNB(alpha=.01)))
results.append(benchmark(BernoulliNB(alpha=.01)))


class L1LinearSVC(LinearSVC):

    def fit(self, X, y):
        # The smaller C, the stronger the regularization.
        # The more regularization, the more sparsity.
        self.transformer_ = LinearSVC(penalty="l1",
                                      dual=False, tol=1e-3)
        X = self.transformer_.fit_transform(X, y)
        return LinearSVC.fit(self, X, y)

    def predict(self, X):
        X = self.transformer_.transform(X)
        return LinearSVC.predict(self, X)

print('=' * 80)
print("LinearSVC with L1-based feature selection")
results.append(benchmark(L1LinearSVC()))


# make some plots

indices = np.arange(len(results))
```

```
results = [[x[i] for x in results] for i in range(4)]

clf_names, score, training_time, test_time = results
training_time = np.array(training_time) / np.max(training_time)
test_time = np.array(test_time) / np.max(test_time)

pl.title("Score")
pl.barh(indices, score, .2, label="score", color='r')
pl.barh(indices + .3, training_time, .2, label="training time", color='g')
pl.barh(indices + .6, test_time, .2, label="test time", color='b')
pl.yticks(())
pl.legend(loc='best')
pl.subplots_adjust(left=.25)

for i, c in zip(indices, clf_names):
    pl.text(-.3, i, c)

pl.show()
```

B.1.8 randomselect.py

```
#select random sample of incidents for balancing
#randomselect.py

import os
import random

#open text file containing items that aren't deaths
#and severes from the training set
listfile = open('list.txt', 'rb')

#load items (incident files)in text file as a list
allitems = [row for row in listfile]
```

```
mylist = allitems #call it mylist

rand_smpl = [ mylist[i] for i in sorted(random.sample(xrange(len(mylist)), 51)
#select a random sample of 51 incidents from the list
#(because we're balancing with incidents that are deaths and severes and
#there are 51 deaths and severes in the training set)

#open a file to write our random selection of 51 incidents to
f = open('randlist.txt', 'w')
for item in rand_smpl:
     print>>f, item

f.close()
```

B.1.9 incidentclassify.py

```
#incident severity classifier
#incidenclassify.py



import sklearn
from sklearn import *

from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
class LemmaTokenizer(object):
    def __init__(self):
         self.wnl = WordNetLemmatizer()
    def __call__(self, doc):
         return [self.wnl.lemmatize(t) for t in word_tokenize(doc)]
```

```
#load incidents from folder called container which contains incidents
#in appropriate category subfolders into an sklearn "bunch"
computer_incidents_training = sklearn.datasets.load_files("training")
computer_incidents_testing = sklearn.datasets.load_files("testing")

#CountVectorizer converts a collection of text documents into a matrix
#of token counts. By default words are tokenized.

from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_counts = count_vect.fit_transform(computer_incidents_training.data)

#term occurances (X_counts) fails to take account of document size.
#Longer documents may have higher average count values than shorter
#ones even though they talk about the same topic.
#To avoid this the number of occurrences in any document is divided
#by the total number of words in that document to give the
#Term Frequency (tf).

from sklearn.feature_extraction.text import TfidfTransformer
tf_transformer = TfidfTransformer(use_idf=False).fit(X_counts)
X_counts_tf = tf_transformer.transform(X_counts)

#a further refinement is to downscale the weighting of words that
#occur in many documents in the corpus and are therefore less
#informative than words only occurring in a small portion of the corpus.
#This is results in the term frequency inverse document frequency
#measure (tf-idf).

tfidf_transformer = TfidfTransformer()
X_counts_tfidf = tf_transformer.fit_transform(X_counts)
#print X_counts_tfidf.shape for error checking,
```

```
#prints number of samples and number of features

print X_counts_tfidf.shape

#ready train up a classifier and evaluate performance :-)

from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(X_counts_tfidf, computer_incidents_training.target)

#test case with an incident
docs_new = ["""Main A / E Radioogy Viewing Area Computers are soooo slow ,
and getting slowing throughout the day . Impacting on patient through - put
and accurate recording of patient exam times ."""]

X_new_counts = count_vect.transform(docs_new)
X_new_tfidf = tfidf_transformer.fit_transform(X_new_counts)

predicted = clf.predict(X_new_tfidf)

for doc, category in zip(docs_new, predicted):
    print '%r => %s' % (doc, computer_incidents_training.target_names[category

#test case using a pipe
from sklearn.pipeline import Pipeline
text_clf = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', MultinomialNB()),
    ])

_ = text_clf.fit(computer_incidents_training.data, computer_incidents_training
```

```
predicted = _.predict(docs_new)
for doc, category in zip(docs_new, predicted):
        print '%r => %s' % (doc, computer_incidents_training.target_names[cate

import numpy as np

docs_test = computer_incidents_testing.data
predicted = text_clf.predict(docs_test)
print np.mean(predicted == computer_incidents_testing.target)
#prints accuracy of the model

from sklearn import metrics
print metrics.classification_report(
    computer_incidents_testing.target, predicted,
    target_names = computer_incidents_testing.target_names)
#prints precision, f1 and support of the model



from sklearn.linear_model import SGDClassifier
text_clf = Pipeline([
     ('vect', CountVectorizer()),
     ('tfidf', TfidfTransformer()),
     ('clf', SGDClassifier(loss='hinge', penalty='l2',
                           alpha=1e-3, n_iter=5)),
])
_ = text_clf.fit(computer_incidents_training.data, computer_incidents_training
predicted = text_clf.predict(docs_test)
print np.mean(predicted == computer_incidents_testing.target)
#prints accuracy of the model

from sklearn import metrics
```

```
print metrics.classification_report(
    computer_incidents_testing.target, predicted,
    target_names = computer_incidents_testing.target_names)
#prints precision, f1 and support of the model


print metrics.confusion_matrix(computer_incidents_testing.target, predicted)

#optimize parameters from grid search
from sklearn.linear_model import SGDClassifier
text_clf = Pipeline([
     ('vect', CountVectorizer(max_df=1.0, max_features=None, ngram_range=(1, 2)
     ('tfidf', TfidfTransformer(norm='l1', use_idf=True )),
     ('clf', SGDClassifier(loss='hinge', penalty='l2',
                           alpha=1e-05, n_iter=80)),
])
_ = text_clf.fit(computer_incidents_training.data, computer_incidents_training
predicted = text_clf.predict(docs_test)
print np.mean(predicted == computer_incidents_testing.target)#prints accuracy

from sklearn import metrics
print metrics.classification_report(
    computer_incidents_testing.target, predicted,
    target_names = computer_incidents_testing.target_names)
#prints precision, f1 and support of the model

predicted = text_clf.predict(docs_test)

print np.mean(predicted == computer_incidents_testing.target)
#prints accuracy of the model

print metrics.confusion_matrix(computer_incidents_testing.target, predicted)
```

```
#optimize parameters from grid search
from sklearn.linear_model import SGDClassifier
text_clf = Pipeline([
     ('vect', CountVectorizer(max_df=1.0, max_features=None, ngram_range=(1, 2)
     ('tfidf', TfidfTransformer(norm='l1', use_idf=True )),
     ('clf', SGDClassifier(loss='hinge', penalty='l2',
                           alpha=1e-05, n_iter=80)),
])
_ = text_clf.fit(computer_incidents_training.data, computer_incidents_training
predicted = text_clf.predict(docs_test)
print np.mean(predicted == computer_incidents_testing.target)
#prints accuracy of the model

from sklearn import metrics
print metrics.classification_report(computer_incidents_testing.target, predicte
target_names = computer_incidents_testing.target_names)
#prints precision, f1 and support of the model

predicted = text_clf.predict(docs_test)

print np.mean(predicted == computer_incidents_testing.target)
#prints accuracy of the model

print metrics.confusion_matrix(computer_incidents_testing.target, predicted)

#optimize parameters from grid search
from sklearn.linear_model import SGDClassifier
text_clf = Pipeline([
     ('vect', CountVectorizer(max_df=1.0, max_features=None, ngram_range=(1, 2)
     ('tfidf', TfidfTransformer(norm='l1', use_idf=True )),
     ('clf', SGDClassifier(loss='hinge', penalty='l2',
```

```
                                alpha=1e-05, n_iter=80, class_weight='auto')),
])
_ = text_clf.fit(computer_incidents_training.data, computer_incidents_training
predicted = text_clf.predict(docs_test)
print np.mean(predicted == computer_incidents_testing.target)
#prints accuracy of the model

from sklearn import metrics
print metrics.classification_report(computer_incidents_testing.target, predict
target_names = computer_incidents_testing.target_names)
#prints precision, f1 and support of the model

predicted = text_clf.predict(docs_test)

print np.mean(predicted == computer_incidents_testing.target)
#prints accuracy of the model

print metrics.confusion_matrix(computer_incidents_testing.target, predicted)
```

B.1.10 gridsearch.py

```
#grid search example adapted to run on computer problem data
#gridsearch.py

"""
============================================================
Sample pipeline for text feature extraction and evaluation
============================================================

The dataset used in this example is the 20 newsgroups dataset which will be
automatically downloaded and then cached and reused for the document
classification example.
```

```
You can adjust the number of categories by giving there name to the dataset
loader or setting them to None to get the 20 of them.

Here is a sample output of a run on a quad-core machine::

  Loading 20 newsgroups dataset for categories:
  ['alt.atheism', 'talk.religion.misc']
  1427 documents
  2 categories

  Performing grid search...
  pipeline: ['vect', 'tfidf', 'clf']
  parameters:
  {'clf__alpha': (1.0000000000000001e-05, 9.9999999999999995e-07),
   'clf__n_iter': (10, 50, 80),
   'clf__penalty': ('l2', 'elasticnet'),
   'tfidf__use_idf': (True, False),
   'vect__max_n': (1, 2),
   'vect__max_df': (0.5, 0.75, 1.0),
   'vect__max_features': (None, 5000, 10000, 50000)}
  done in 1737.030s

  Best score: 0.940
  Best parameters set:
      clf__alpha: 9.9999999999999995e-07
      clf__n_iter: 50
      clf__penalty: 'elasticnet'
      tfidf__use_idf: True
      vect__max_n: 2
      vect__max_df: 0.75
      vect__max_features: 50000
```

```
"""
print __doc__

# Author: Olivier Grisel <olivier.grisel@ensta.org>
#         Peter Prettenhofer <peter.prettenhofer@gmail.com>
#         Mathieu Blondel <mathieu@mblondel.org>
# License: Simplified BSD

from pprint import pprint
from time import time
import logging
import sklearn

from sklearn.datasets import fetch_20newsgroups
from sklearn.datasets import load_files #allow to use my data
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.linear_model import SGDClassifier
from sklearn.grid_search import GridSearchCV
from sklearn.pipeline import Pipeline

# Display progress logs on stdout
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(levelname)s %(message)s')


###############################################################################
# Load some categories from the training set
categories = [
    'alt.atheism',
    'talk.religion.misc',
]
```

```
# Uncomment the following to do the analysis on all the categories
#categories = None

print "Loading 20 newsgroups dataset for categories:"
print categories

#data = fetch_20newsgroups(subset='train', categories=categories)
data = sklearn.datasets.load_files("training") #make read my data

print "%d documents" % len(data.filenames)
print "%d categories" % len(data.target_names)
print

################################################################################
# define a pipeline combining a text feature extractor with a simple
# classifier
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', SGDClassifier()),
])

parameters = {
    # uncommenting more parameters will give better exploring power but will
    # increase processing time in a combinatorial way
    'vect__max_df': (0.5, 0.75, 1.0),
    'vect__max_features': (None, 5000, 10000, 50000),
    'vect__ngram_range': ((1, 1), (1, 2)),  # unigrams or bigrams
    'tfidf__use_idf': (True, False),
    'tfidf__norm': ('l1', 'l2'),
    'clf__alpha': (0.00001, 0.000001),
    'clf__penalty': ('l2', 'elasticnet'),
```

```
    'clf__n_iter': (10, 50, 80),
}

if __name__ == "__main__":
    # multiprocessing requires the fork to happen in a __main__ protected
    # block

    # find the best parameters for both the feature extraction and the
    # classifier
    grid_search = GridSearchCV(pipeline, parameters, n_jobs=-1, verbose=1)

    print "Performing grid search..."
    print "pipeline:", [name for name, _ in pipeline.steps]
    print "parameters:"
    pprint(parameters)
    t0 = time()
    grid_search.fit(data.data, data.target)
    print "done in %0.3fs" % (time() - t0)
    print

    print "Best score: %0.3f" % grid_search.best_score_
    print "Best parameters set:"
    best_parameters = grid_search.best_estimator_.get_params()
    for param_name in sorted(parameters.keys()):
        print "\t%s: %r" % (param_name, best_parameters[param_name])
```

B.1.11 Python brewery script

```
 #audits csv file for field completion
 #brewery_script.py

import brewery
from sys import argv
```

```
script, inputfilename = argv

b = brewery.create_builder()
b.csv_source("%s" % inputfilename)
b.audit()

b.pretty_printer()

b.stream.run()
```

Bibliography

1 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

2 Wikipedia. http://en.wikipedia.org/wiki/software_bug, 5 2013. URL `http://en.wikipedia.org/wiki/Software_bug`.

3 Joyce C Niland, Tracey Stiller, Jennifer Neat, Adina Londrc, Dina Johnson, and Susan Pannoni. Improving patient safety via automated laboratory-based adverse event grading. *J Am Med Inform Assoc*, 19(1):111–115, 2012. doi: 10.1136/amiajnl-2011-000513. URL `http://dx.doi.org/10.1136/amiajnl-2011-000513`.

4 M.S. Ong, F. Magrabi, and E. Coiera. Automated identification of extreme-risk events in clinical incident reports. *Journal of the American Medical Informatics Association*, 2012.

5 CMO. An organization with a memory. *Department of Health*, 1:1, 2000. URL `http://www.dh.gov.uk/prod_consum_dh/groups/dh_digitalassets/@dh/@en/documents/digitalasset/dh_4065086.pdf`.

6 CMO. Building a safer nhs for patients, implementing an organization with a memory. *Department of Health*, 1:1, 2001. URL `http://www.dh.gov.uk/prod_consum_dh/groups/dh_digitalassets/documents/digitalasset/dh_098565.pdf`.

7 T. Stephenson. The national patient safety agency. *Archives of disease in childhood*, 90(3):226–228, 2005.

8 Report an incident. URL `http://www.npsa.nhs.uk/pleaseask/experience/reportanincidentcontentid4618/`.

9 A. F. Smith and R. P. Mahajan. National critical incident reporting: improving patient safety. *Br J Anaesth*, 103(5):623–625, Nov 2009. doi: 10.1093/bja/aep273. URL `http://dx.doi.org/10.1093/bja/aep273`.

10 D.J. Hand, H. Mannila, and P. Smyth. *Principles of data mining*. The MIT press, 2001.

11 U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

12 Jake Vanderplas. General concepts. github, Sep 2012. URL `https://github.com/astroML/sklearn_tutorial/blob/master/doc/general_concepts.rst`.

13 Ken Lang. URL `http://qwone.com/~jason/20Newsgroups/`.

14 DMW Powers. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

15 Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 3121–3124. IEEE, 2010.

16 Maureen Baker and Stuart Harrison. Lessons from connecting for health, March 2011. URL `http://www2.warwick.ac.uk/fac/med/staff/sujan/research/safety_case_review/wp3_workshop/baker_harrison_scr.pdf`.

17 Committee on Patient Safety and Health Information Technology; Institute of Medicine. *Health IT and Patient Safety: Building Safer Systems for Better*

Care. The National Academies Press, 2012. ISBN 9780309221122. URL `http://www.nap.edu/openbook.php?record_id=13269`.

18 WHO. Who draft guidelines for adverse event reporting and learning systems. URL `http://www.who.int/entity/patientsafety/events/05/Reporting_Guidelines.pdf`.

19 William Runciman, Peter Hibbert, Richard Thomson, Tjerk Van Der Schaaf, Heather Sherman, and Pierre Lewalle. Towards an international classification for patient safety: key concepts and terms. *International Journal for Quality in Health Care*, 21(1):18–26, 2009.

20 C. Billings, RI Cook, DD Woods, et al. Incident reporting systems in medicine and experience with the aviation safety reporting system. In *Report from a NPSF Workshop on Assembling the Scientific Basis for Progress on Patient Safety. Chicago, IL: American Medical Association. Retrieved June*, volume 5, page 2005, 1998.

21 Steven D Williams and Darren M Ashcroft. Medication errors: how reliable are the severity ratings reported to the national reporting and learning system? *Int J Qual Health Care*, 21(5):316–320, Oct 2009. doi: 10.1093/intqhc/mzp034. URL `http://dx.doi.org/10.1093/intqhc/mzp034`.

22 Denise M Hynes, Ruth A Perrin, Steven Rappaport, Joanne M Stevens, and John G Demakis. Informatics resources to support health care quality improvement in the veterans health administration. *J Am Med Inform Assoc*, 11(5):344–350, 2004. doi: 10.1197/jamia.M1548. URL `http://dx.doi.org/10.1197/jamia.M1548`.

23 David W Bates and Atul A Gawande. Improving safety with information technology. *N Engl J Med*, 348(25):2526–2534, Jun 2003. doi: 10.1056/NEJMsa020847. URL `http://dx.doi.org/10.1056/NEJMsa020847`.

24 M.I. Harrison, R. Koppel, and S. Bar-Lev. Unintended consequences of information technologies in health care–an interactive sociotechnical analysis. *Journal of the American Medical Informatics Association*, 14(5):542–549, 2007.

25 J.S. Ash, D.F. Sittig, E.G. Poon, K. Guappone, E. Campbell, and R.H. Dykstra. The extent and importance of unintended consequences related to computerized provider order entry. *Journal of the American Medical Informatics Association*, 14(4):415–423, 2007.

26 Emily M Campbell, Dean F Sittig, Joan S Ash, Kenneth P Guappone, and Richard H Dykstra. Types of unintended consequences related to computerized provider order entry. *J Am Med Inform Assoc*, 13(5):547–556, 2006. doi: 10.1197/jamia.M2042. URL `http://dx.doi.org/10.1197/jamia.M2042`.

27 Y.Y. Han, J.A. Carcillo, S.T. Venkataraman, R.S.B. Clark, R.S. Watson, T.C. Nguyen, H. Bayir, and R.A. Orr. Unexpected increased mortality after implementation of a commercially sold computerized physician order entry system. *Pediatrics*, 116(6):1506–1512, 2005.

28 CW Johnson. Why did that happen? exploring the proliferation of barely usable software in healthcare systems. *Quality and Safety in Health care*, 15 (suppl 1):i76–i81, 2006.

29 G. Caldwell. Logging in and logging out: patient safety on ward rounds. *British Journal of Healthcare Management*, 17(11):547, 2011.

30 Gregory P T Scott, Priya Shah, Jeremy C Wyatt, Boikanyo Makubate, and Frank W Cross. Making electronic prescribing alerts more effective: scenario-based experimental study in junior doctors. *J Am Med Inform Assoc*, 18 (6):789–798, 2011. doi: 10.1136/amiajnl-2011-000199. URL `http://dx.doi.org/10.1136/amiajnl-2011-000199`.

31 S. Hoffman and A. Podgurski. Meaningful use and certification of health information technology: What about safety? *The Journal of Law, Medicine & Ethics*, 39:77–80, 2011.

32 K.G. Shojania. The frustrating case of incident-reporting systems. *Quality and Safety in Health Care*, 17(6):400–402, 2008.

33 L.L. Leape, T.A. Brennan, N. Laird, A.G. Lawthers, A.R. Localio, B.A. Barnes, L. Hebert, J.P. Newhouse, P.C. Weiler, and H. Hiatt. The nature

of adverse events in hospitalized patients. *New England Journal of Medicine*, 324(6):377–384, 1991.

34 CJ Cassidy, A. Smith, and J. Arnot-Smith. Critical incident reports concerning anaesthetic equipment: analysis of the uk national reporting and learning system (nrls) data from 2006–2008*. *Anaesthesia*, 2011.

35 S.S. Panesar, A. Carson-Stevens, B.S. Mann, M. Bhandari, and R. Madhok. Mortality as an indicator of patient safety in orthopaedics: lessons from qualitative analysis of a database of medical errors. *BMC Musculoskeletal Disorders*, 13(1):93, 2012.

36 J. Arnot-Smith and AF Smith. Patient safety incidents involving neuromuscular blockade: analysis of the uk national reporting and learning system data from 2006 to 2008. *Anaesthesia*, 65(11):1106–1113, 2010.

37 J. Bentham and D.J. Hand. Data mining from a patient safety database: the lessons learned. *Data Mining and Knowledge Discovery*, 24:195–217, 2012.

38 R. Harpaz, W. DuMouchel, N. H. Shah, D. Madigan, P. Ryan, and C. Friedman. Novel data-mining methodologies for adverse drug event discovery and analysis. *Clin Pharmacol Ther*, 91(6):1010–1021, Jun 2012. doi: 10.1038/clpt.2012.50. URL `http://dx.doi.org/10.1038/clpt.2012.50`.

39 F. Magrabi, M.S. Ong, W. Runciman, and E. Coiera. An analysis of computer-related patient safety incidents to inform the development of a classification. *Journal of the American Medical Informatics Association*, 17(6):663–670, 2010.

40 F. Magrabi, M.S. Ong, W. Runciman, and E. Coiera. Using fda reports to inform a classification for health information technology safety problems. *Journal of the American Medical Informatics Association*, 19(1):45–53, 2012.

41 A.B.A. Sari, T.A. Sheldon, A. Cracknell, and A. Turnbull. Sensitivity of routine system for reporting patient safety incidents in an nhs hospital: retrospective patient case note review. *BMJ*, 334(7584):79, 2007.

42 M.S. Ong, F. Magrabi, and E. Coiera. Automated categorisation of clinical incident reports using statistical text classification. *Quality and Safety in Health Care*, 19(6):1–7, 2010.

43 D. Huynh and S. Mazzocchi. Google refine. URL `http://code.google.com/p/google-refine/`.

44 Stefan Urbanek. URL `http://packages.python.org/brewery/`.

45 Apache solr 3.6.0. URL `http://lucene.apache.org/solr`.

46 S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'Reilly Media, 2009.

47 C. Vincent. Incident reporting and patient safety. *BMJ*, 334(7584):51–51, 2007.

48 P.J. Pronovost, L.L. Morlock, J.B. Sexton, M.R. Miller, C.G. Holzmueller, D.A. Thompson, L.H. Lubomski, and A.W. Wu. Improving the value of patient safety reporting systems. *Advances in patient safety: new directions and alternative approaches*, 1, 2011.

49 R. Koppel. Monitoring and evaluating the use of electronic health records. *JAMA: The Journal of the American Medical Association*, 303(19):1918–1918, 2010.

50 D.F. Sittig and D.C. Classen. Safe electronic health record use requires a comprehensive monitoring and evaluation framework. *JAMA: the journal of the American Medical Association*, 303(5):450–451, 2010.

51 J.M. Walker, P. Carayon, N. Leveson, R.A. Paulus, J. Tooker, H. Chin, A. Bothe Jr, and W.F. Stewart. Ehr safety: the way forward to safe and effective systems. *Journal of the American Medical Informatics Association*, 15(3):272–277, 2008.

52 C. Huckvale, J. Car, M. Akiyama, S. Jaafar, T. Khoja, A.B. Khalid, A. Sheikh, and A. Majeed. Information technology for patient safety. *Quality and Safety in Health Care*, 19(Suppl 2):i25–i33, 2010.

53 Malavika Govindan, Aricca D Van Citters, Eugene C Nelson, Jane Kelly-Cummings, and Gautham Suresh. Automated detection of harm in healthcare with information technology: a systematic review. *Qual Saf Health Care*, 19 (5):e11, Oct 2010. doi: 10.1136/qshc.2009.033027. URL `http://dx.doi.org/10.1136/qshc.2009.033027`.

54 G. Singal and P. Currier. How can we best use electronic data to find and treat the critically ill?*. *Critical Care Medicine*, 40(7):2242–2243, 2012.