Aberystwyth University

Understanding Crimes of the Past - a Machine Learning Look into 19th Century News

Author:
Diana Silvia Teodorescu
(dst1@aber.ac.uk)

Supervisor:
Dr. Amanda Clare
(afc@aber.ac.uk)

This report is submitted as partial fulfilment of a BEng degree in Software Engineering (G600)

Version: 1.0 (Release)
May 6, 2014

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature ....

Date ....

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature ....

Date ....

Acknowledgements

I would like to take the opportunity to express my profound gratitude to my mentor and supervisor Amanda Clare, who guided me and encouraged me all the way, always cheering me forward and being as excited about the research and the possibilities as much as I was. Special thanks to the National Library of Wales, for the valuable data and information provided through their Newspaper digitisation program without whom I wouldn't have an exciting and challenging dissertation subject.

I will be forever grateful to my parents for their constant encouragement and support, without them I wouldn't be graduating from amazing university.

Last but not least I would like to thank my great friend Will for working beside me, and nudging me when I got distracted and my patient boyfriend Ed who has been there each time I was panicking about an issue to pick me up an push me forward. Sometimes they believed in me more than I did (and as a plus they were amazing rubber ducks).

Abstract

The following work will present the research and development of a machine learning system that can be used for the classification of articles into set domains. The project will be an analysis of the data collection provided from the historical newspapers found at the National Library of Wales. The newspapers are digitised and have a basic API that will be used to search, request and extract specific data.

The aim of the project is to build a system that can process, download and manipulate article text from this API and develop an algorithm to separate only the crime specific articles and then split them into different classes of crime: murder, theft, fraud, no crime, assault.

The paper will present from start to finish the stages through which the system has been built.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

The National Library of Wales, also referred as NLW, has a rich collection of newspapers and periodicals available in physical form. In their effort to create an easier way to both research and keep intact these magnificent records of everyday knowledge, the Library has started the Welsh Newspaper On-line digitisation project [29].

The newspapers were digitised through preparation and scanning in a specialist studio and went through a post processing stage using Optical Character Recognition software also known as OCR. The OCR is used to take the scanned images of the articles and convert them into computer readable text, stored into a database. This final process made each publication's content searchable and separated into different articles [30]. Due to the condition and quality of the papers, the data obtained varies in accuracy. At this moment, the digitised collection has reached approximately 650,000 articles from a period of over 100 years (1804 - 1919).

1.1.1 Project choice

Newspaper articles in the past haven't been made available in clear structured topics and this project is seeking to achieve a better understanding of how crime was portrayed in the 19th century news.

In January 2013, the Library made the on-line collection and it's API available to Aberystwyth University for research purposes. This opportunity has been taken by the university and given as a dissertation topic to the final year students. The scope of the project is vast and the theme of the research was intentionally left open offering the choice of topic up to the student.

There were multiple wide tasks to choose from including: improving the API, improving the OCR accuracy using image processing, cleaning the text from unnecessary characters and misspellings, article analysis and documentation.

Due to a keen interest in research and text analysis, the choice was made to analyse and document the occurrence of events and to classify specific data. During the initial stage there were considered different options and ideas about how the data could be used and what trends could be observed across the publications from the 19th Century.

The following are a few examples of possible research projects considered:

Advertisements in Wales' history - an analysis of all advertisements; What type of products or services are more advertised? How did this change in time? What products or services are mentioned in news articles?

Scientists in Wales - discover mentions of scientists; Why are they mentioned? What type of science? Where they being describe in a positive or negative way? What words were most frequently used to describe them?

Crime Stories from the past- a classification and analysis of crime articles; What are the different types of crime? Can the articles be classified depending on specific words? Is the page number important? Are thereperiods of time that contain more murders or robberies?

Regarding negative and positive articles, it can be discussed that sentiment analysis is a relatively new concept in natural language processing. Gathering a sentiment or a feeling from a written text is an easy task for a human unless the text is written formally. Newspapers and news articles follow the same style of formal writing that doesn't give up anything about what the actual feeling of the author is and doesn't convey a personal opinion, it portrays exact facts with no emotion attached to them.

The articles in the collection contain news and events that extend over a variety of domains such as: science, sport, politics, crime, and advertisements. These are not classified as the news is today in sections depending on theme and there is no evident connection between the type of article and the page it's found on.

The final decision was made taking into consideration not only the technical complexity and final knowledge but also how attractive and interesting the results might be to the general public. Separating only the crime specific articles and developing an algorithm that can split them into different types of crime: theft, murder, fraud, bodily harm(assault), no crime, can result in conclusions that many can be shocked but also intrigued by.

1.1.2 Machine learning

Machine learning is the ability of a computer to learn from patterns and improve prediction and behaviour through data.

Text classification is a very well studied problem and several methods have been used to solve it. These methods can be applied to news articles as long as there is a well prepared and labelled data set.

There are different types of classification problems that require separating examples into given set of categories. Studies have been made regarding problems such as: text classification(spam filters, natural language processing), machine vision(face recognition, character recognition), market segmentation (predicting customer response), bio-informatics (separating cells depending to their function).

Text classification becomes more and more important especially in the days of big data. The volume of data available is increasing at extraordinary speed and it comes in all types of formats and structures. On most news websites, the articles are labelled and introduced into categories with human input but with text from scanned books or newspapers the amount of data that would need to be analysed is far too large for manual labelling. This is why machine-learning techniques have been used and improved during the years within the natural language processing area.

Within machine learning there are different types of tasks [37] that can present the methods through which a machine can learn. The main ones are:

- Supervised learning - a process that happens in stages and requires a training data set that is labelled; the machine can make connections about the similarity between the variables within each class and learn to make future classifications. This system requires training and testing sets. The test set is an important part of the learning process as the machine will evaluate its knowledge by comparing it to the test data.
- Unsupervised learning - a process that doesn't use labelled data sets and is based on clustering and density estimation. This type of learning doesn't have an evaluation method.
- Reinforcement learning - is different from the other types of learning methods because the machine isn't given specific rules or commands, the learning is done entirely using trial and error.

This project will use general classification to split the articles topic into separate types of crimes. For this, a number of categories have been considered such as: murder, robbery, fraud, assault.

1.1.3 Natural language processing

Natural language processing is the field of artificial intelligence concerned with the interaction between computers and human language. There is a continuous flow of research undertaken regarding artificial intelligence and it seems that natural language processing is still far from being perfect. As with any machine learning algorithms, understanding human language in all its complexity is a difficult goal not only because of its diversity and multiple word meanings, but also because the human language is continuously changing and evolving.

The different branches of natural language processing use tools and techniques to undergo tasks such as: part of speech (POS), base form transformation (lemmatisation), word sense disambiguation, morphological segmentation. To give the best result in such tasks it is important to have a clear problem setting, a well defined corpora on which the task can be evaluated. A corpora is a large set of well structured text, a collection of representative sentences or words. This can be a list of dictionary words from a single language, a parallel representation of different type of words (multilingual, synonyms) or an annotated representation (for classification of parts of speech).

Collecting information about the way the language was 100 years ago, between 1804 - 1919, by reading through the digitised newspapers, made it clear that words and sentences are more formal and vague. The end system will evaluate each text article using machine learning techniques. Other than that the project will use natural language processing tools for the extraction of the data patterns and attributes used in the final machine learning stage.

1.1.4 Relevant literature and existing systems

The most well known news classification system is so obvious that it could be overlooked. It is Google's news service that classifies news across particular domains: sports, crime, politics, entertainment. But the search platform doesn't only utilise complex categorisation techniques. It also takes into consideration the article's credibility depending on the source it came from, in the case of this project this is not a problem as the articles are from real printed newspapers. Google News uses, what it calls, "grouping technology" to classify and group similar news items.

The stories are chosen and the page is updated without human intervention. Google crawls news sources constantly, and uses real-time ranking algorithms to determine which stories are the most important at the moment [10]. They are utilizing clustering techniques using a centroid (the center of a cluster of related words) of keywords that are representative to a specific domain. As a system, it takes into consideration not only techniques such as TF-IDF(term frequency - inverse document frequency) but it is also aware of entity extraction to be able to cluster news centred around a specific location [9].

A similar project as the one presented in this thesis has been attempted with web based expert medical forums [1]. The goal was creating a semi-automated system that would help classifying requests on the Internet forum using a combination of different text mining techniques. The process through which this was undertaken had a few different stages starting from labelling a set of 988 requests into more that 38 categories, calculating statistics regarding the association of each word with each category and applying training regression models. The system was obtaining for any request the probability of belonging to each of the main categories. The result was a way of classifying and analysing the unstructured information to help human experts to better separate and handle the amount of requests when giving feedback.

News analysis is widely studied using different tactics. One of the interesting systems that has been researched was Lydia - a System for Large-Scale News Analysis [21] built with the purpose to recognize important entities and structure in news article depending on them. These entities could be people, places, companies, products.

The main tasks that were undertaken in this project were: gathering the juxtaposition (the words that are side by side) of entities and how were they relating to each other, spatial analysis for observing the sphere of influence of the entity in a specific location depending on the newspaper circulation, observing trends using temporal analysis. The techniques used were mostly probability and statistical algorithms combined with the extraction of entities using Name Entity Recognition and Synonym Set Identification.

1.2 Objectives

The main goal of the project is to design and develop an system which is capable of processing and separating only the crime specific articles and then developing an algorithm that can split them into different types of crime: theft, murder, fraud, assault, no crime. The "no crime" class has been added because the queries could result in articles that can refer to a crime but not as an offence or criminal event or activity but as the name of a play or a statistic report.

The goal is to be achieved by building an automatic system that can take information from the digitized database and classify them just by using a few commands on the command line. The resulting model can then be tested and applied on the other articles using the same data processing. The purpose is to end up with a fully automated system that can run easily. Also such system if built autonomous can be re-utilised to classify other types of articles within the same collection(sport, politics) as long as the labelled data for new classes is available.

After applying the model on all the articles the results will be displayed to the user on a website through charts and info-graphics. The objective of this final part of the project is to showcase different statistics and connections between places, time, page of the article and the type of crime article that would portray the most talked about crimes and the their fluctuation in time.

1.2.1 Research questions

The project seeks to address a few questions regarding both the specific publication structure and the idea of crime in the news between the years 1804 - 1919. The answers to these questions will be determined at different stages of the project: during the research of the data, during the manually labelling or after the system has been built and can offer a graphical representation. The questions are outlined here and will be discussed in detail together with conclusions in the Evaluation chapter.

1. What machine learning algorithm would showcase a better classification of the articles?
2. What types of crime should be classified? Are there classes that should be added to the list of categories?
3. What are the words that better define a crime article? What about the ones to define theft or fraud?
4. Were the articles in a specific part of the newspaper, on a specific page?
5. Were specific locations more prone to crime activity?
6. Did crime drop or spike during the years?

Chapter 2

Development process

2.1 Development methodology

With machine learning and research software, most of the work goes into selecting the best possible data. That means trying, rethinking, processing and change should be welcomed. Due of the nature of the project and the uncertainty of the data analysis results, to be restricted to a specific design that is set from the beginning would be a wrong choice. Regular tests, feature planning and simple design are the points that can improve the work undertaken.

Evolutionary design it's meant to use coding as the base of decision making along with the process of development. Design is considered an important part of work but it doesn't need to be completed before the coding starts, it evolves in the same time as the program takes shape, it's respondent to changes and is flexible [24].

2.1.1 Project planning

The development methodology approach had many similarities to agile development methodologies and more exactly to Feature Driven Development due to the flexibility that it offers.

Feature driven development can provide the amount of freedom needed when the entire requirements are unknown from the beginning while still offering a structure to the development.

The project was started on the base of evolutionary design and taking into account some of the XP practices such as constant refactoring and simple design to make sure that the code is easy to understand in case changes need to be done further ahead. The "You Ain't Gonna Needed It (YAGNI)" approach has been considered during the implementation stage and this meant the structure of the system and it's features were built step by step without adding specific code until it was actually needed. This brings up a constant need for refactoring to make place for the new features.

Taking into account the FDD best practices there was developed an initial object model that was been assessed with the project supervisor. This explained the main objective of the project and took into consideration the risks involved to make sure the expectations are in accord with the time and resources available. The initial object model can be reviewed in the Appendix A.

2.1.2 The client side

The FDD methodology has been picked over XP due to the fact that a customer is not as important in the planning and development stage as it is in Extreme programming. In this case the customer can either be considered the National library or the supervisor.

The National Library holds the entire data and can provide information regarding the API and the other resources available. They might be interested in the end product but they won't be involved the project progress and don't have specific requirements regarding the execution, except for their need to represent the data in an exciting way that would attract people to use it.

During the course of the project there were a few discussions with the National Library of Wales regarding their API and the topic and goal of the project. They were most helpful with answering questions about the API and offering full access to the collection server from any network and computer, providing developers own login details. NLW also had some face to face interaction with the developer due to their interest in the outcome of the system. NLW made clear that they want to follow up close the progress and even have plans regarding the future of the system. In this regard they can be perceived as a client that would be interested in improving and continuing the developers work digitisation project.

2.1.3 Agile with no team

The feature development steps had to be moulded on what resources and roles are available in this case. Because this was a one person team the plan was kept as a bullet point list into a notebook. With coding errors or bugs these were rethought by using pseudo-code on paper and with the help of the Bug reporting feature of GitHub. Also providing regular builds will help verifying the system and also spot any integration errors early giving a better chance to fix them.

The problem with choosing feature driven development as with other agile methods is the need of a team. The fact that it provides individual code ownership is a plus over XP in this case as the developer will be the only person that owns the code. Regarding progress reporting a way to accomplish this is to constantly test my code and refactor it and request assessments on the supervisor weekly meetings.

The XP refactoring principle was constantly acknowledged as the code structure was checked and modified regularly and this eliminated the issue that comes up with ad-hoc design decisions. Also using "rubber duck" debugging proved invaluable when dealing with errors and with building complicated connection and selections from the MySQL database.

2.2 Work organisation

Using the agile way meant that the work had been done in small stages or features that were tested after they were implemented. This methodology has specific charts used to observe the complition of each task. In this case the work load has been divided in daily tasks that were set-up at the start of the day. These were small goals that were either code related or research related.

With a project of these proportions where research, coding and documentation have all a broad part in the final result the work needs to be well organised. Since the first weeks a clear plan has been put in place regarding the working hours: a new working environment has been found in The Orchard, and Apple computer room in the Computer Science department. This was a perfect work place most of the times: quiet, well lit and in the good company of peers, and the work hours were from 10 to 6 every day of the week.

2.2.1 Development tools

The most important consideration regarding work organisation was back-up and repositories. The code and libraries were saved on a private Github [14] repository and so did the Documentation on a separate repository. GitHub is a source control system that allows developers to keep their code safe and structured. It's especially good for agile development because if used properly and committing regularly it stores every change made in the code and allows the user to roll back to previous versions. It was the tool of choice mostly because of its popularity and its simple interface. The advantages are obvious and on the duration of the project the big changes were regularly submitted at the time a new functionality of the system could be roll out.

Using source control and GitHub was a new experience for the author and it required some time of adjustment. The commits were made when features were implemented and when the program could be run resulting in the desired results with no errors. According to GitHub the system can be summarised as 33 commits and over 6.300 lines of code over a period of two month and a half(18th of February to 5th of May) all being saved in a private repository shared with Dr. Amanda Clare, the supervisor of the project.

To make things safer the two GitHub local repositories are set-up in a Dropbox [12] folder on the developer's computer. Dropbox is a free file storage and sharing in the cloud. Its main advantage is the fact that the folders can be accessed from any computer with the user-name and password and it also keeps a history of all the file changes during the period of a month. The project code is actually stored in three different places: Dropbox servers, GitHub and the developer's computer to of which use archiving technologies.

A vital tool in the development process was Eclipse Kepler IDE. The Integrated Development Environment offered the means to develop and debug Java written systems. This is an open source system that can extend its functionality using different plug-ins and connectors. The developer had previous knowledge of using both Java and Eclipse. Eclipse IDE was connected through the Data Source Explorer and the JDBC driver to the MySQL Workbench which is another very helpful tool that was used constantly.

The MySQL Workbench is a very useful user interface for the SQL tables and databases created using the MySQL server. Even if all of the database functionality and structure was created through Java the workbench was particularly useful while testing the correctness of SQL statements before implementing them into code. This tool was also used to visualise the tables and their records.

Chapter 3

Design

3.1 Design overview

The National Library of Wales is the only entity that holds the digitised newspapers dating from 1804 - 1919. In contrast to how newspapers are structured and written now, these publications don't have a known structure divided by topics or page numbers. It would be highly likely that a language barrier exists between the past and the present for a computer to be able to process. Because of this, using a data set found on-line to create a model for the classification could end up with an increased margin of error. The stages of finding, processing and manually labelling the crime articles to set-up a training set would be the first task to be tackled during this project.

For this purpose, the amount of processing the data needs to undertake and the data structures required, are taken in consideration. One of the initial decisions with this task was regarding the amount of data that should be used. It was chosen to extract the articles that contain specific crime related words such as: police, investigation, arrested and crime. This was a decision that resulted from the realisation that the time needed to download and process the entire collection would have been over the time that could be spent on this stage.

3.1.1 Language choice

An important programming language that is used for natural language processing and machine learning that was briefly researched is Python. This has multiple libraries and resources that can complete multiple tasks within this domain. It could be thought that Python is a better choice when dealing with natural language processing as it's libraries touch a wider set of subjects and there are solid optimization and matrix libraries which are helpful. This has been also been observed from the large amount of links that Google offered when researching machine learning.

Another strong choice that is mentioned in journals and papers was the Java programming language. The reason for this is the Weka software that deals with machine learning using a user interface, with APIs that offer access to a variety of algorithms. In the end, this was the language of choice due to the knowledge and previous experience that the author has with it. Taking into consideration the new technologies and ideas that would be involved in the progress of this research, a familiar language was strongly desired.

Other languages that are worth mentioning are Matlab for machine learning and R for statistics.

3.1.2 System stages

The data will undergo different processes during its flow through the entire system starting from the National Library of Wales, going through a cleaning and pre-processing stage, being downloaded locally, labelled and classified and finally analysed and displayed to the user. This following image reveals the end design for the system. To review an earlier version visit the appendix A.1.

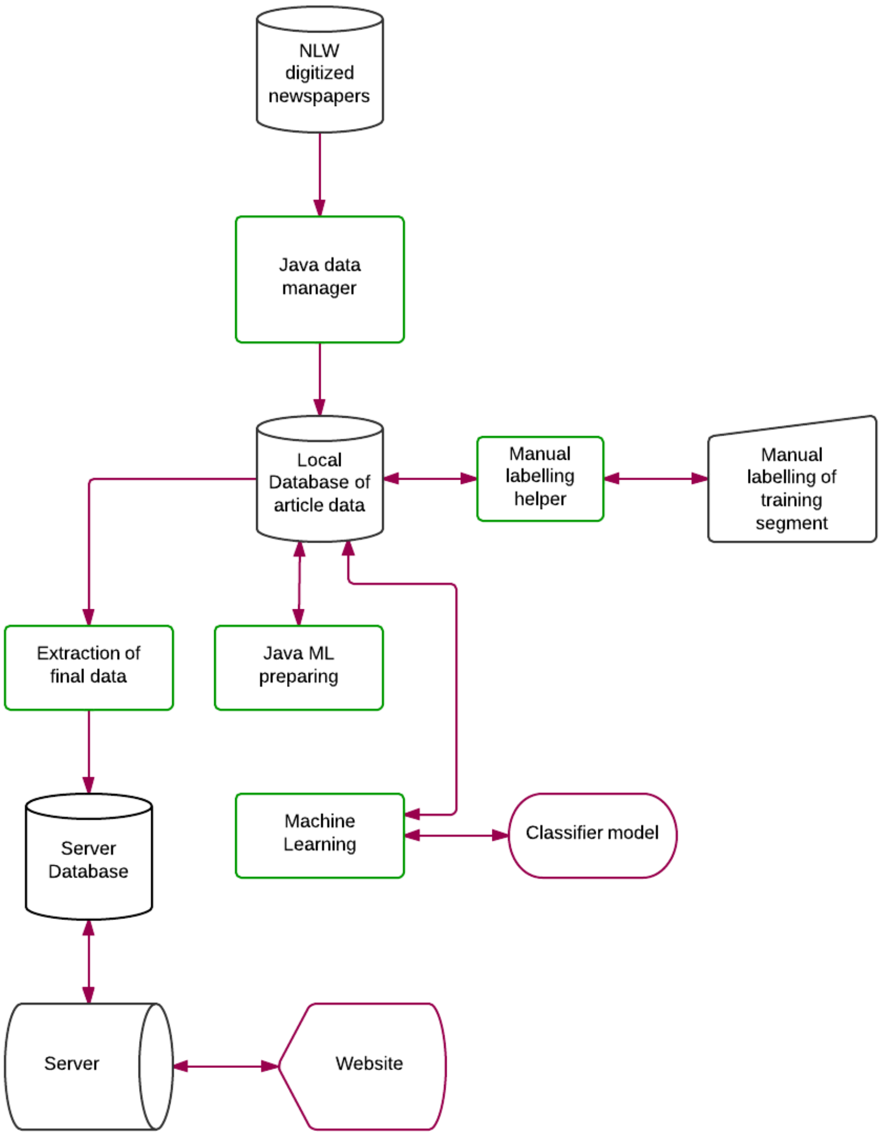


Figure 3.1: The flow of data within the system

The Figure 3.1 describes the entire process from start to finish. The data is accessed and queried from the *National Library Digitised Newspapers* Server, `http://hacathon.llgc.org.uk`, and sent to the *Java data manager.* The latter is a Java based process that will parse the XML response from the server and collect the data using an article object. Each article object will be saved locally into a *Database of article data* building a base structure for the machine learning attributes.

For the classification of each article into their domain, the training and testing data will go through a *Labelling Helper* constructed in Java and will then undergo the *Manual labelling of training segment* process. The results of this process will be saved in the database ending with a number of articles being tagged with a label corresponding to the class they are apart of. The labelled records and the machine learning selected attributes from each article will be joined into new tables during the *Java ML preparing* stage and then put through the *Machine Learning* algorithm that will result in the development of the *Classifier Model.* Once a model is in place the rest of the articles can be automatically tagged with the appropriate labels and saved in the *Local Database of article data.*

The final section of the data flow is related to the output of the results on a website that can be viewed by external users. During the *Extraction of data* stage, parts of the findings will be processed using percentages and statistics and sent to a *Server Database* where they will be displayed on a user friendly *website.*

Regarding the techniques used in each stage the work can be split into different sections:

1. Data processing and cleaning
2. Data storage
3. Manual Labelling
4. Machine Learning algorithms and set-up
5. Front end presentation of findings.

3.2 Text analysis and classification

For the purpose of this project there were initially five main crime classes chosen:

1. **Murder** - an article that describes a murder, the act of taking someone's life
2. **Theft** - an article that describes a robbery, the act of stealing money or goods from another person
3. **Assault(bodily harm)** - an article that describes a fight, attack or an altercation resulting in one or multiple people being injured
4. **Fraud** - an article that describes an act of earning money or goods using false pretences, or impersonation;
5. **No crime** - an article that doesn't refer to any type of real event relating crime: a screen play, a conference, a meeting, politics;

Articles can result in being part of multiple of multiple categories. e.g. A robbery that has as result the loss of human life would be considered apart of both murder and robbery classes. The type of machine learning technique used will be supervised learning and that will mean using training and testing data that will be labelled to teach and evaluate the classification rules.

For the classification of these domains it was decided an initial separation of the articles by extracting and downloading locally only text that contains crime related words. These decision was made due to the immense amount of data present on the National Library of Wales server.

An important goal of the code is to require as little customisation and editing as possible so that the system can be applied not only to crime articles. The end result will be a set of processes that will work together to classify any type of articles that come in an XML format of the type described. The only real time constraint with applying this to other type of articles will be the process of forming a new training data set by manually labelling and adding new classes.

3.2.1 Machine learning

Machine learning models are developed using training and testing sets. In this case the training and testing data for the model development had to be manually labelled.

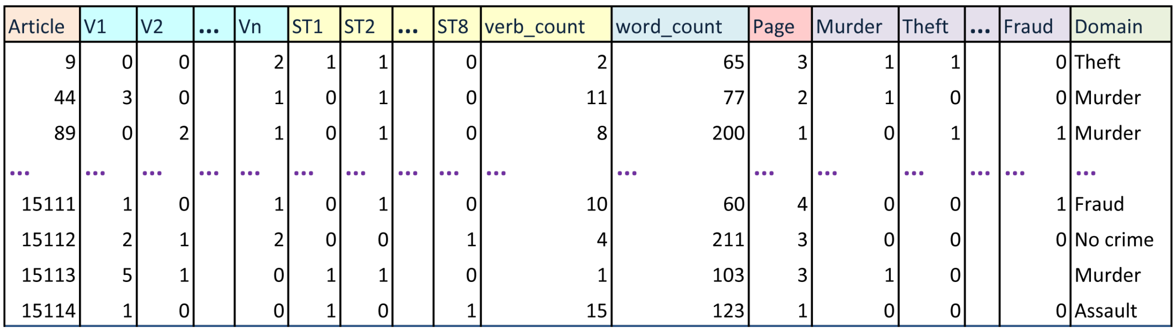
The final classification will be made not only using a one type classification but also having the option to classify an article as multiple classes.

Machine learning is a process that happens in stages and requires a training data set that it can make connections about the commune variables for each class and learn to make future classifications.

Building a machine learning system has two main stages: the data processing and selecting and the results stage. The most important part of it is the attribute selection - choosing the right attributes.

Regarding the machine learning algorithms it has been decided that the patterns will be learned by observing attributes as: the page number, the number of words in the article the verbs and nouns and their occurrence in each article; Each article varies in readability, clarity, number of words. The verbs that will be selected as attributes will be only the verbs extracted from the manually labelled articles. These will become columns names in the table that will be fed into the machine learning algorithm. Resulting in a comparison between these and the articles that haven't been labelled.

An initial example of a machine learning table has been developed to better comprehend it's structure and how to join and prepare the tables in the implementation stage. This table can be seen in Figure 3.2



Article	V1	V2	...	Vn	ST1	ST2	...	ST8	verb_count	word_count	Page	Murder	Theft	...	Fraud	Domain
9	0	0		2	1	1		0	2	65	3	1	1		0	Theft
44	3	0		1	0	1		0	11	77	2	1	0		0	Murder
89	0	2		1	0	1		0	8	200	1	0	1		1	Murder
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15111	1	0		1	0	1		0	10	60	4	0	0		1	Fraud
15112	2	1		2	0	0		1	4	211	3	0	0		0	No crime
15113	5	1		0	1	1		0	1	103	3	1	0			Murder
15114	1	0		0	1	0		1	15	123	1	0	0		0	Assault

Figure 3.2: An initial representation of the design considered for the Machine Learning Table

There will be two machine learning tables for each type of classification: one for the trained(labelled) data and one for testing. Also because the system will be using two different approaches: one taking in consideration verbs and one nouns these will be separated and compared in the testing and results stage. It will be interesting to notice weather the algorithms chosen would be able to evaluate the data from one part of speech more accurate then the other.

Incorporating human input is a crucial part of the machine learning process when dealing with natural language processing. To have an accurate model a large set of training and testing data will be needed. This training data will be manually labelled with the help of an labelling program coded by the author. The plan is to use this as an extra project that will show an article text from the local database and give the option to label them using the ID number of the class. The training set chosen is set to be of approximate 500 articles, at least 100 for each of the classes.

The type of analysis that will be applied will use a number representation for each attribute, and to achieve the multiple classification the machine learning table will have two different views and classification type:

1. The attributes and articles will have one type classification column in text form, representing the exact word of each of the 5 classes;
2. The attributes and articles will have the classification columns in nominal form, using the classes as columns and using 0 or 1 to represent weather the article belongs to the specified class or not.

The machine learning stage is a process through which the training data passes through an algorithm, or classifier in this case, and results in a learned classification rule or model [19]. The rule discovered is tested using testing data at this point, and the accuracy of the system can be observed. There will be a comparison between the labels of the testing data and the labels predicted by the model. If the accuracy is high and no changes need to be applied to the attributes, the model can finally be applied to the unclassified data. A representation of this system can be observed in Figure 3.3

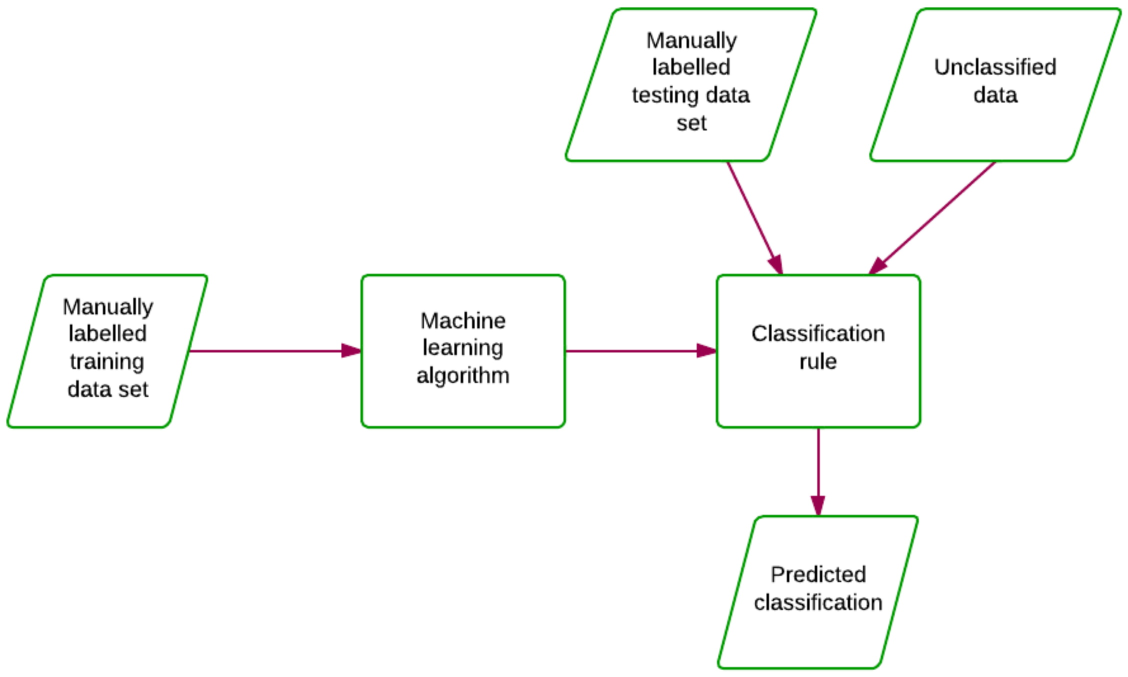

Manually labelled testing data set
Unclassified data
Manually labelled training data set
Machine learning algorithm
Classification rule
Predicted classification

Figure 3.3: Graphic representation of the data flow in the Machine Learning process

Weka API was chosen as the tool used for the implementation of the Machine Learning Algorithms. This workbench is very well documented, having its own book [2]. It is also easy to understand and can be used either with its own Graphical Interface or just its library within Java, accepting data from both ".arff" file format and a database table. Because the data in this case will be in different tables inside a relational database the decision to use joint tables to create the format necessary was clear. It was also decided to input 70% of the labelled article into the training data and 30% into the testing data.

Algorithms taken into consideration at this point were the ones originating from the Weka API. During the research stage it has been observed that machine learning can classify articles using a set of different algorithms [34]. These can represent the data sets using different techniques and views and built a path towards understanding the attributes and how they connect together to obtain an accurate classification.

Decision Trees

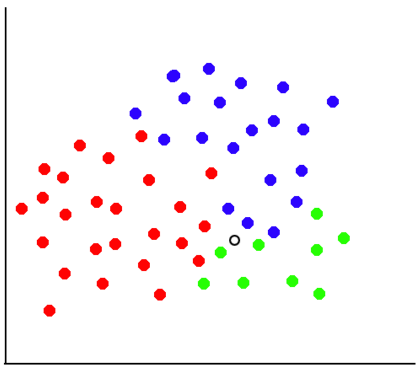
Decision tree learning [42] uses prediction algorithms to map specific conclusions about an item and the observations that help getting to those conclusions. It is called a decision tree because of its representation. The leaves are the class labels and the branches are the features and attributes that lead to those labels. Each branch will hold a value of the attribute. The decision tree will gather information about how the attributes of a specific class connect together to get to that class, the nodes will test the attributes. The type of decision trees that are used to analyse the relationship to a specific class are called classification trees. Decision trees are easy to understand and built and can handle both numerical and categorical data.

Naive Bayes classifier

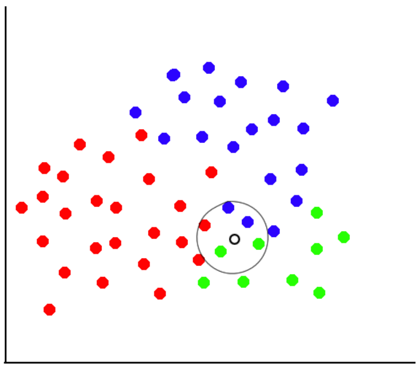
The Bayes theory [38] is a difficult one to grasp and remember by humans due to its mathematical formulae but it is the default algorithm of general-purpose because it is easy to implement. What this type of classifier does is it uses probability to predict the membership to a specific class. In first instance it takes into consideration the prior probability of the next article to be tagged with a specific label, so if there are more article classed as murder and theft in the training data then the probability that the next ones will be apart of these classes is bigger. Secondly the algorithm chooses to look at the likely hood to be classified as a specific class by taking into consideration the articles that are already labelled and are in the vicinity. Lastly the Bayes classifier uses the so-called Bayes' rule that calculates the final posterior probability.
Example:

Labelled articles:
Murder - 25 (red)
Theft - 20 (blue)
No crime - 9 (green)

Prior probability for new article to be tagged as:
(number of murder labelled divided by the total number of labelled articles)
murder = 25/54
theft = 20/54
no crime = 9/54



(a) Representation of labelled articles



(b) The likely-hood circle surrounding the new unlabelled article

Figure 3.4: Graphic representation of the Bayes Classifier algorithm

Likely-hood for new article to be tagged as:
(number of murder labelled in the vicinity of new unlabelled article divided by the total number of murder labelled articles)
murder = 1/25
theft = 2/20
no crime = 2/9

The final posterior probability for the unlabelled article to be:
(multiplying the prior probability with the likely-hood)
murder = 25/54 x 1/25 = 0.0185185
theft = 20/54 x 2/20 = 0.0370
no crime = 9/54 x 2/9 = 0.0370
In this type of algorithm the final decision is made depending on the highest posterior probability.

SVM(Support Vector Machine) Classifier

SVM classifiers [20] have a good performance within multiple domains: text, bioinformatics, image recognition. [16] This algorithm builds an *hyperplane* that is built with the cope of regression, classification or other tasks. To obtain the smallest margin of error this *hyperplane* should be placed at the furthest distance from any of the classified articles. SVM is using Kernel Functions to help with the performance. This type of classifier is more complex than the previous two and has a wider area of options depending on the type of attributes and labelled records.

In the end it is important to aim towards a highly accurate prediction on the test data provided.

3.2.2 Natural language processing libraries

Two of the important attributes chosen to create the ML model were the occurrence of verbs and the occurrence of nouns. These will have to be extracted from each sentence using Part of Speech Tagger.

POS tagger is a library that analysis each word in a sentence in its morphological environment and on its own and assigns a grammatical tag to each of them. The verbs and nouns would be then taken into consideration as part of the list of attributes, to make this easier for the machine learning to process and diminish the margin of error the choice has been made to lemmatise each of the words. To find a words lemma is to find its general base form. This is particularly useful when using verbs as they can have different forms and tenses that can increase the margin of ML error when having such a variety of attributes in the training set that in the end mean the same thing.

There were three main Libraries considered for this type of problem: Alchemy API, LingPipe, Apache NLP and Stanford Libraries. They are all machine learning based libraries that use models built from data sets to offer support for different Natural Language Processing tasks.

Alchemy API [4] was the first natural language processing service researched even before the selection of a clear project objective. It encompasses multiple text analysis techniques such as: entity extraction, text categorisation, sentiment analysis, language detection, text extraction, concept tagging (a total of 11 functions). The service provides meta-data response in multiple formats and offers support in multiple languages. The difficulty with using this was due to the fact that it isn't an open source product and it is protected by copyright.

Another similar product that offered more freedom but still under specific conditions and with different pricing options is **LingPipe** [5]. This library is a tool for processing text using computational linguistic. This specific service has a better set of features from which the Topic Classification and Part-of-Speech Tagging.

The Stanford Natural Language Processing Group has developed a series of NLP software available freely under the GNU General Public Licence. The Stanford Libraries [41] are developed fully in Java and are open source. Besides the well written documentation these libraries have multiple tutorials and discussions on forums and blogs as well their own mail member list

Apache OpenNLP [39] was the second contender while choosing the machine learning based tool-kit for processing of natural language text. It is an open source software available under the Apache licence. While it seems like it has a good API the documentation is not regularly updated and the parsers and analysis libraries seem to have less option than the Stanford ones.

The decision was made towards the Stanford Natural Language Processing for it's POS Tagger accuracy, over 97%, and its list of libraries. Both lemmatisation and POS Tagger were important but also the presence of software that can recognise and extract entity names. Working with one group of libraries that can offer all the functionality needed for both the machine learning stage and the analysing and displaying the results under one umbrella was thought to be the best decision for the developer.

3.3 Data processing and cleaning

Data processing and selecting is the first stage of a machine learning system. The collection of articles that will be fed into the machine learning algorithm need to be understood, downloaded and prepared for the classification process.

3.3.1 National Library of Wales API

The National Library of Wales is the only entity that holds the digitised newspapers dating from 1804 - 1919 and in contrast to how newspapers are structured and written now these don't have a known structure divided by topics or page numbers, we can also think of a language barrier between the past and the present for a computer to process. Because of this using a data set found on-line to create a model for the classification could end up with a increased margin of error so finding and manually labelling the crime articles to set-up a training set would be the first stage of the project.

The entire collection has over 650,000 articles from a period of over 100 years. Currently the project is being developed on a test server with only 60,000 articles.

The National Library of Wales API can be accessed via a link through SOLR queries. SOLR is a very powerful open source platform with configurable response formats and complex queries possibility that uses a RESTful API.

The following is a list of fields that are searchable in National Library of Wales API and that would give a better understanding of the type of information the Database holds [28]:

- **PID:** The ID of the issue for an article
- **ArticleID:** The ID of an article, corresponding to CouchDB
- **PublicationPID:** The ID of the Project for an Article
- **IssueDate:**The publication date of an article in the form YYYY-MM-DDTHH:MM:SSZ
- **PagePID:** The Page ID for an Article
- **PublicationTitle:** The title of a Project/Publication
- **ArticleSubtitle:** The subtitle of an Article

- **PageLabel:** The label of a page, e.g., [2]
- **ArticleTitle:** The main title of an Article
- **ArticleAuthor:** The author of an Article
- **ArticleSubject:** The subject of an Article, e.g., Advertising, News etc.
- **ArticleAbstract:** The abstract field of an Article
- **ArticleText:** The main body of the Article text

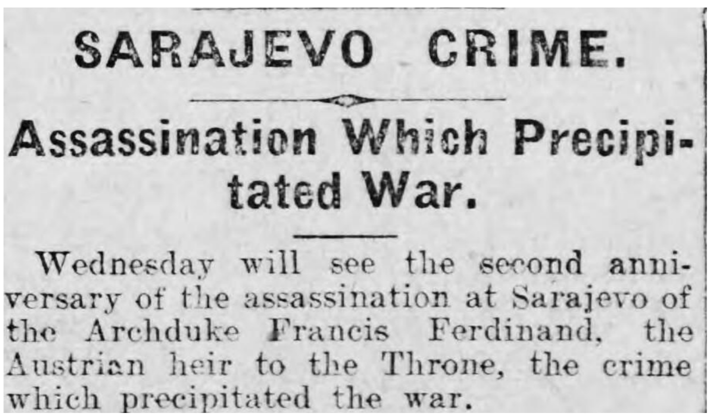
3.3.2 Data limitations

The data was analysed initially by reading some of the articles. As a first impression there was a clear difference between what the scanned newspapers were showing and what the OCR text result was. Some articles were clearly more difficult to read by the software and there were some problems with character recognition resulting in misspellings, strange characters and divided words arising from the scanning process and age and condition of newspaper [6].

The following paragraphs will describe some of the initial data limitations regarding the articles their content and their extraction from the NLW server:

- The API was initial available only from the Computer Science department. The request of a user-name and password was granted and the data was able to be accessed on any device from any network eventually.
- Some articles are written in Welsh.
- The NLW API accessed is on a test server that has only a limited amount of news articles - this wasn't a real problem as the data that will be used will only be a portion of the entire collection on the test server.
- Smudges of ink that are read as punctuation signs
- Words that were hyphenated have been written in the NLW database with the dash and space as can be seen in the following example:

```
Wednesday will see the second anni- ! versary of the assassination
at Sarajevo of ! the Archduke Francis Ferdinand, the ! Austrian heir
to the Throne, the crime !
```


SARAJEVO CRIME.

Assassination Which Precipitated War.

Wednesday will see the second anniversary of the assassination at Sarajevo of the Archduke Francis Ferdinand, the Austrian heir to the Throne, the crime which precipitated the war.

Figure 3.5: The text in the printed article

- Article text that resulted in a line of gibberish from a very long and readable article if viewed in the actual scanned newspaper:

```
v^nejcennam 77, had d 1 jrgjxiVrePorte  i "as '>een appreEenaea
_ui_eonnftctiou wltk-th  crime.
```

- Articles are split in two records with different IDs in the database due to either their occurrence on different columns/ pages or separated by an image.
- The archive contains articles that are repeating due to reprinting of the events in different issues.

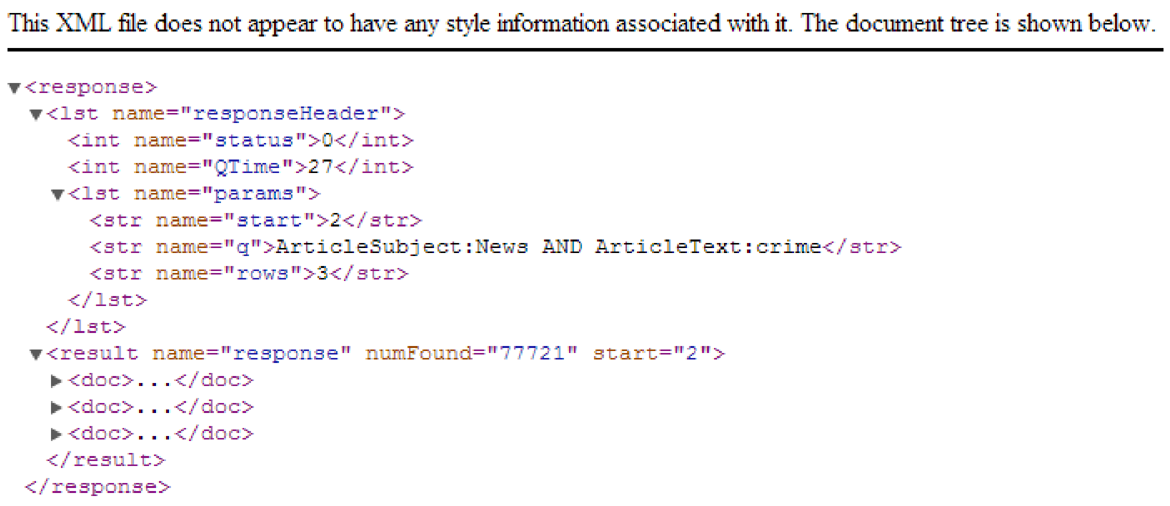
3.3.3 Pre-processing

The data needs to be cleaned and cleared from any recurrent unwanted characters. This will be done using both natural language techniques and text processing techniques within Java either using Regex or the replace methods.

Also the articles will be requested from the database in order to be processed using parsing methods, and separated in article attributes.

XML vs. JSON

A query to the NLW server looks like: `http://hacathon.llgc.org.uk/solr/select/?q=ArticleSubject:News%20AND%20ArticleText:crime&start=2&rows=3` and it implies requiring from the server a number of 3 articles (number of rows) with the subject of News and that contain the word “crime” in their text. The response should start from the second article found(the start parameter). The response to this type of query is the XML file represented in Figure 3.6.


This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<response>
  ▼<lst name="responseHeader">
     <int name="status">0</int>
     <int name="QTime">27</int>
    ▼<lst name="params">
       <str name="start">2</str>
       <str name="q">ArticleSubject:News AND ArticleText:crime</str>
       <str name="rows">3</str>
     </lst>
   </lst>
  ▼<result name="response" numFound="77721" start="2">
    ▶<doc>...</doc>
    ▶<doc>...</doc>
    ▶<doc>...</doc>
   </result>
 </response>
```

Figure 3.6: The query response in XML format in a browser window

Appending &wt=json to the URL would transform the response into a JSON string and the print-screen of the exact query in JSON format can be found in the appendix B.1.

One of the first design decisions to be made was choosing the query response standard format. The choice was made after some research on both options available: XML or JSON.

XML (Extensible Markup Language) is a mark-up language used to store and structure data but not necessarily to display it. The XML response from the SOLR queries was represented between different tags and with the help of significant attributes names such as: PageLable, ArticleTitle, ArticleText, WordCount. An extended version of the XML response can be observed in the appendix B. This will show the content of a response query and the content of the ¡doc¿ tag that organises in sub-tags the entire information about one article: the metadata from the database and the data extracted from the published article and it's issue.

JSON (JavaScript Object Notation) is a data format built for easy reading by both human and computer that is language independent. It is built on two structures: a collection of pairs as name/value and a list of values [18]. The JSON response returned in this case was more difficult to comprehend as its structure was a long line of text with no spaces containing multiple objects within objects delimited by curly brackets.

The decision was made towards XML due to a more extended knowledge of it and because specific parts of it can be accessed depending on the specific tag or attribute name. Getting each article detail into an Article Object from a JSON structure would mean iterating through the entire response each time.

For parsing XML in Java there were considered two main libraries [7]:

1. **DOM Parser/Builder** - Used when needing the entire tree structure available at once and where it's important to access specific tags and attribute names.

2. **SAX Library** - Used for parsing large XML files using a small amount of memory due to the way of downloading information.

The DOM parser was selected because allows more freedom with accessing each and every tag or attribute name and that requests the whole tree. This means that a lot of resources will be used to hold the whole XML tree structure so during the implementation the developer will take into consideration the amount of rows will be requested in one query.

3.3.4 Post-processing

Using the new model will result in the automatic labelling of the rest of the articles. After building and tagging the entire collection of articles the data needs to be edited again and displayed to the user. At this stage the results will be analysed with the purpose of discovering relevant statistics such as:

- Are nouns predicting weather an article is classed as a specific crime better than verbs?
- How does the Issue and page on which the article is found influence it's class.
- Can there be a relevant connection made between the type of crime and the year or season it occurred in using the issue date?
- Were specific locations more prone to crime activity? This final question will require a deeper analysis of the article text and the use of Entity recognition to extract the region the crime has occurred in where these are available. Using just the region where the newspaper is published wouldn't give a clear idea because most of these articles describe facts that occur within the United Kingdom and sometimes even outside its borders.

To answer these questions and present them to an interested user it has been decided to create a second database and an website. These will be set-up on an Centos 6 server. The choice was made towards this server because the developer is familiar with its structure and has one accessible.

The website will be a series of pages that will show different patterns in the data using info-graphics and pie-charts that will be constructed using HTML5, CSS3 and JavaScript. These web technologies were chosen due to their ongoing development, good documentation and their ability to perform more and more complex operations, animations and graphics that will offer an user friendly experience. Also the author has enough experience with them to rapidly develop and launch a site. For a structured and consistent design across the entire website the Bootstrap Framework will be used. This has the advantage of being highly customizable and it's documentation is easy to understand and follow [23].

The database will be an MySQL database just to keep the design of the back-end of the website on the same line as the Java project. PhpMyAdmin will be used for the administration of the database. This is an open source tool used for the management of MySQL databases over the web.

The initial idea for these is to contain labelled article titles and text, publication titles and article IDs. Depending on weather there are any copyright restrictions from the National Library of Wales the website can have different information displayed:

- Only percentages using graphs, pie charts and line charts of: different crimes in time, percentage of a specific crime class compared to others, etc.
- A detailed view of the percentages with specific article text examples on date, domain, page or publication.
- A link to the exact crop of the publication found on the Nation Library of Wales using the CouchDB view. This can be obtained with a request of the image using an URL as `http://hacathon.llgc.org.uk/scifimagehelper/getarticle/` and appending the ArticleID from the SOLR response at the end.

3.4 Data storage

Regarding storing the data the design of the system started from thinking about using HashMaps and arrays initially. Each article can have its ID as a key and the rest of its data as separate elements of an object in the HashMap. This type of organisation of the data can also help with solving the problem with some of the split articles and the duplicate ones.

After processing the data into the right attributes and having for each article the occurrence of the verbs and nouns this will be send to a database that will contain the Machine Learning Tables necessary for applying the classifiers.

The process of testing the SOLR queries in a browser with larger and larger numbers of rows has resulted in the realisation that even if choosing only articles with specific crime related words, this would still be a very large data set. At this point the decision was made to extract only a certain number of articles for each word. The exact sum would be selected, taking into consideration the amount of time the data would take for the needed attributes to be extracted and processed in the right form.

3.4.1 Data storage and data structures

Each article will be split during parsing into the corresponding XML tags within a DOM document. The best way to continue working with each of the information is to use data structures that could enable the developer to modify and compare the content.

The article object

Knowing that the purpose of the API response is to display different aspects of an article the decision has been made towards using a Java Article Object. This will separate all the important segments such as: region, publication, word count, page, abstract etc. and allow editing the data. The object will be created as a separate class and will contain the getters and setters required to set the information to the right parameter of the object.

The array

Array lists were chosen due to the impossibility of knowing the size of any of the lists created. These will be used for gathering data regarding the words in each article and also for developing further tables and machine learning attributes. Because of the variety of articles and the unknown results to POS Tagging and lemmatisation, arrays of verbs and nouns have been created not only to ease the insertion into the future tables but also to help with the count.

The database

When choosing the database there were three open source Relation Database Management System taken into consideration: MySQL, PostgreSQL, SQLite [11]; The most important aspect when comparing these systems was their speed and how well they can deal with a large amount of data. From the start SQLite doesn't scale as well as the others and it is limited in size [36].

Most of the researches show that MySQL is faster in queries compared to PostgreSQL, however the latter is more reliable because is fully ACID (Atomicity, Consistency, Isolation and Durability) compliant [25]. The system will probably be more secure using PostgreSQL, but it will be more scalable with MySQL and with this type of project the database security is not a real issue.

In the end the decisions was made towards to MySQL due to its speed and graphical interface: MySQL Workbench and also because it is also usually used on websites and will have more documentation and plug-ins to optimise the work on-line.

3.4.2 Database design

The local database was designed keeping in mind all the attributes required and the structure of the machine learning table. All tables have been designed to contain an auto increment ID that will help creating joint tables and relationships between them. The following are the tables that the database will contain initially:

- **Article** - containing each of the data that describes an article and that can be also be used later for the statistic study. The table will contain fields such as: title, text, article date, word count, verb count etc.
- **Publication** - containing representative data for the publication such as: name, region, publication ID. This table with be in a one to many relationship with the article table as a publication may contain multiple articles but an article is contained by only one publication.
- **Verb** - containing an unique ID and the name of the verb. This table is needed to keep an exact count of the verbs in each article and also to be used in the process of building the machine learning training set.
- **Noun** - similar to the Verb table it will help with keeping in check the nouns in each article

- **Search Term** - the table will keep record of each search term and it's connection to the article. The search terms will be the words used to request crime related articles from the internet. This table is needed because some articles might be found multiple search queries.
- **Domain** - this will be a list of all the domains discussed and their ID.

Verbs, nouns, domains and search terms were set-up as separate tables because they will have a many to many relationship with an article. This means for example that an article can have multiple occurrence of the same verb and a verb can be in multiple articles. To represent and manage this into a SQL relational database the creation of junction tables was required. In the 3.7 image there is a clear representation of the many to many relationship that will need to be put in place between an Article and a Verb:

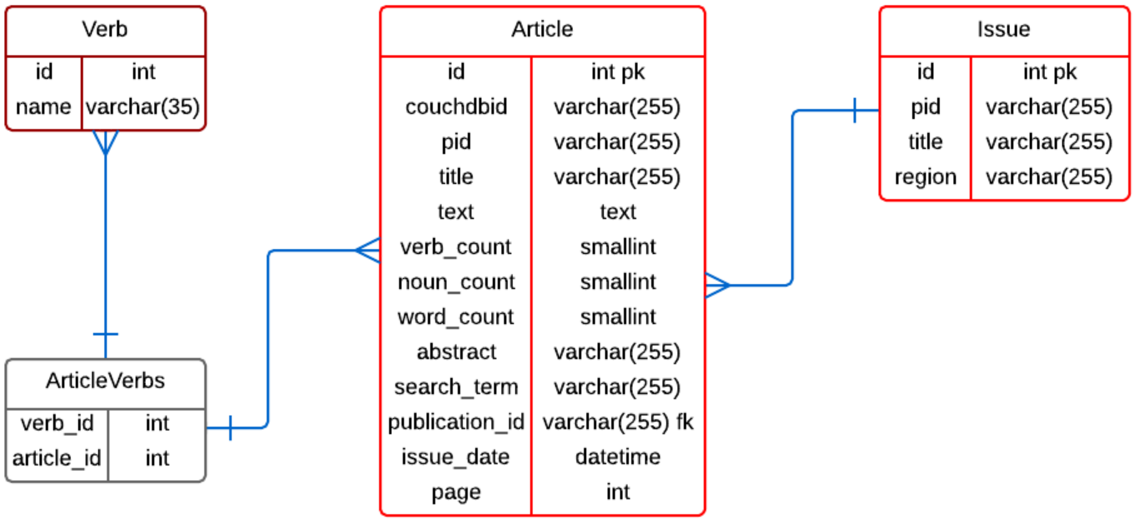


Figure 3.7: A visual description of the tables schema and the one to many and many to many relations

The structure of the database, its design and tables can be fully viewed in the appendix C.1. The database structure discussed and visualised here is the latest structure and the reason to its set-up and table arrangement is the result of an implementation realisation that is discussed in Section 4.1

Chapter 4

Implementation and Results

4.1 Data processing

In the data processing stage there was a lot of fiddling with ideas about removing bad characters and extracting the correct attributes. As with every system that takes data from a different source the first part of the implementation was testing the queries made to the National Library of Wales database. These helped with noticing the exact structure of the XML, the name of the attributes and the response time.

It was important to build an article object that will contain each article one by one with each of its attributes. This object was created with the idea that will be send to a database so each of its elements were data that needed to be a column in the database structure.

4.1.1 Changes in the design

The code was built feature by feature starting with SOLR queries and XML parsing, text cleaning and analysis. At this stage the data called from the National Library API was a set of just 10-20 articles to see the results of the processing easily and tweak any errors or issues that were overlooked.

When a good system was put in place for processing the next step that was taken was testing it on a bigger scale. This brought up an issue regarding the number of requests the API could handle and also how much data could the Java machine hold in memory by bringing up a memory error.

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
```

This was due to the large amount of data that the systems was trying to keep in memory as an array and correcting the Virtual Memory arguments in the Eclipse IDE didn't change the issue. At this point the decision was made to scale the data download down. There was no way both the XML parsing and processing of text to be done in one single task so the data had to be downloaded locally in smaller parts and in stages after each part is processed.

The new design was set-up to process and download the attributes needed one article at a time into a multi relational database. This is then allowing the developer to join the information together in the machine learning stage.

4.1.2 XML parsing

XML parsing has gone through a few modifications since its first attempt. The first lines of code considered were testing and researching the way the DOM library works and were using a few to many `for` loops and `if` arguments that were difficult to follow and more time consuming. In the end it was modified to pick each XML tag and attribute just through one loop and add article objects to array of objects by looking into each tag and extracting the Attribute name.

As with any DOM parsing method the XML is taken as a `String` and a DOM document is being built and building the tree structure with elements as nodes [27].

For example when dealing specifically with the ArticleText element of the XML response there were a few things to take into consideration:

1. Is this the right part of the response? The parser was splitting the response into elements in a doc. These elements had Attributes that were found by using a `String`comparison in Java:

   ```
   if (elem.getAttribute("name").equals("ArticleText"))
   ```

2. Is the text clean? To make sure of that the text was sent through the `replaceCharact` method B.3.2 that was built to replace the unwanted characters and save it as a `String`

   ```
   String article = replaceCharact(elem.getTextContent());
   ```

3. Set the text element of the article object to the article `String`;
4. The article `String` needs then to go through the extraction of machine learning attributes such as verbs and nouns and the lemmatisation process;
5. After the list of verbs and nouns are collected they are set as parts of the Article Object: `setVerbList, setVerbCount, setNounList, setNounCount`;

The part of the code representing these steps can be found in the Appendix B.3.

4.1.3 Data cleaning

After being parsed the data was checked by printing text samples. In the first instance it was noticed a series of strange characters that weren't in the XML response but ended up in the text of the article object. There was a clear issue with the way the XML was returning the text and especially the white space at the end. The idea of an encoding problem was researched and a few different changes in the coding methods have been tried unsuccessful: changing the encoding of the response before the XML is parsed and also changing the encoding for the local XML doc.

The strange characters that appeared after the text processing were the same across all the articles. Due to this observation the choice made was to take these symbols and replace them with an empty `String`.

```
.replace("\u00c2", "").replace("", "")
```

Other text cleaning has been done in concordance with the database building and populating stage where specific characters weren't accepted by the SQL statements. The characters, such as apostrophe and quotations, were escaped using the backslash.

4.1.4 POS Tagger and Lemmatisation

As specified in the Design Chapter for the Part of Speech tagger and lemmatisation stage the Stanford Libraries have been chosen. The way these libraries were built was using a machine learning model that tags the words in a sentence using morphology analysis techniques and an English dictionary. The methods used to extract verbs and nouns and lemmatised them were built in their own class. These were taking the article text as a parameter as a `String` and using the `MaxentTagger` class to request the pre-built model was tagging the part of speech in the entire text [13].

```
MaxentTagger tagger = new MaxentTagger(
"lib/models/english-left3words-distsim.tagger");
```

The result is a `String` similar to:

```
The_DT above_JJ sketch_NN shows_VBZ the_DT position_NN of_IN the_DT boy_NN\\
```

Because both verbs and nouns need tagging the method is using an `if` statement and an option variable. This latter parameter is used to split the function into two parts and it will request two variables then: the text that will be analysed and the result wanted: verb or noun `taggIT(theText, ‘‘verb’’)`.

To end up with the actual required part of speech the result was stripped by using just a few simple replace functions.

```
.replace("_VBN","").replace("_VBZ", "")}
.replace("_NN "," ").replace("_NNS","")}
```

and then the sentence is split into words. To obtain the words wanted a `for` statement select only the ones that don't contain an underscore "_"

```
for (String word : sentence.split(" ")) {

        if (!word.contains("_")) {

            attributeString.append(" " + word);

         }
```

The method will then return a `String` that will contain the verbs (or nouns) separated by spaces.

A `String` has been chosen as the result of this method due to the next stage of the attribute processing which is lemmatization. To find a words lemma is to find its general base form. This is particularly useful when using verbs as they can have different forms and tenses that can increase the margin of Machine Learning error due to a large variety of attributes that in the end have the same meaning in the context.

```
e.g.: be, is, been, was, were
  escape, escaped, escaping
```

The lemmatizing method uses the *Stanford coreNLP Library*, it splits the `String` into sentences and tokens(words) and reduces them to the base form using the `LemmaAnnotation.class`. Another way to achieve this would be to use stemming - a process which would require cutting the end of words (inflectional and sometimes derivational affixes) in the hope of producing similar results.

Before performing the operation on an entire XML response the POS tagger was tested through separate class on one article. After making sure the words are correctly selected the code has been included as a separate class in the project, the class `TaggerWords`.

4.2 Database set-up and populating

The SOLR queries are displayed as an enumeration of articles held into `<doc>` tags. Their order is dependent on relevancy score. This type of score is calculated using the tf.idf model - a formula that takes into consideration the term frequency and the inverse document frequency [15]. This sorting algorithm was useful when facing the decision of downloading only part of articles due to the large amount of time that the process would require. If the sorting was dependent on region, publication, date or other data significant to the research then using only a part of it would have caused issues with the machine learning predictions and with the final results and percentages.

In the next stage of the implementation all the stripped XML response data was inserted into the corresponding fields. During this process other aspects had to be considered: eliminating inputs with the exact same data, updating fields where necessary, retrieving primary keys to be added as foreign keys in the linked tables. Most of these Java methods had to call SQL queries multiple time and check their response.

To access a relational database through Java a connector interface called Java Database Connectivity (JDBC) is needed [31]. The JDBC driver uses an API that builds the protocol that allows transfers between the client and the database [43].

4.2.1 Creating and inserting data

After the connection between the Eclipse IDE and the MySQL was in place using the user-name and password of the MySQL server the initial tables could be created. Each of the tables was built using Statement objects and a String query `CREATE IF NOT EXIST table` that was also containing data about each column name and type and about the relationships that had to be set-up: primary and foreign keys.

The primary keys selected for each table were auto-increment IDs and these were the records used as foreign keys in the connected tables. The decision to use an auto-increment ID instead of the obvious unique keys such: ArticleID and PublicationPID was made to increase the response of the database later on during `SELECT` and `INSERT` queries. Using the unique elements already provided would have overly complicated the structure of the database as they were both `VARCHAR` types of 255 characters.

When the methods and queries were put into place the actual download was done in three stages:

1. Creating the tables - the methods were run and checked by observing the MySQL Workbench.
2. Testing the insert methods - as specified earlier it was important that the methods were tested thoroughly. This has been done a couple of times, testing for errors and adding more Java replace characters methods in place where errors were reported due to SQL syntax issues. These type of issue happened regularly as most of the articles had text containing
3. Inserting the entire data - due to its size this has been done in separate stages, remembering the last row or article downloaded and restarting the download from there the next.

The actual database populating is done by calling the information gotten from the XML parsing method. To make it possible to add records in the publication and article tables by reading through data necessary for the table only once the XML parsing method has two parts that are called through a given option parameter.

For example when inserting data about the Article the method will be called while reading through the XML tags in a for loop:

```
theArticle = printNote(doc.getElementsByTagName("doc").item(i),"article");
```

This method returns an article object that contains only the data related to the article and its attributes: article ID, title, abstract, word count, verb count, verb list etc. Part of the information in this object is then transferred to other tables to keep a better record of the attributes: search term, domain, verb, noun and the other part is inserted in the main `article` table.

However the method presented above has a second part that can be reached only when the end parameter is changed to `"publication"`. In this case the resulted object contains data about the publication that is required to be inserted in the `publication` table. Finally sending these objects to the database is done by calling the different `add` methods, for example `database.addArticleObject(theArticle, db, searchTerm);`

Due to the process of tagging each verb and noun and bringing them to their base form before adding them to the right column and table the process was taking approx. an hour for each 2,000 articles downloaded, depending on broadband speed. The local database of the system downloaded 16,095 records in separate stages over a period of 2-3 days.

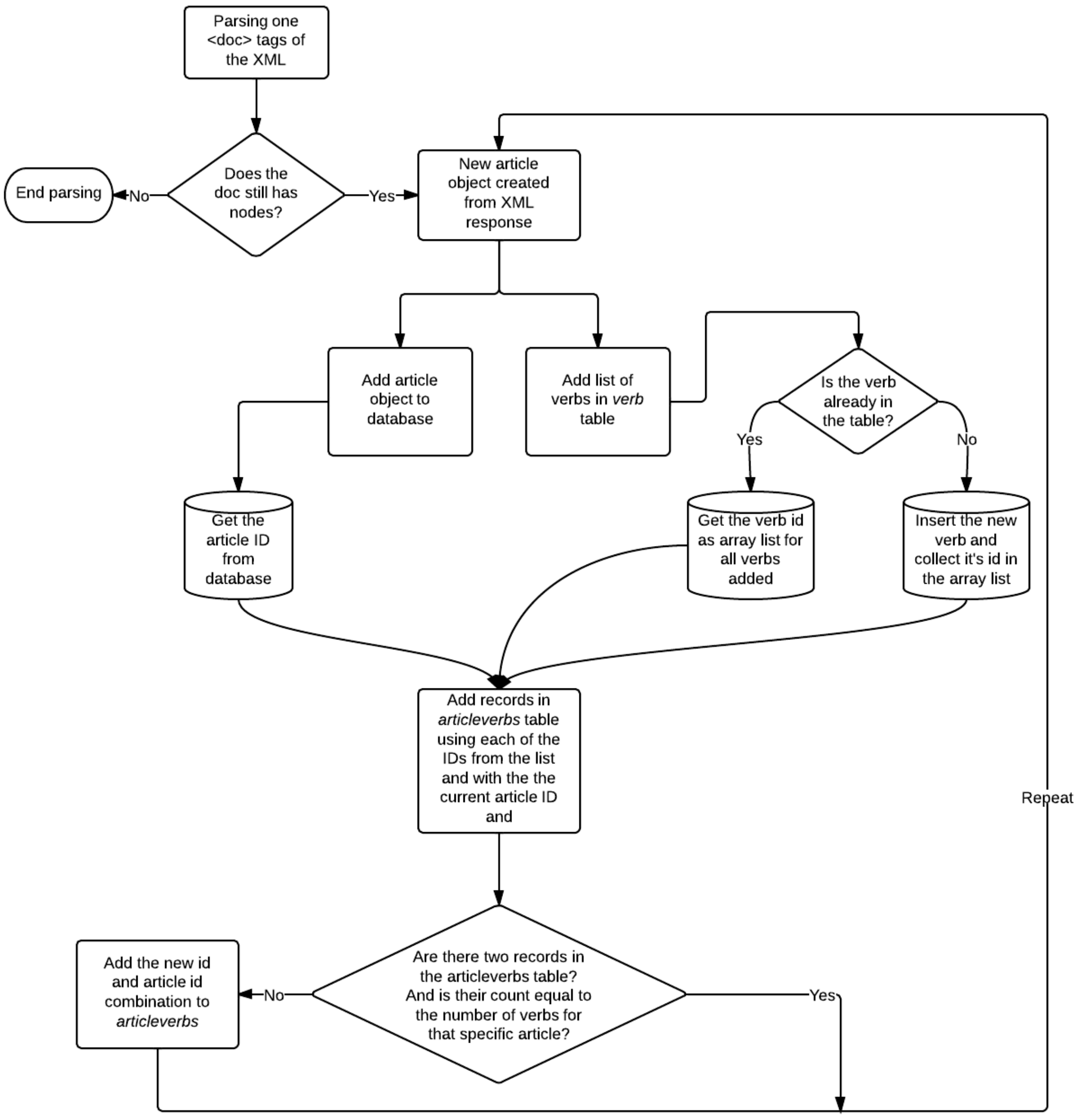
Table relationships

All the attributes tables will connect to the main article table through many to many relationship as per the graphic in the appendix C .

For example this was the case with the `verb` and `noun` tables. To create the many to many relationship between verb and article a third table was built, a junction table called `articleverbs` [26]. Even if the SQL syntax for building this and linking the three tables together is straightforward applying it in Eclipse IDE using Java was a more complex task.

A issue the developer was faced with was the fact that the database was update with one article object at a time, but when dealing with verbs, nouns, search terms and domains these had lists of records that had to be inserted all at once, each in two different tables.

Junction tables were created first containing three columns each, for example: auto-increment id, article_id and verb_id. Also the verb table would be created and populated with verb names taken from the article object array of verbs. Then the junction table was populated with the ID of the article record that the parser was dealing at that moment and the id of the verbs from the verb table. To get the specific IDs of the verbs from the verb table the method that was populating it was returning the IDs of the verbs added into an array. It must be specified that this process was done for each of the article, one by one. These stages are better explained through the visual representation in Figure 4.1



Parsing one <doc> tags of the XML
Does the doc still has nodes?
End parsing
No
Yes
New article object created from XML response
Add article object to database
Add list of verbs in *verb* table
Is the verb already in the table?
Yes
No
Get the article ID from database
Get the verb id as array list for all verbs added
Insert the new verb and collect it's id in the array list
Add records in *articleverbs* table using each of the IDs from the list and with the the current article ID and
Are there two records in the articleverbs table? And is their count equal to the number of verbs for that specific article?
No
Yes
Add the new id and article id combination to *articleverbs*
Repeat

Figure 4.1: The representation of how a many to many relationship is built

To deal with identical articles or records that came from more than one SOLR query the inserting method had as first stage a query that was looking into the table for the specific Article ID or unique record using a `SELECT FROM` statement. If this was returning a result (fact checked in Java using a `ResultSet`) in the table no action would be effectuated. Some of the insert methods had this stage different:

1. Creating the connection between the `publication` table and the `article` table. These tables have a one to many relationship, because a publication can have many articles but an article can have only one publication. Their relationship meant the `publication` table needs to be created first and then its auto increment `id` has to be returned by the method and inserted as a primary key when inserting the related data in the `publication` table.

2. Adding the same article in the `article` table because it exists in the XML response for multiple search terms results in updating only the `article.search_term`

column with the concatenated string of the previous search term and the current one. This was done for better testing that no duplicates were allowed in the database even if they resulted from querying with different search terms.

4.2.2 Machine learning tables

To utilise the Weka API within Java while taking and inserting data from and to the SQL database required utilising the JDBC controller again and creating a connection with the MySQL server. After the connection has been activated the training data table had to be created connecting and comparing and joining multiple tables from the database. The skeleton of this table has been made using a complex SQL statement and separate `SELECT` queries that were bind together through Java.

The basic structure of the table had to contain all the articles in fields, each of them would then have a page number, search term, a word count, a verb count, a noun count. The other columns will have the name of each verb that was presented in the articles that were labelled. The table design has been discussed and can be revised using the Figure 3.2

Building the machine learning tables

The approach to building the structure can be described better by following these steps:

- **First step** - retrieve the list of verbs (nouns) in an array that appear more than 5 times in the table obtained by joining the verb table with the domain table where the two are connected through the article IDs and the joint tables. This means the only the verbs that are from articles that were labelled and that appear more than 5 times in this list will be taken into consideration. This has been done using SQL statements that will join the required tables and get the information. The following statement is the ones used in requiring the nouns from the articles that were mentioned in the `articledomains` table.

  ```
  SELECT COUNT(*), noun.name
  FROM articledatabase.articlenouns AS av
  JOIN articledatabase.articledomains AS ad
  ON ad.article_id=av.article_id
  JOIN articledatabase.noun
  ON av.noun_id=noun.id
  GROUP BY av.noun_id
  HAVING COUNT(*)>= 5"
  ```

  The tables resulted have a very large number of columns. For the training table that will use nouns as attributes there was obtained a number of 607 nouns as columns. In the case of verbs there was a smaller number: 298

- **Second step** - retrieve the search terms of the articles that were labelled in an array.

- **Third step** - create the article table using StringBuilder and the `CREATE table` statement appending each verb from the array as a column name by using a `for` statement: For each verb from the array append part of the SQL statement that will contain the verb name within as seen bellow and in the more detailed example in the appendix C.2.

  ```
  stringBuilder.append("‘"+verbList.get(i)+"‘"+" SMALLINT, ");
  ```

Populating the machine learning table

For populating the machine learning table there was the issue of taking into consideration only the occurrence of the verbs and nouns that are in the articles that have been labelled. The first step was to populate the columns that could be easily retrieve from the article table such as: article_id, word_count, verb_count, noun_count, page. The tables were set-up from the start with `NOT NULL DEFAULT 0` columns to make it easier for the records that didn't have a specific verb or noun occurrence.

These tasks have been accomplished using a `SQL SELECT` and `JOIN` query within an `INSERT INTO` command.

The attempt to count the occurrence of each verb and insert them all at once was unsuccessful and a different approach has been made by inserting each occurrence one by one. The training and testing data were two different tables and the information that would go in them had to be different too so that meant splitting the labelled articles into two sets. This was done using a simple mathematical calculation and two for loops. The articles labelled were split into three quarters representing the records that will be added into the `mlnountraining` and `mlverbtraining` tables. The remaining articles were added to the test tables.

Because part of the columns of the machine learning tables were already populated the occurrences of a particular verb were calculated and inserted using two for loops that were taking the `article_id` data of the machine learning tables and the names of the verbs added as columns and then comparing them to the presence of similar linked records in the junction tables. To do this a join table was made in the `SELECT` statement.

```
String query = "SELECT COUNT(*) AS counts "
+ "FROM articledatabase.verb AS v "
+ "JOIN articledatabase.articleverbs AS av "
+ "ON v.id=av.verb_id "
+ "WHERE v.name='"
+ verbList.get(i)
+ "' AND av.article_id='"
+ articlesID.get(j) + "'";
```

Testing the database

After the machine learning tables were created, the entire system had to undergo some changes. For this a new database was created. To test the sys-

tem with this new database, the only change that was required was changing the name in the `DB_URL` line of code from `static final String DB_URL = ¨jdbc:mysql://localhost:3306/the_test_schema";` to `static final String DB_URL = ¨jdbc:mysql://localhost:3306/articledatabase";` wherever the MySQL connection was required. This was the only change because none of the SQL statements were referencing a specific database name, so if the test one had the exact structure but a different name then the system would still work perfectly and apply all the statements as with the initial database.
Before altering any database tables, copies were made to help with testing the integrity of the system especially after a big alteration to the structure: such as removing the Issue Date column or adding new columns, for example.

```
CREATE TABLE mlmultitraining LIKE mlverbtraining;
INSERT mlmultitraining SELECT mlverbtraining
```

Most of the tests were done using the MySQL Workbench and then implemented in the Java code where and if needed. MySql Workbench was immensely helpful especially while coding the SQL statements into JDBC in Java. It was important to test the SQL statements within an environment that can give a number of rows and a direct answer before implementing the SQL syntax into the Java methods and running the entire program.

4.3 Machine learning implementation

At this stage of the project, the machine learning becomes a priority and most time spent here will be time used for developing the training and testing data set and for evaluating the model results. After all of the data was downloaded locally, there was a short research period where ideas regarding what would be the easiest way to tag all the articles was thought about and also about how the Weka API works with an SQL database and the way the results were displayed [22].

The training data was labelled over the course of 3 to 4 days. The process was uncomplicated and repetitive and after reaching the desired number of articles tagged for each of the domains, the `articledomains` table had 652 labels of which 589 were unique articles.

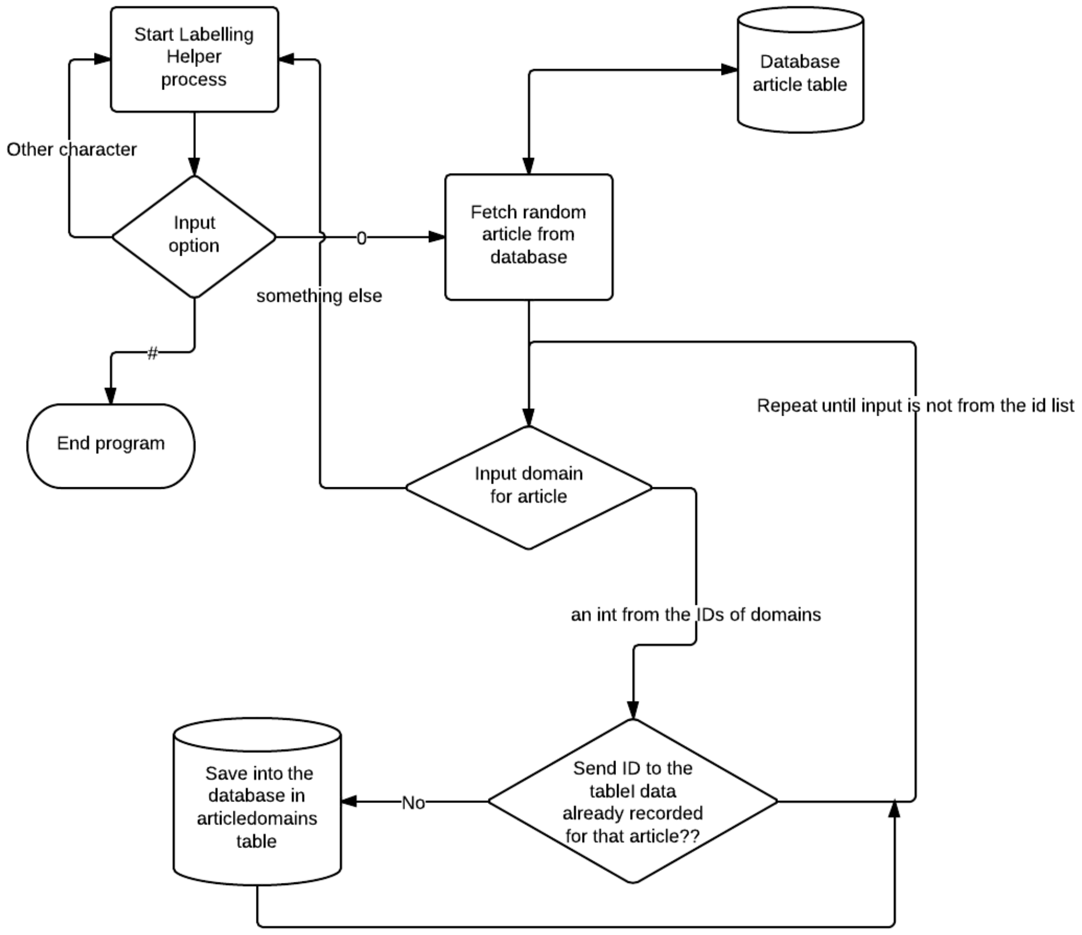
4.3.1 Labelling helper

It was decided that the best way to code a new system that will help with article labelling is to implement it so the developer can use the command window and a few requests to read from keyboard.

The Labelling Helper system was programmed to request a random article with its text, id, abstract and title by typing "0". The random record is selected using an `int randomID = min + (int)(Math.random() * ((max - min) + 1));` that has min = 1 and the max = 16,095 (the number of articles downloaded). Then the program would ask for a domain. The options were to input a number from 1 to 5 and the to require another input in case the text would be considered as being from multiple domains.

Before showing the article, the program would check if this has been labelled already and if so it wouldn't show it but would offer the option to look for another one by pressing "0" again. Everything then goes into the junction table that connects the Article table with the Domain table. Inputting from keyboard the actual ID of the domain instead of the name is making things work faster.

Figure 4.2 explains the system using processes and decision nodes.

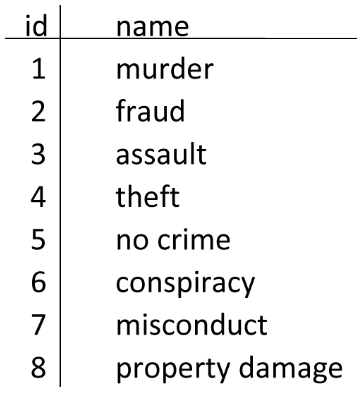


Start Labelling Helper process
Database article table
Other character
Input option
0
Fetch random article from database
something else
#
End program
Repeat until input is not from the id list
Input domain for article
an int from the IDs of domains
Save into the database in articledomains table
No
Send ID to the tablel data already recorded for that article??

Figure 4.2: The flow chart describing the labelling system

Observations

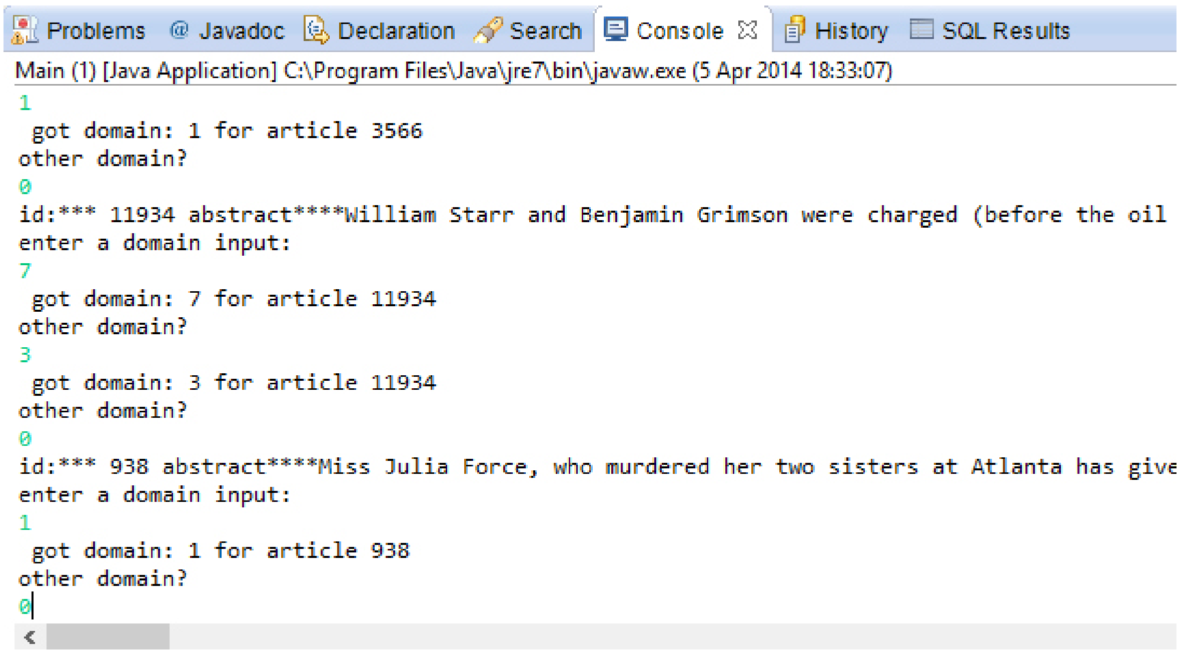
During the first hour of manually labelling the data it has been observed that there are some articles that define other types of crimes that haven't been included into the 5 initial domains. Due to this it has been decided to add three more classes: conspiracy, misconduct, property damage; This meant a few changes in the code was made both for the creation of the domain tables and the Labelling Helper program. The numbers that could now be given as a label can be observed as ID of the classes in the domain table:



id	name
1	murder
2	fraud
3	assault
4	theft
5	no crime
6	conspiracy
7	misconduct
8	property damage

Figure 4.3: The classes and their IDs as represented in the domain table

An example print-screen of the `LabellingHelper` view at run time can be observed in the Figure 4.4



```
Main (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (5 Apr 2014 18:33:07)
1
 got domain: 1 for article 3566
other domain?
0
id:*** 11934 abstract****William Starr and Benjamin Grimson were charged (before the oil
enter a domain input:
7
 got domain: 7 for article 11934
other domain?
3
 got domain: 3 for article 11934
other domain?
0
id:*** 938 abstract****Miss Julia Force, who murdered her two sisters at Atlanta has give
enter a domain input:
1
 got domain: 1 for article 938
other domain?
0
```

Figure 4.4: Example of the user input in the Labelling Helper program

In the image it can be noticed that an input such as 1, 7, 3 was saved as a label in the database, as each number is representing one of the 8 domains. Also 0 was giving a new random article with its ID, abstract, title and text.

It was also observed that there was a fear of discussing intimate subjects. Crime was at that point in time defined as different acts; There are the known acts of murdering, stealing, embezzling but many articles were describing specific crimes that are now considered less aggravating or terrible such as: bigamy, crime against religion, drunkenness, disorderliness;

4.3.2 Weka implementation

For its API to work, Weka requires either an ".arff" type document or a table that has its data structured in the right way. The entire text analysis and information about each article has been arranged correspondingly in a separate table that can be queried for columns containing the machine learning attributes. Before using the Weka API with a SQL database, there are a few changes that need to be done. It is mostly about using the right driver for the MySQL database and modifying the `DatabaseUtils.props` from the `weka.jar` library with the settings of the database structure [22].

The MachineLearning class begins with making a connection to the database with the help of the JDBC driver. This connection will allow the existence of InstanceQueries that would attempt to get the training data and the data from testing table. It is important to specify the column that will be reviewed as the index or label. In this case, it will be the last column having a nominal representation. It can be observed that the queries to the database are normal SQL statements and that means that, if needed, there is the option to select only a few of the columns or limit the number of rows to be used.

```
query.setQuery("select * from mlverbtraining");
Instances traindata = query.retrieveInstances();
traindata.setClassIndex(traindata.numAttributes() - 1);
```

The model will then be built by using the chosen Classifier and applying it to the training data in the table. The Classifiers tested on the data were:

1. `Classifier cModel = (Classifier)new NaiveBayes();` - this is the Naive Bayes Classifier that uses a formulae of prior and posterior probability;

2. `Classifier cModel = (Classifier)new J48();` - this is the Decision Tree classifier, showing the model in the form of a pruned tree with leaves that represent the class labels

The MachineLearning class not only shows the result of the model, but it also shows an evaluation of it against the testing data using different visualisation outputs:

1. The statistics - shows a summary of the evaluation, its predictions and their accuracy.

2. The detailed class statistics - shows more in depth information about the summary statistics.

3. The confusion matrix - is showing a comparison between the actual labels in the testing table and the predicted within a matrix [44]

The test set has the same format as the training set but it is very important for the test set to have distinct elements from the training corpus. Reusing the training data set in the testing stage will allow the model to memorise its input without training and learning to extend to other examples. Even if it would have very good results, this representation wouldn't be accurate. This assumption has been tested within the program and the difference was obvious.

4.3.3 Machine learning results

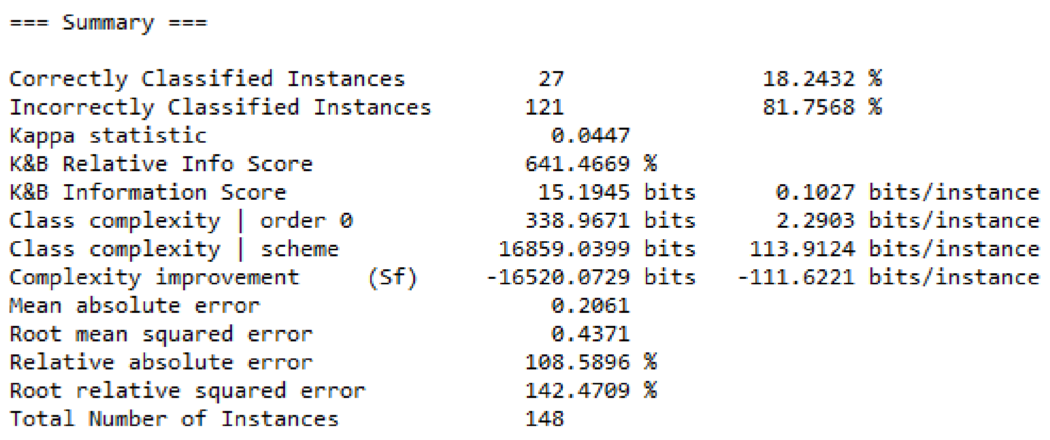
Due to the large amount of words, the machine learning algorithms couldn't predict an accurate classification when dealing with the testing data. The author believes this can be a sign that many of the verbs or nouns were present in a lot of the phrases and there was no clear differentiation between the classes.

Comparing the two classifiers, we notice a very similar result between them. If the success of the model in the evaluation is taken into consideration, then the result is less accurate than expected.

Naive Bayes classifier

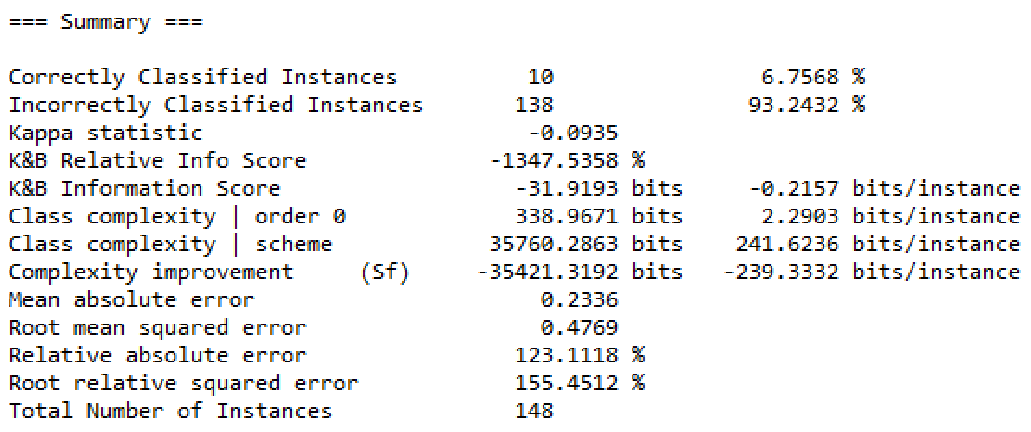
The first Figure 4.5 displays the summary of the evaluation using the Bayes classifier on both of the training tables: the one with noun attributes and the one with verb attributes. As it can be observed, the articles weren't labelled correctly in the majority of cases. However there is a clear difference between the correctly classified instances in the table using verbs evaluation, when compared to the nouns one.

It seems like the accuracy is better when dealing with verbs, which is possibly due to the smaller size of the table. Having a balanced number of attributes for each class increases the chances of a better prediction and decreases the confusion. The table that utilises nouns as attributes has 3 times more columns that the verb one and it is possible that this makes the classes more difficult to be predicted.



```
=== Summary ===

Correctly Classified Instances          27               18.2432 %
Incorrectly Classified Instances       121               81.7568 %
Kappa statistic                          0.0447
K&B Relative Info Score                641.4669 %
K&B Information Score                   15.1945 bits      0.1027 bits/instance
Class complexity | order 0             338.9671 bits      2.2903 bits/instance
Class complexity | scheme            16859.0399 bits    113.9124 bits/instance
Complexity improvement     (Sf)     -16520.0729 bits   -111.6221 bits/instance
Mean absolute error                      0.2061
Root mean squared error                  0.4371
Relative absolute error                108.5896 %
Root relative squared error            142.4709 %
Total Number of Instances              148
```

(a) Summary cross-validation of verb table



```
=== Summary ===

Correctly Classified Instances          10                6.7568 %
Incorrectly Classified Instances       138               93.2432 %
Kappa statistic                         -0.0935
K&B Relative Info Score              -1347.5358 %
K&B Information Score                  -31.9193 bits     -0.2157 bits/instance
Class complexity | order 0             338.9671 bits      2.2903 bits/instance
Class complexity | scheme            35760.2863 bits    241.6236 bits/instance
Complexity improvement     (Sf)     -35421.3192 bits   -239.3332 bits/instance
Mean absolute error                      0.2336
Root mean squared error                  0.4769
Relative absolute error                123.1118 %
Root relative squared error            155.4512 %
Total Number of Instances              148
```

(b) Summary cross-validation of noun table

Figure 4.5: The detailed results of the Naive Bayes classifier on the tables

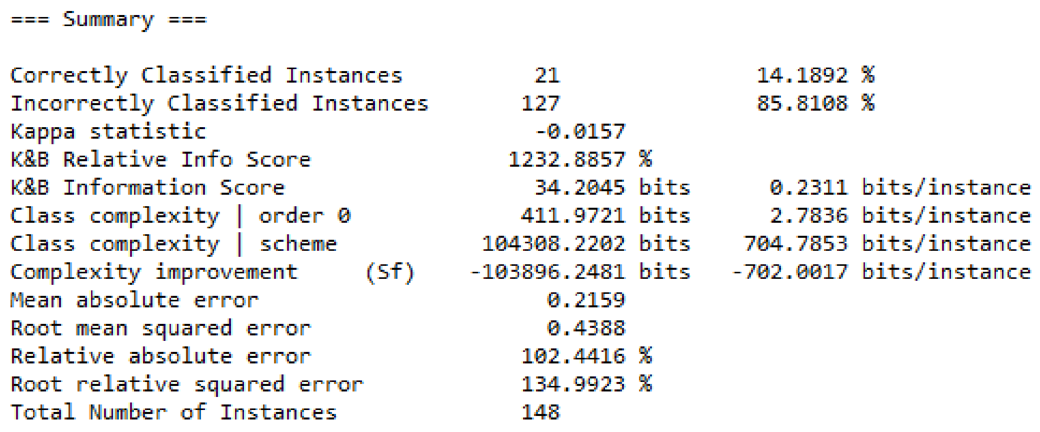
The data in detail can be viewed in the appendix D as well as the Confusion Matrix for this classifier D.1.

Decision Tree

With the Decision Tree algorithm, the number of correctly classified articles was low and very similar to what the Naive Bayes classifier was showing.
The Decision tree models for both of the tables show the following:

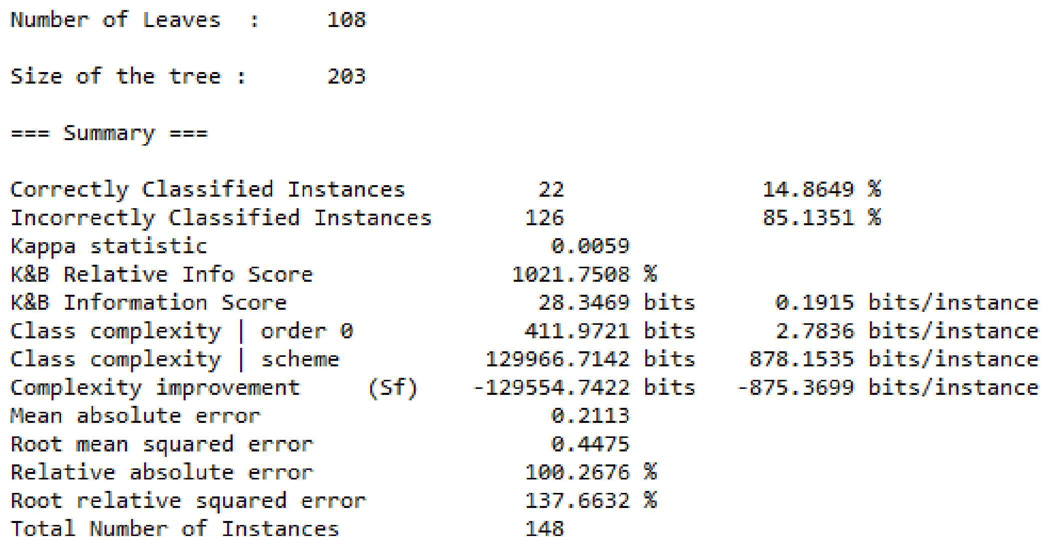
- For verbs attributes the number of leaves was 75 and the size of the tree was 149.
- For nouns attributes the number of leaves was 108 and the size of the tree was 203.

The following figure shows the summary of the evaluation using the Decision Tree classification.



```
=== Summary ===

Correctly Classified Instances          21               14.1892 %
Incorrectly Classified Instances       127               85.8108 %
Kappa statistic                         -0.0157
K&B Relative Info Score               1232.8857 %
K&B Information Score                   34.2045 bits      0.2311 bits/instance
Class complexity | order 0             411.9721 bits      2.7836 bits/instance
Class complexity | scheme           104308.2202 bits    704.7853 bits/instance
Complexity improvement     (Sf)    -103896.2481 bits   -702.0017 bits/instance
Mean absolute error                      0.2159
Root mean squared error                  0.4388
Relative absolute error                102.4416 %
Root relative squared error            134.9923 %
Total Number of Instances              148
```

(a) Summary cross-validation of verb attributes table



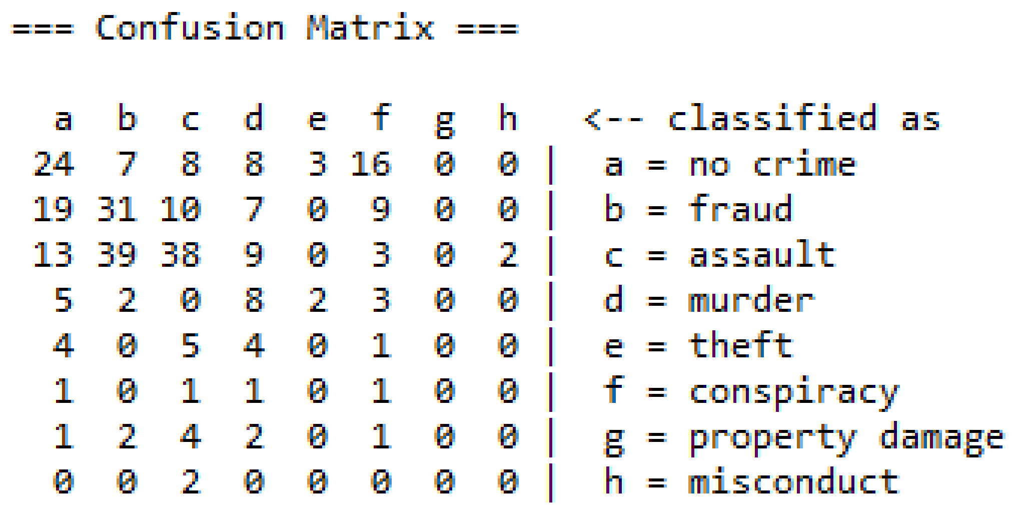
```
Number of Leaves  :     108

Size of the tree :      203

=== Summary ===

Correctly Classified Instances          22               14.8649 %
Incorrectly Classified Instances       126               85.1351 %
Kappa statistic                          0.0059
K&B Relative Info Score               1021.7508 %
K&B Information Score                   28.3469 bits      0.1915 bits/instance
Class complexity | order 0             411.9721 bits      2.7836 bits/instance
Class complexity | scheme           129966.7142 bits    878.1535 bits/instance
Complexity improvement     (Sf)    -129554.7422 bits   -875.3699 bits/instance
Mean absolute error                      0.2113
Root mean squared error                  0.4475
Relative absolute error                100.2676 %
Root relative squared error            137.6632 %
Total Number of Instances              148
```

(b) Summary cross-validation of noun attributes table

Figure 4.6: The detailed results of the Decision Tree classifier on the tables

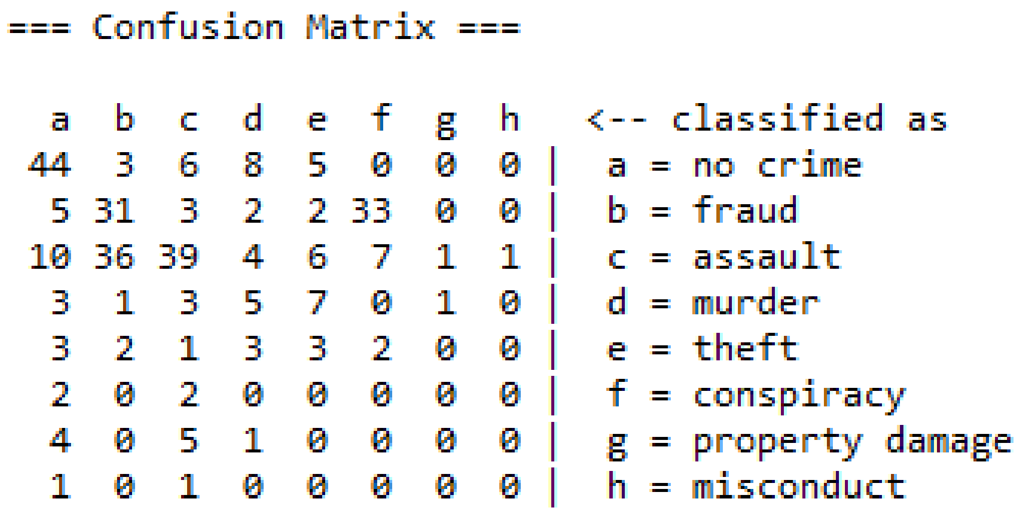
For the detailed evaluation of the Decision Tree classifier see the appendix D.2.

Visible in Figure 4.7, is the difference between the correct predictions and wrong ones as presented in the Confusion Matrix.



```
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h   <-- classified as
 24  7  8  8  3 16  0  0 |  a = no crime
 19 31 10  7  0  9  0  0 |  b = fraud
 13 39 38  9  0  3  0  2 |  c = assault
  5  2  0  8  2  3  0  0 |  d = murder
  4  0  5  4  0  1  0  0 |  e = theft
  1  0  1  1  0  1  0  0 |  f = conspiracy
  1  2  4  2  0  1  0  0 |  g = property damage
  0  0  2  0  0  0  0  0 |  h = misconduct
```

(a) Confusion matrix for verb attributes table



```
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h   <-- classified as
 44  3  6  8  5  0  0  0 |  a = no crime
  5 31  3  2  2 33  0  0 |  b = fraud
 10 36 39  4  6  7  1  1 |  c = assault
  3  1  3  5  7  0  1  0 |  d = murder
  3  2  1  3  3  2  0  0 |  e = theft
  2  0  2  0  0  0  0  0 |  f = conspiracy
  4  0  5  1  0  0  0  0 |  g = property damage
  1  0  1  0  0  0  0  0 |  h = misconduct
```

(b) Confusion matrix for noun attributes table

Figure 4.7: The Confusion matrix for the Decision Tree classifier

The Confusion Matrix shows that for example the "no crime" articles were labelled correctly 24 times when the attributes were verbs and 44 times when dealing with nouns. Also it can be noticed that the classifier can't recognise correctly the articles labelled as conspiracy as most of them are confused with either fraud or no crime.

It can be clearly seen in the above results that the attributes given don't represent the class enough and maybe a different approach needs to be taken. Inverse term frequency is a notion that has been continuously discussed in natural language processing and it has been studied in depth by Karen Sprck Jones. This refers to taking in consideration not only the frequency in which a word occurs in a specific article but also how little is the word present in the other text articles.

Another option that can be applied is to extract the words that are usually together or N-Grams, these might give a better understanding of the content when brought together into a compound rather than individually.

Chapter 5

Critical Evaluation

5.1 Achievements

Looking back over the initial objectives, we can discuss what has been achieved, what has been started and what is left to be done using the stages described in the Design Chapter 3.1.1.

1. National Library of Wales is queried for specific articles using search terms;

2. The XML response is processed and sent to a local database in the planned structure;

3. 589 articles have been manually labelled with the help of a Labelling Helper program that I developed;

4. The machine learning training and testing tables are created and populated with the article data. Unfortunately the decision was made to remove the columns for the multiple classification from the tables due to an issue with the result when applying a machine learning algorithm. However the methods that calculate the domain occurrence for each article and that add record to the database are still in the Java code but commented out.

5. Weka is connected to Eclipse IDE and the MachineLearning Class can use the machine learning tables to predict new classifications resulting into a model that is applied on the testing data. This stage is still a work in progress as there are multiple types of evaluations that can be applied and results that can be printed. Also there are issues with the attributes used as the classifiers don't have accurate results in the evaluation process.

6. Extraction of data resulted and it's display on a website are stages that haven't been reached because the machine learning model is not ready to classify correctly the articles.

The program as it stands is usable but, to obtain the accuracy of results needed for a good model, a much larger data set would be required and a rethinking of the attributes considered.

5.2 Problems encountered

Across the entire process of the Understanding crimes of the past project there have been multiple bumps in the road that arise from either inattention, misspellings or lack of depth knowledge in an area. In the following I will try to summarise these problems and how they have been(or could have been) dealt with.

I believe that the project scope and objectives was ambitious for the amount of time given. My objective was to make this system automatic so that there is no need of code editing, especially in regard to SQL queries and working with the MySQL Workbench.

There were statements and inserts or create queries that could have been done by working with the MySQL interface that would have split in half the amount of time needed. For example populating the machine learning tables could have been done directly through the Workbench and not using JDBC driver and Java methods. It is a difficult decision to make weather you build an entire system or focus only on the research, applying the machine learning algorithm as soon as possible to be able to cross validate it and analyse it. I am happy with the fact that I take the time to work with Java and the SQL statements as this experience would be invaluable later on.

Labelling the data was a major part of the project both as importance but also as the amount of time spent on it. It was a process that took approximately 3 days that had some tricky parts such as articles that where difficult to label even for a human: the attempted murders, the articles about crime statistics, the ones with so many strange characters that you wouldn't recognize the words or understand the sentence. Also its seemed for me that I started to look at the words within the articles (just as a computer would do): with theft, assault and murder I kept noticing verbs: "stealing", "robbed", "cut", "beating", "stabbed"; with conspiracy was more about: "dynamite", "bomb", "plot"(nouns); with fraud: "false", "false" "pretences", "falsified", "forgery", "forged"(a mix). The results don't seem to show the same patterns that I have observed but I believe the classifiers prediction could be optimised with more time.

Beside labelling the data, I noticed that the SQL and JDBC development was the stage that took the longest. During this process there was a steep learning curve starting from Statements and later using PreparedStatements that offered me an easier way to interact with the String queries. It was not necessarily a flow of difficult problems to solve but they were time consuming and in the grand scheme of things I wish I had more time.

There were moments of frustration resulting from SQL errors after errors that, due to the lack of a clear description of the problem, ended up taking more time that they should. In the end most of them were just small misspellings, negligence or extra or missing characters.

Another problem seemed to be the time necessary to work with large sets of data. The computer processor i5-3210M, 2.50Ghz was clearly not powerful enough to produce the performance required and to speed up the text processing stage and to insert records faster. With more time the system could have had more training and testing data that would prepare the classifier model to label articles more accurately. As a guide populating the largest machine learning table with the nouns occurrences (which are over 600) it took from 10:39 - 16:50.

After the XML parsing and the download was done I have noticed that the initial tables had a structure issues that required to alter the table and the code. This was a misunderstanding of my part and the Publication table had the Issue Date instead of the Article table. I noticed this issue after all the articles planned were already downloaded locally in the database.

The real problem came with having to re-download al the IssueDate information from the server plus the extra data that wasn't there as the publication table only kept the Issue date of the first article. Starting from scratch wasn't an option so I had to improvise: created a separate function that was inserting data only into the new IssueDate column and commented out most of the other functions so the data from the tables doesn't get updated and the process doesn't take too much time. The initial stage was to remove the column entirely from the Publication table and rebuild it as an extension of the Article table. This was done using the MySQL Workbench to quickly drop the column and rebuild it in the Article table. The damage was fixed and it only took an evening to have everything back on track.

Some changes and decisions were made even in the final weeks. This was relating the machine learning tables and their structure. Because a closer look at the Weka API showed no evidence of how to use the database without taking into consideration a couple of the columns (like the ones that were implemented for the multiple classification) the columns with domain names that were holding data about the labelling of multiple domain for different articles had to be removed. This resulted into two tables for the classification using verbs: one for testing and one for training the model and two tables for the classification using nouns. These were all classified only using text domain classification (the actual labels). Making this decision resulted in me actually being able to run two of the algorithm chosen and record the results. Even if these results came at a very late date this still proves that the methods implemented are working correctly.

5.2.1 Future development

Building a machine learning system that uses natural language processing is a complex task that can be continuously improved and remodelled.

I believe that the most important step to take forward is to better test the machine learning algorithms and see how the results change from analysing the text using verbs and analysing the text using nouns and even using a combination of the two. This would be the first part that I would be look into as it's important for the model to classify the articles accurately before improving any other parts of the system. After obtaining a better model I would also continue looking into how to apply it to unlabelled articles.

Text processing and cleaning

A major weakness of the program is its inability to deal with duplicate and split articles. With duplicate articles a different direction could have been taken, using a comparing method to check if the article has same context but different IDs. The algorithm that would be needed should take into consideration a collection of factors such as: the number of words, the number of verbs or nouns and either the title or the beginning and ending word of the text. It is not a perfect solution but it could be implemented and it could eliminate a good amount of articles that have duplicates.

Regarding the split articles this appears to be a much harder problem especially in the structure of this system. This is mostly because of the way the articles are selected from the database: by using search terms and querying the database. This means there is a high possibility for split articles that one of the parts wouldn't appear in the results due to the lack of the specific search term in its content. If the articles were selected without a search term query then both parts would have be returned in the XML result offering a better chance to find them and put them together as a whole article of text. This would have been an easy fix because it was noticed that for split articles there is a specific way of IDing them they have a number at the end of their ID representing what part they are. All the articles IDs end in "-1" and the one that are split have a second record that has the first part of the ID identical but it ends in "-2".

Another important drawback that can be discussed was the clarity of text. There were multiple records especially in the nouns table that weren't real words. This is mostly due to the state of the text in the NLW database resulted from the OCR misreadings and the Publication paper and ink quality. It would be a very interesting project to tweak the image processing and also if not enough to clean the text by using natural language processing, and finding the misspelled words and unwanted characters by analysing the entire context. But this for another time, another student.

Other future plans

Other than the obvious future developments that have been described above there are little things that could be done to improve the system and can make it more useful to users and to the National Library of Wales, starting from completing the rest of my objectives such as the possibility of multiple classification and the statistics regarding different types crime in time and space, building the website.

Another important consideration regarding future work is a better system in regard to its maintainability. This means refactoring of code, especially within the classes that deals with the MySQL database queries. I believe refactoring is a very important practice especially in regard to simplicity. Even if the practice might contradict in certain ways with YAGNI it doesn't affect the overall idea of removing unnecessary parts, clearing code duplications and keeping classes separate. Also testing the performance of the system under a larger data load and fixing the `java.lang.OutOfMemoryError: Java heap space` error.

I would also want to look into connecting the Labelling Helper to the entire system so the labelling process could be done within the same process without having to start a different program. This could be done either by actually including the code into the main program or by calling the Labelling Helper jar file from the system Menu using different Java libraries such as the ProcessBuilder Class or the Runtime Class.

As specified in the Development Process Chapter my work is saved entirely in a GitHub private repository. After results day and graduation I will make the data available and open source if Aberystwyth University permits it for other people to look at, edit, use as inspiration. For this I would like to make a read me file and a maintenance manual that explains better the options included, the downsides and what parts can be improved.

5.3 Conclusions

The process has been successful in building an automatic system that can take specific articles queried from the National Library of Wales and process them into a local database that can be used with a machine learning algorithm.

During this project I have noticed that the agile development methodology brings with it a constant need for refactoring. So even if I wouldn't think about rewriting or going over my code again this would be automatic required whenever new features had to be added. The process of finding a new way to add a new function without duplicating code or recalling a memory or process consuming task (such as large SELECT queries) was challenging but intriguing, similar to a new problem that you have to solve. Finding a solution and ending up with cleaner and more effective code was very satisfying. This however raised the question whether YAGNI is actually a more time consuming way of developing. Would a clear plan that has thought about almost every task and how they connect be more time efficient? However, I believe, because this project was a combination between research and development it needed the freedom to change tasks and objectives at any stage of the project.

The aim was to start with basic requirements for a simple product and build more features into it along the way, to make it nicer and offer more functionality to the user. As much as I appreciate the idea of no design planning and more customer input that XP offers I believe in my case some structure would be welcomed. Feature driven development has offered me the opportunity to change my mind and reconsider decisions. Looking back I feel that sometimes I should have taken a step back and looked at the bigger picture rather then get caught in details.

I am, in the end, proud of what I have achieved, not only my understanding of the machine learning processes has expanded but this project also thought me a lot about other areas of computer science and not only. As a computer science student I believe this project rounded up the whole university experience and knowledge. To explain this better I can refer to the following:

Logical thinking and problem solving - I enjoyed to build each task within the system and I designed starting from written pseudo code to real code to refactored code. Working with MySQL within Java was a new interesting experience and continuously trying not to repeat statements and methods that sometimes differ only due to the table name was intriguing.

Understanding new APIs and working with external software and libraries - I am now able to look into new technologies and understand how to implement them into a known language. From the tools used during the implementation of this system I can mention: JDBC, MySQL, Stanford Libraries, URLConnectionReader. Also the project has introduced me to other areas that were new to me: Machine Learning algorithms, Natural Language Processing, Lemmatisation.

Organisation - I appreciate the introduction to source control. Even if this was a known tool it was the first time I have actually used it properly by uploading commits and commenting on stages of the progress.

Presenting my work to both people that understand it, in technical terms or to others, in layman terms - I was worried about talking with people about my work but as I was getting more and more into its detail this fear transformed into excitement, I learned that talking about what you are doing not only helps you improve the way you communicate but also helps you consolidate your own knowledge by finding new way of presenting it. Talking to people also made me realised what seem to be the most important and interesting parts, for example the fact that this system can be used not only with crime articles but can be extended on other domains.

Working on my own - This was the first real experience of having a project of my own. There was no group task and no other students doing similar work to mine. It started with my own ideas and all the decisions towards completing it were mine. It is the perfect ending to my university course because what I know now is that when I will have the next idea I won't be afraid to approach it on my own.

Chapter 6

Bibliography

[1]

[2] The book is a very good resource for a beginner in data mining as it explains the process starting with simple examples of machine learning tables and continuing with more and more complicated ones. It was used at the beginning to understand the way a machine learning system works.

[3] "Wikibooks - TeX and LaTeX ," http://en.wikibooks.org/wiki/LaTeX, 2014.

The website was very helpful with explaining basic and advanced LaTeX techniques to a beginner like me. Mainly used when dealing with spaces, title page and images within LaTeX

[4] AlchemyAPI, Inc., "AlchemyAPI FAQ," http://www.alchemyapi.com/developers/faq/, Last accessed: 1st February 2014.

This is the first Natural Language Processing Library that I found. The website explains sentiment analysis and other interesting features of this software.

[5] Alias-i LingPipe, "LingPipe API Tutorials," http://alias-i.com/lingpipe/demos/tutorial/read-me.html, Last accessed: 1st February 2014.

The website of one of the LingPipe Library. It shows its features and the licence.

[6] Allu, "Difference between SAX and DOM parsers?" http://allu.wordpress.com/2006/12/28/difference-between-sax-and-dom-parsers/, Last accessed: 07th Macrh 2014.

This post explains the differences between the XML parsers in JAVA and was useful when researching what Library to use to parse the XML response resulted from the SOLR queries.

[7] ——, "What is the difference between SAX and DOM?" http://stackoverflow.com/questions/6828703/what-is-the-difference-between-sax-and-dom, Last accessed: 01nd May 2014.

The answers in this thread are very insightful and provide information that help with understanding the differences between the XML parsers that were considered.

[8] Andrew Ng of Stanford University, “Machine Learning Lectures,” Online Course iTunes U, 2008, Last accessed: 26th March 2014.

I have used the first lectures to understand better what machine learning and data mining is and how to approach it. It thought me about data patterns and different types of machine learning: supervised, unsupervised, reinforcement.

[9] Bharath Kumar M, “How-does-Google-News-cluster-stories,” http://www.quora.com/Google-News/How-does-Google-News-cluster-stories, 2008, Last accessed: 25th April 2014.

This link has offered me a better understanding of the classification of Google News by reviewing the answers of a previous employee.

[10] Chris Sherman, “Google News Search Leaps Ahead,” http://searchenginewatch.com/article/2068044/Google-News-Search-Leaps-Ahead, Last accessed: 6st February 2014.

A news article about how Google is approaching news articles on the internet and what is more important. This helped with the research regarding how the Google team is classifying their news and what strategies they have.

[11] DB-engines, “System Properties Comparison MySQL vs. PostgreSQL vs. SQLite,” http://db-engines.com/en/system/MySQL%3BPostgreSQL%3BSQLite, Last accessed: 2nd March 2014.

The post on this website helped me with my decision regarding which RDMS to use. It is a comparison of the most used open source systems.

[12] Dropbox Inc., “Dropbox,” http://www.dropbox.com/, Last accessed: 02th May 2014.

This URL links to Dropbox. The website explains the features of using Dropbox and all its advantages. It was a tool that I was already using so it was easy to work with.

[13] Galal Aly, “Tagging text with Stanford POS Tagger in Java Applications,” http://www.galalaly.me/index.php/2011/05/tagging-text-with-stanford-pos-tagger-in-java-applications/, Last accessed: 15th March 2014.

This URL links to a blog post describing how the Stanford Library API is used in Java. It was very useful for applying the libraries on the articles text

[14] GitHub, “GitHub,” http://github.com/Techiemouse, Last accessed: 02th May 2014.

This URL links to my GitHub profile where my code was saved. This website has been used for source control and for keeping a history of the fixed bugs and edited code

[15] HossMan,The Apache Software Foundation, "Solr Relevancy FAQ," https://wiki.apache.org/solr/SolrRelevancyFAQ, Last accessed: 2nd March 2014.

The page presents the way SOLR queries get the response on the page and the reason behind their order. I have learned about their relevancy score.

[16] Jason Weston, "Support Vector Machine (and Statistical Learning Theory) Tutorial," http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf, Last accessed: 21th April 2014.

A detailed tutorial regarding SVN classification. This link was used when deciding towards the algorithms that will be used in the machine learning process with Weka

[17] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," in *Journal of Documentation*, vol. 28, 1972, pp. 11–21. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.115.8343

The article presents the results of analysis made regarding term frequency within a document and the term weighting that takes into consideration the value of a term within the collection and not only within a document.

[18] JSON, "Introducing JSON," http://json.org/, Last accessed: 21st February 2014.

This is the URL of the JSON website. It was used for researching this data format and the way it can be parsed

[19] Junling Hu, "Machine Learning Guidance For Beginners," http://www.aboutdm.com/2013/03/basic-steps-of-applying-machine.html, Last accessed: 8th February 2014.

This URL links to a short article regarding the stages of data mining. It was an insightful description especially when I was trying to understand the data mining stages and what would be the steps that needed to be considered during implementation.

[20] C.-H. C. A. S. E.-P. Lim, "Automated online news classifcation with personalization," vol. 173(2), pp. 320–329, 2001. [Online]. Available: http://dx.doi.org/10.1162/0891201053630237

An article explaining the Support Vector Machine classifier and the way it uses the hyperplane.

[21] L. Lloyd, D. Kechagias, and S. Skiena, "Lydia: A System for Large-Scale News Analysis," *: String Processing and Information Retrieval*, vol. 3772, pp. 161–166, 2005. [Online]. Available: http://dx.doi.org/10.1007/11575832_18

The report regarding the Lydia project wasn't specifying exact Natural Language Processing techniques regarding entity recognition but it was a guide towards understanding statistical analysis and also made me aware of what different approaches could be made towards presenting the data in an interesting way

[22] Machine Learning Group at the University of Waikato, "Machine Learning Guidance For Beginners," http://www.cs.waikato.ac.nz/~ml/weka/, Last accessed: 3rd April 2014.

The Weka website was more than helpful with software download, libraries and documentation links, especially when working with Weka using MySQL and changing the DatabasUtils file.

[23] Mark Otto, Jacob, "Bootstrap," http://getbootstrap.com/getting-started/, Last accessed: 10th March 2014.

The URL links to the Bootstrap software and documentation considered for the design of the final website.

[24] Martin Fowler, "Is Design Dead? - The Enabling Practices of XP," http://martinfowler.com/articles/designDead.html, Last accessed: 24th February 2014.

Very good article about agile methodologies, XP and evolutionary design. This article raises a few question regarding development with or without the initial design stage and it gave me a better understanding of advantages and disadvantages when using agile development methods

[25] Mike Baukes, "MySQL vs Postgres," https://www.scriptrock.com/articles/postgres-vs-mysql/, Last accessed: 2nd March 2014.

This link shows the main features of both MySQL and PosgreSQL. It informed me about the fact that MySQL is far more used in building websites and that for that reason it has more plug-ins and add-ons.

[26] Mkyong, "Hibernate - Many-To-Many Example - Join Table + Extra Column (Annotation)," http://www.mkyong.com/hibernate/hibernate-many-to-many-example-join-table-extra-column-annotation/, Last accessed: 26th April 2014.

The post gave me a better understanding of how to approach SQL statements when building the tables that have many to many relations.

[27] ——, "How To Read XML File In Java - (DOM Parser)," http://www.mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/, Last accessed: 14th April 2014.

An tutorial that was very useful with understanding the Java DOM library and how XML is parsed using it.

[28] National Library of Wales, "Guide to the APIs exposed for the digitised Welsh newspapers hacathon," http://hacathonwiki.llgc.org.uk/w/images/9/9b/SCIFAPIDoc.pdf, Last accessed: 10th April 2014.

This document was previously used to inform developers about the API of the Digitised Newspaper of National Library of Wales. It describes all the relevant queries and views and it helped me better understand what options I have when accessing the article information on their server.

[29] ——, "National Library of Wales Project," http://hacathonwiki.llgc.org.uk/w/index.php/Historic_newspapers_project_information, Last accessed: 5 March 2014.

This website gave me a better understanding of the digitization program and its history. It explains how the process has been started and about the OCR.

[30] ——, "National Library of Wales Project," http://www.llgc.org.uk, Last accessed:5th April 2014.

The website is the collection of newspapers that can be reviewed by publication, year or region and can also be filter using search terms. The website was used to check the accuracy of the printed text in comparison with the text extracted from the NLW database.

[31] Oracle and/or its affiliates, "MySQL Connector/J Developer Guide," http://dev.mysql.com/doc/connector-j/en/index.html, Last accessed: 25th March 2014.

This is the main source of understanding how the MySQL database connects to Eclipse so that it allows statements to be operated within methods written in Java

[32] Rahul Kumar Mishra, "Machine Learning Guidance For Beginners," http://mlthirst.wordpress.com/2013/01/08/machine-learning-guidance-for-beginners/, Last accessed: 8th February 2014.

The blog post was very useful in my search for materials regarding Machine Learning. As a beginner I wanted to start from the basics.

[33] Y. Shinyama and S. Sekine, "Named entity discovery using comparable news articles," in *Proceedings of the 20th international conference on Computational Linguistics*, ser. COLING '04. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004. [Online]. Available: http://dx.doi.org/10.3115/1220355.1220477

This is an article that describes a way to check the distribution of Named Entities in news articles. It was thought to be relevant due to the objective of finding more information within the text of the article about the place the event described occurred. This would require Named Entity recognition

[34] Sione Palu, "The Use of Java in Machine Learning," http://www.developer.com/java/other/article.php/1559871/The-Use-of-Java-in-Machine-Learning.htm, Last accessed: 2nd May 2014.

This article explains some of Machine Learning algorithms in relation to the Java programming language. It was an introduction to some of the algorithms in Weka

[35] K. Sparck, “Some Points in a Time,” *Computational Linguistics*, vol. 31, no. 1, pp. 1–14, Mar. 2005. [Online]. Available: http://dx.doi.org/10.1162/0891201053630237

Reading this article I was introduced to the idea of inverse term frequency and I am thinking that this would help with building a better attribute set for the machine learning algorithm together with n-Grams

[36] SQLite, “Appropriate Uses For SQLite,” http://www.sqlite.org/whentouse.html, Last accessed: 2nd March 2014.

This document describes situations where SQLite is an appropriate database engine to use versus situations where a client/server database engine might be a better choice.

[37] stackoverflow - kris, ffriend, “Supervised Learning, (ii) Unsupervised Learning, (iii) Reinforcement Learn,” http://stackoverflow.com/questions/15782956/supervised-learning-ii-unsupervised-learning-iii-reinforcement-learn, Last accessed: 28th April 2014.

This is an overview of the basic machine learning tasks. It helped me with understanding the difference between supervised, unsupervised and reinforcement learning

[38] StatSoft, “Naive Bayes Classifier Introductory Overview,” http://www.statsoft.com/Textbook/Naive-Bayes-Classifier, Last accessed: 20th April 2014.

The URL links a very good explanation of the Naive Bayes algorithm. It helped me understand its formulae and how it works using simple graphics and math.

[39] The Apache Software Fondation, “Apache OpenNLP Developer Documentation,” https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html, Last accessed: 6th February 2014.

The website of Apache OpenNLP - the open source library for natural language processing this was a strong contender but it wasn’t chosen because the number of features presented was limited.

[40] The Apache Software Foundation, “Apache SOLR Open Source,” http://https://lucene.apache.org/solr/, Last accessed: 5th March 2014.

The website has a lot of documentation regarding the SOLR open source search platform. I have learned about how queries are formed and specifically about the way to get a specific amount of results starting from a specific row.

[41] The Stanford Natural Language Processing Group, “The Stanford Natural Language Processing Group - Supported software distributions,” http://nlp.stanford.edu/software/index.shtml, Last accessed: 6th February 2014.

This links to the Stanford NLP Group that was the software of choice for the natural language processing stage because it was an open source software with a high accuracy and a large number of libraries.

[42] M. H. Tom M. Mitchell, "Machine learning - chapter 3 - decision trees," pp. 52–80, 1997. [Online]. Available: http://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf

A paper that explained the Decision Tree classification algorithm.

[43] Tutorialspoint, "JDBC - Create Tables Example," http://www.tutorialspoint.com/jdbc/jdbc-create-tables.htm, Last accessed: 25th March 2014.

This is an example of using the JDBC driver. It gave me a clear example of how to use it's API to connect to the database and create MySQL tables using Java

[44] University of Regina DBD, "Confusion Matrix," http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html, Last accessed: 4th May 2014.

This link has offered me a better understanding of what a confusion matrix means and how it is represented. It was particularly helpful when dealing with the evaluation of the model resulted from applying different classifiers.

Appendix A

Initial Designs

A.1 Object Model

The initail Object model can be reviewed on the following page

Object Model
06/02/2014

Diana Silvia Teodorescu, dst1@aber.ac.uk

About the project (the aim and the background)

The project will be an analysis of the data collection provided from the historical newspapers found at the National Library. The newspapers are digitised and have a basic API that will be used to search, request and extract specific data.
The National Library holds the entire data and can provide information regarding the API and the other resources available. They might be interested in the end product but they won't be involved the project progress and don't have specific requirements regarding it execution except for their need to represent the data in an exciting way that would attract people to use it.

I am choosing to analyse specific articles related to crime in the past. This means having an algorithm to separate only the crime specific articles and then splitting them on type of crime. After this I will apply some statistic techniques to find different connections between age of victims, sex, place, year. I am thinking of extracting the data into another database and then representing it in a user friendly way on a website.

Technical/compsci challenges
- access of API only from computer science computers
- finding the best attributes to use the machine learning techniques
- applying machine learning and statistics in code - this will be most time consuming as I am new to it.
- thinking of what to hold the data in to so I am able to do the analysis fast.
- not all data is ladled correctly and the articles can have misspelled words

Other (non-technical) challenges
- Newspapers have welsh articles - this might be an issue as don't want to not take them into consideration but I know nothing about the language;

Ideal finished result and intermediate stages.
The ideal finish result would be a web application that presents the data in a user friendly way(graphs, pie charts) depending on user input. And depending on results maybe an interactive map.
Intermediate stages:
- algorithm for finding only crime related articles
- algorithm of splitting crimes into different types
- analysing and finding statistics depending on age, or sex or place, or year
- making a website where these will be presented to users with nice charts and graphs
- look into making an interactive map of the crimes on years and places; (if data is sufficient)

The parts of the project that most worry me are actually the ones I want to learn more about like: machine learning and statistical techniques. I feel that at the moment I don't have enough knowledge about these areas and I am eager to do some research and learn more about what are the best ways to implement statistics and machine learning into code.

A.2 Initial Data Flow Design

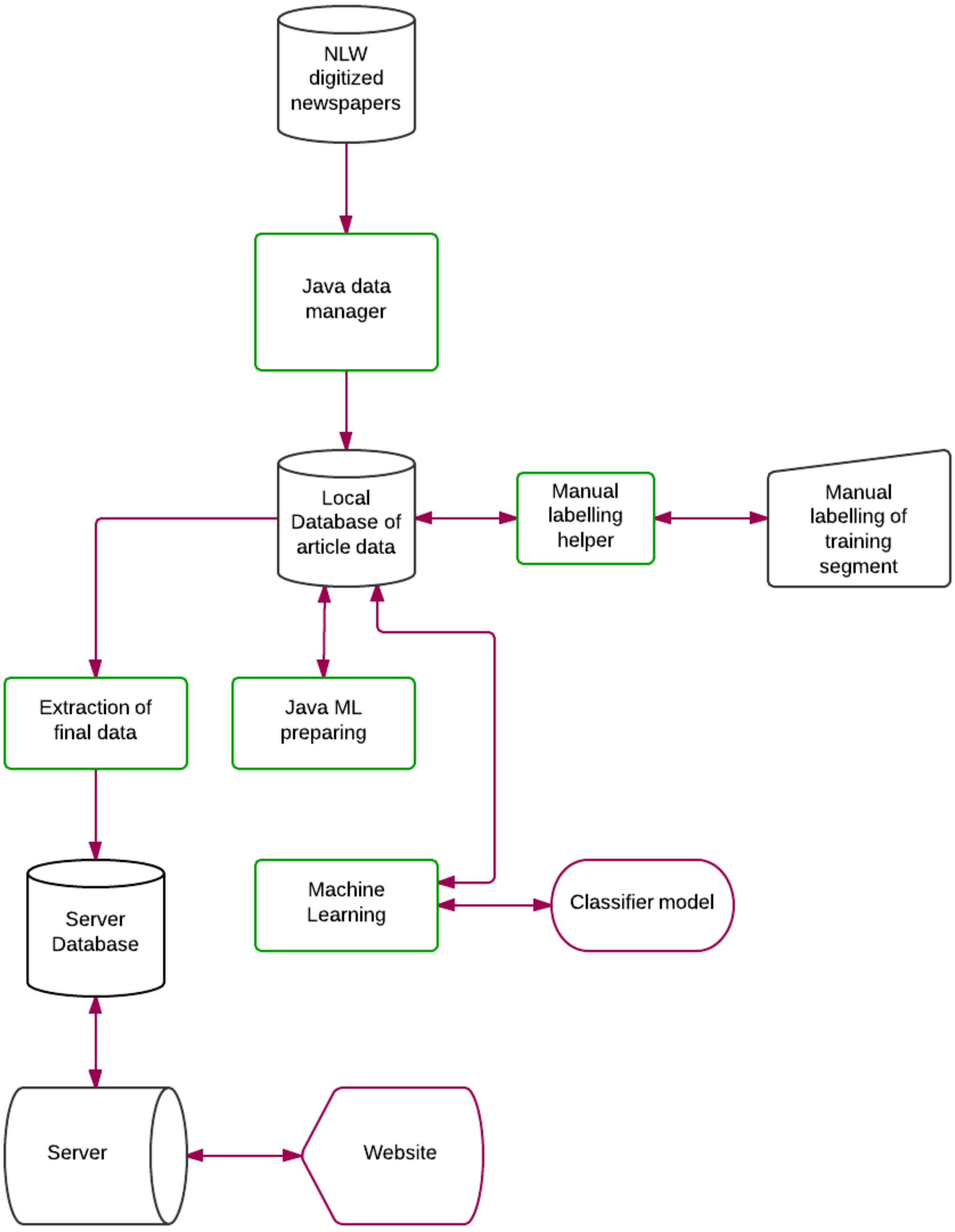
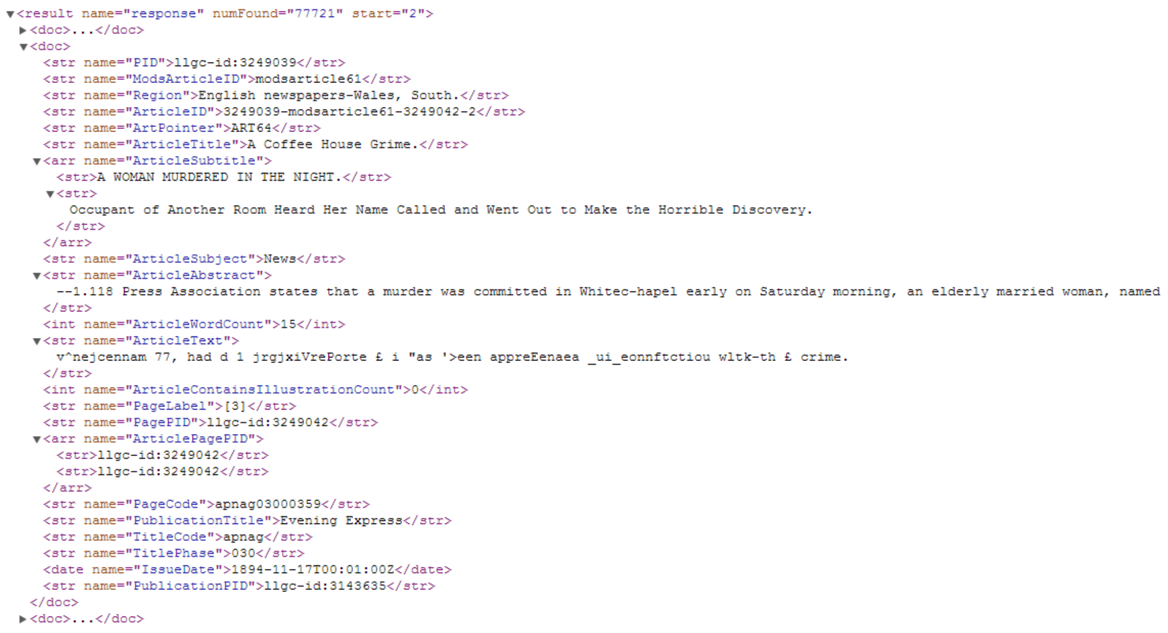


Figure A.1: The initial representation of the data flow designed on 14 February 2014

Text processing

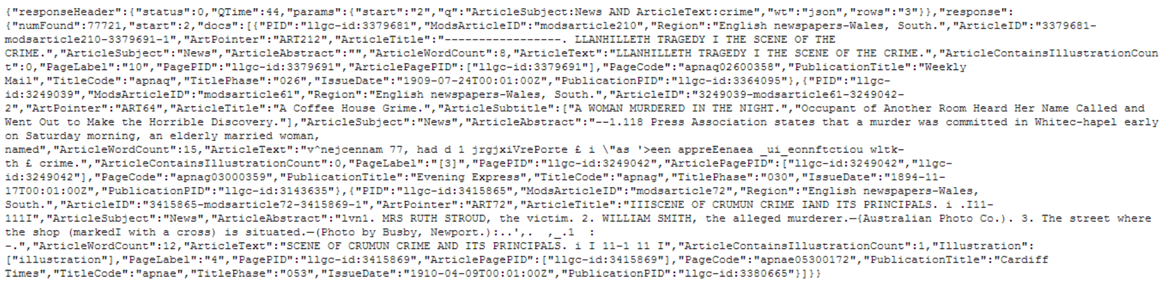
B.1 XML format



```
▼<result name="response" numFound="77721" start="2">
  ▶<doc>...</doc>
  ▼<doc>
     <str name="PID">llgc-id:3249039</str>
     <str name="ModsArticleID">modsarticle61</str>
     <str name="Region">English newspapers-Wales, South.</str>
     <str name="ArticleID">3249039-modsarticle61-3249042-2</str>
     <str name="ArtPointer">ART64</str>
     <str name="ArticleTitle">A Coffee House Grime.</str>
   ▼<arr name="ArticleSubtitle">
       <str>A WOMAN MURDERED IN THE NIGHT.</str>
     ▼<str>
         Occupant of Another Room Heard Her Name Called and Went Out to Make the Horrible Discovery.
       </str>
     </arr>
     <str name="ArticleSubject">News</str>
   ▼<str name="ArticleAbstract">
       --1.118 Press Association states that a murder was committed in Whitec-hapel early on Saturday morning, an elderly married woman, named
     </str>
     <int name="ArticleWordCount">15</int>
   ▼<str name="ArticleText">
       v^nejcennam 77, had d 1 jrgjxiVrePorte £ i "as '>een appreEenaea _ui_eonnftctiou wltk-th £ crime.
     </str>
     <int name="ArticleContainsIllustrationCount">0</int>
     <str name="PageLabel">[3]</str>
     <str name="PagePID">llgc-id:3249042</str>
   ▼<arr name="ArticlePagePID">
       <str>llgc-id:3249042</str>
       <str>llgc-id:3249042</str>
     </arr>
     <str name="PageCode">apnag03000359</str>
     <str name="PublicationTitle">Evening Express</str>
     <str name="TitleCode">apnag</str>
     <str name="TitlePhase">030</str>
     <date name="IssueDate">1894-11-17T00:01:00Z</date>
     <str name="PublicationPID">llgc-id:3143635</str>
   </doc>
  ▶<doc>...</doc>
```

Figure B.1: The representation of the article within an XML response

B.2 JSON format



```
{"responseHeader":{"status":0,"QTime":44,"params":{"start":"2","q":"ArticleSubject:News AND ArticleText:crime","wt":"json","rows":"3"}},"response":
{"numFound":77721,"start":2,"docs":[{"PID":"llgc-id:3379681","ModsArticleID":"modsarticle210","Region":"English newspapers-Wales, South.","ArticleID":"3379681-
modsarticle210-3379691-1","ArtPointer":"ART212","ArticleTitle":"-----------------. LLANHILLETH TRAGEDY I THE SCENE OF THE
CRIME.","ArticleSubject":"News","ArticleAbstract":"","ArticleWordCount":8,"ArticleText":"LLANHILLETH TRAGEDY I THE SCENE OF THE CRIME.","ArticleContainsIllustrationCoun
t":0,"PageLabel":"10","PagePID":"llgc-id:3379691","ArticlePagePID":["llgc-id:3379691"],"PageCode":"apnaq02600358","PublicationTitle":"Weekly
Mail","TitleCode":"apnaq","TitlePhase":"026","IssueDate":"1909-07-24T00:01:00Z","PublicationPID":"llgc-id:3364095"},{"PID":"llgc-
id:3249039","ModsArticleID":"modsarticle61","Region":"English newspapers-Wales, South.","ArticleID":"3249039-modsarticle61-3249042-
2","ArtPointer":"ART64","ArticleTitle":"A Coffee House Grime.","ArticleSubtitle":["A WOMAN MURDERED IN THE NIGHT.","Occupant of Another Room Heard Her Name Called and
Went Out to Make the Horrible Discovery."],"ArticleSubject":"News","ArticleAbstract":"--1.118 Press Association states that a murder was committed in Whitec-hapel early
on Saturday morning, an elderly married woman,
named","ArticleWordCount":15,"ArticleText":"v^nejcennam 77, had d 1 jrgjxiVrePorte £ i \"as '>een appreEenaea _ui_eonnftctiou wltk-
th £ crime.","ArticleContainsIllustrationCount":0,"PageLabel":"[3]","PagePID":"llgc-id:3249042","ArticlePagePID":["llgc-id:3249042","llgc-
id:3249042"],"PageCode":"apnag03000359","PublicationTitle":"Evening Express","TitleCode":"apnag","TitlePhase":"030","IssueDate":"1894-11-
17T00:01:00Z","PublicationPID":"llgc-id:3143635"},{"PID":"llgc-id:3415865","ModsArticleID":"modsarticle72","Region":"English newspapers-Wales,
South.","ArticleID":"3415865-modsarticle72-3415869-1","ArtPointer":"ART72","ArticleTitle":"IIISCENE OF CRUMUN CRIME IAND ITS PRINCIPALS. i .I11-
111I","ArticleSubject":"News","ArticleAbstract":"lvn1. MRS RUTH STROUD, the victim. 2. WILLIAM SMITH, the alleged murderer.—(Australian Photo Co.). 3. The street where
the shop (markedI with a cross) is situated.—(Photo by Busby, Newport.):..',.  ,_.1  :
-.","ArticleWordCount":12,"ArticleText":"SCENE OF CRUMUN CRIME AND ITS PRINCIPALS. i I 11-1 11 I","ArticleContainsIllustrationCount":1,"Illustration":
["illustration"],"PageLabel":"4","PagePID":"llgc-id:3415869","ArticlePagePID":["llgc-id:3415869"],"PageCode":"apnae05300172","PublicationTitle":"Cardiff
Times","TitleCode":"apnae","TitlePhase":"053","IssueDate":"1910-04-09T00:01:00Z","PublicationPID":"llgc-id:3380665"}]}}
```

Figure B.2: The representation of a query response to the NLW server using JSON

B.3 Text cleaning and parsing

B.3.1 XML parsing code examples

```
if (elem.getAttribute("name").equals(
"ArticleText")) {

String article = replaceCharact(elem
.getTextContent());
artOb.setArticleText(article);

ArrayList<String> vLemmas =new ArrayList<String>();
ArrayList<String> nLemmas =new ArrayList<String>();

vLemmas = tagg.lemmAttributes(tagg
.taggIT(article, "verb"));


artOb.setVerbList(vLemmas);
artOb.setVerbCount(vLemmas.size());

nLemmas = tagg.lemmAttributes(tagg
.taggIT(article, "noun"));

artOb.setNounList(nLemmas);
artOb.setNounCount(nLemmas.size());
```

B.3.2 Replacing characters method

```
public String replaceCharact(String badText) {

return badText.replace("\u00c2", "").replace("", "")
.replace("\"", "\\\\\"").replace("’", "‘");

}
```

Appendix C

Database structure

C.1 Relationship Diagram

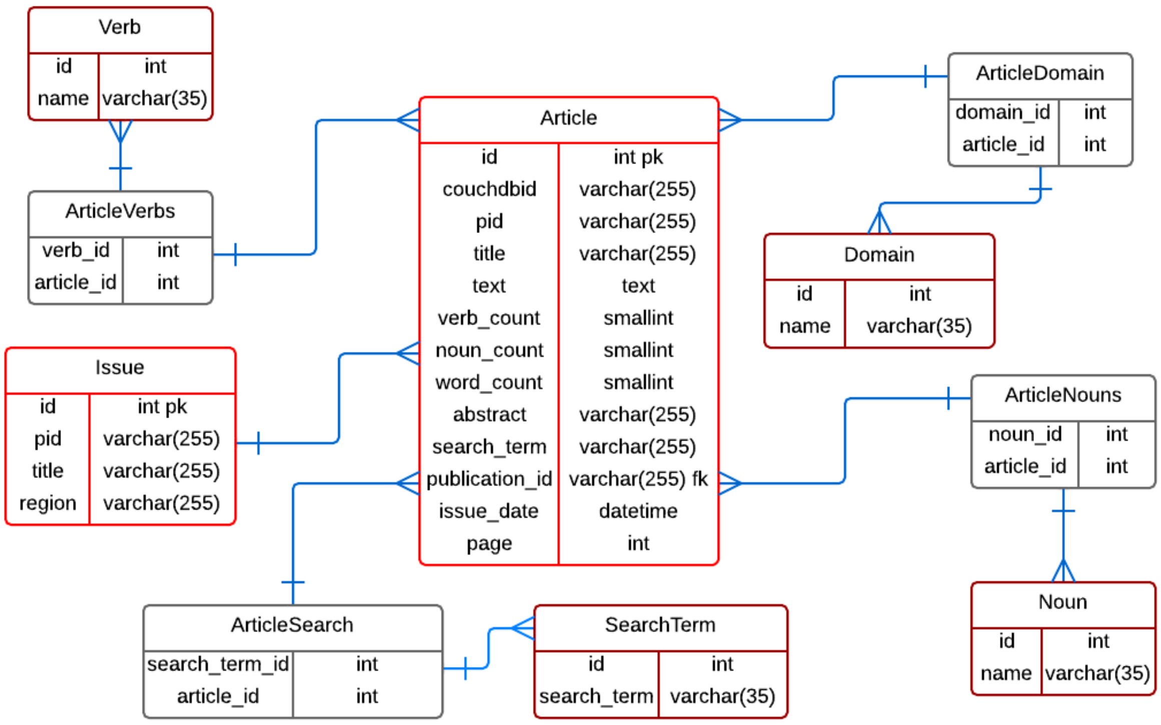


Figure C.1: A visual description of the relationships between the database tables

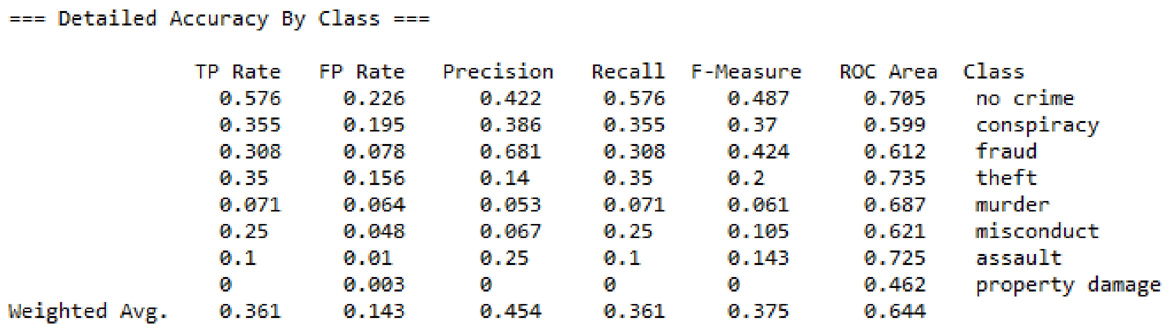
C.2 Building the Machine Learning Table

```
stringBuilder.append("CREATE TABLE IF NOT EXISTS "+tablename+" ( id INT NOT NULL AUTO_
 for (int i=0; i<verbList.size(); i++){
 stringBuilder.append("`"+verbList.get(i)+"`"+" SMALLINT, ");
 }
 for (int i=0; i<searchTermList.size(); i++){
 stringBuilder.append("`"+searchTermList.get(i)+"`"+" SMALLINT, ");
 }
 stringBuilder.append(" verb_count SMALLINT, ");
 stringBuilder.append(" word_count SMALLINT, ");
 stringBuilder.append(" page VARCHAR(20), ");
 stringBuilder.append("PRIMARY KEY (id))");

String finalString = stringBuilder.toString();
```

Appendix D

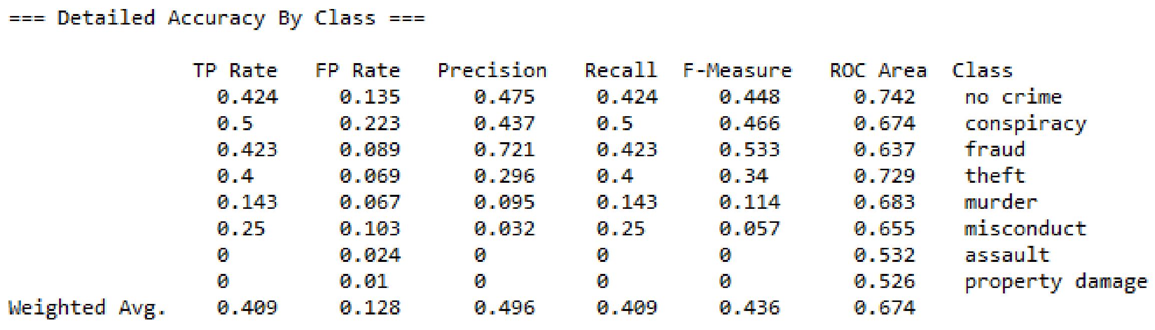
Machine Learning

D.1 Detailed results using Naive Bayes



```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.576     0.226      0.422     0.576     0.487      0.705    no crime
                 0.355     0.195      0.386     0.355     0.37       0.599    conspiracy
                 0.308     0.078      0.681     0.308     0.424      0.612    fraud
                 0.35      0.156      0.14      0.35      0.2        0.735    theft
                 0.071     0.064      0.053     0.071     0.061      0.687    murder
                 0.25      0.048      0.067     0.25      0.105      0.621    misconduct
                 0.1       0.01       0.25      0.1       0.143      0.725    assault
                 0         0.003      0         0         0          0.462    property damage
Weighted Avg.    0.361     0.143      0.454     0.361     0.375      0.644
```

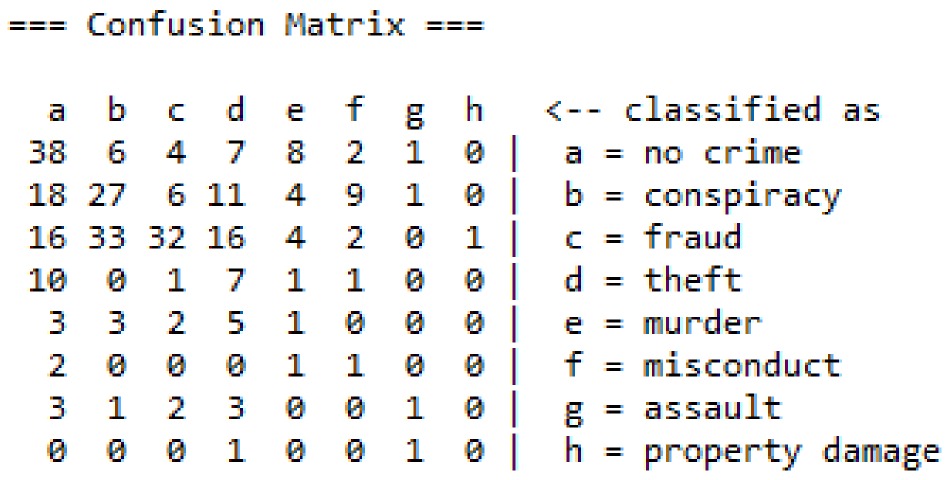
Figure D.1: The detailed results of the Naive Bayes classifier on the verb attributes table



```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.424     0.135      0.475     0.424     0.448      0.742    no crime
                 0.5       0.223      0.437     0.5       0.466      0.674    conspiracy
                 0.423     0.089      0.721     0.423     0.533      0.637    fraud
                 0.4       0.069      0.296     0.4       0.34       0.729    theft
                 0.143     0.067      0.095     0.143     0.114      0.683    murder
                 0.25      0.103      0.032     0.25      0.057      0.655    misconduct
                 0         0.024      0         0         0          0.532    assault
                 0         0.01       0         0         0          0.526    property damage
Weighted Avg.    0.409     0.128      0.496     0.409     0.436      0.674
```

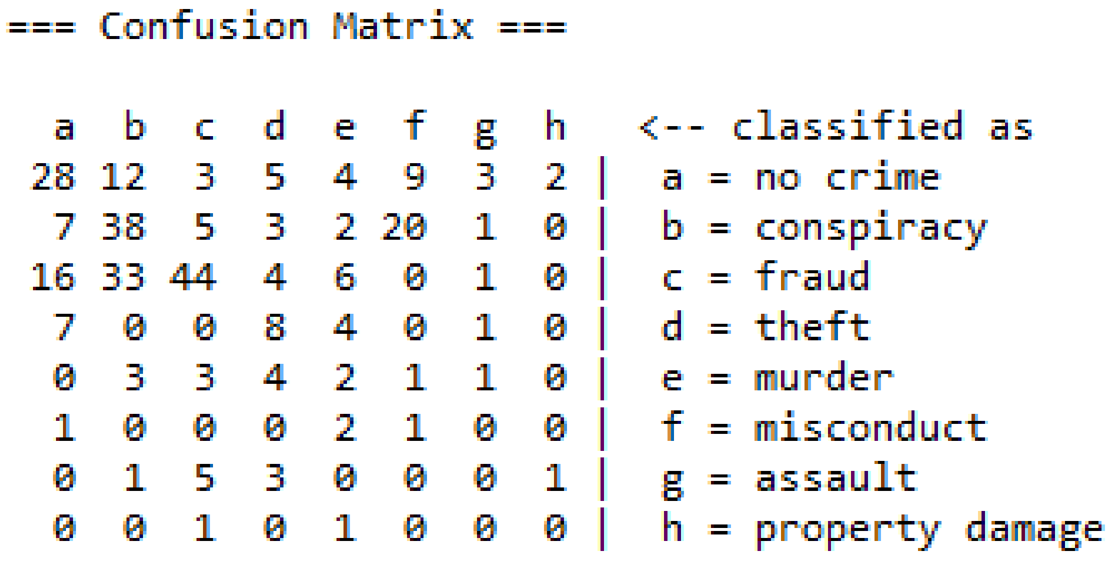
Figure D.2: The detailed results of the Naive Bayes classifier on the noun attributes table

D.2 Confusion matrix results using Naive Bayes



```
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h   <-- classified as
 38  6  4  7  8  2  1  0 |  a = no crime
 18 27  6 11  4  9  1  0 |  b = conspiracy
 16 33 32 16  4  2  0  1 |  c = fraud
 10  0  1  7  1  1  0  0 |  d = theft
  3  3  2  5  1  0  0  0 |  e = murder
  2  0  0  0  1  1  0  0 |  f = misconduct
  3  1  2  3  0  0  1  0 |  g = assault
  0  0  0  1  0  0  1  0 |  h = property damage
```

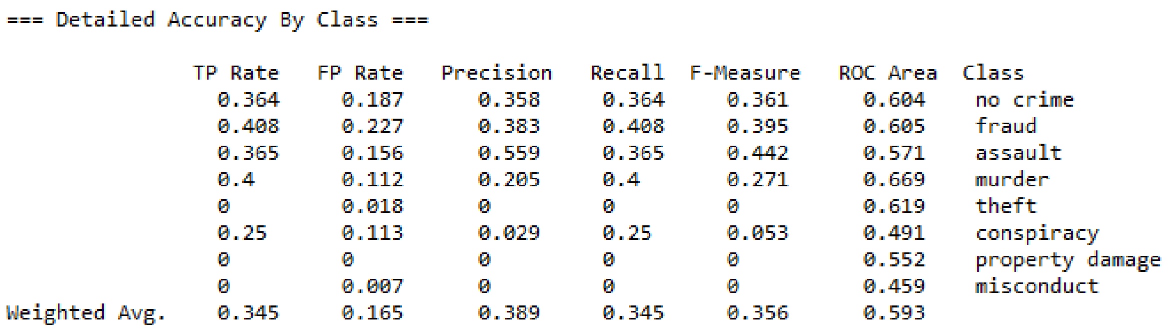
Figure D.3: Confusion matrix for verb attributes table



```
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h   <-- classified as
 28 12  3  5  4  9  3  2 |  a = no crime
  7 38  5  3  2 20  1  0 |  b = conspiracy
 16 33 44  4  6  0  1  0 |  c = fraud
  7  0  0  8  4  0  1  0 |  d = theft
  0  3  3  4  2  1  1  0 |  e = murder
  1  0  0  0  2  1  0  0 |  f = misconduct
  0  1  5  3  0  0  0  1 |  g = assault
  0  0  1  0  1  0  0  0 |  h = property damage
```

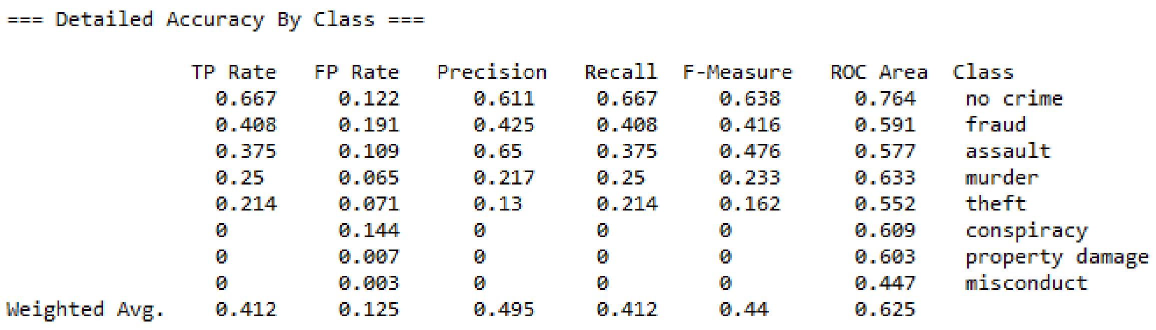
Figure D.4: Confusion matrix for noun attributes table

D.3 Detailed results using Decision Trees



```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.364     0.187     0.358     0.364     0.361      0.604     no crime
                 0.408     0.227     0.383     0.408     0.395      0.605     fraud
                 0.365     0.156     0.559     0.365     0.442      0.571     assault
                 0.4       0.112     0.205     0.4       0.271      0.669     murder
                 0         0.018     0         0         0          0.619     theft
                 0.25      0.113     0.029     0.25      0.053      0.491     conspiracy
                 0         0         0         0         0          0.552     property damage
                 0         0.007     0         0         0          0.459     misconduct
Weighted Avg.    0.345     0.165     0.389     0.345     0.356      0.593
```

Figure D.5: The detailed results of the Decision Tree classifier on the verb attributes table



```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.667     0.122     0.611     0.667     0.638      0.764     no crime
                 0.408     0.191     0.425     0.408     0.416      0.591     fraud
                 0.375     0.109     0.65      0.375     0.476      0.577     assault
                 0.25      0.065     0.217     0.25      0.233      0.633     murder
                 0.214     0.071     0.13      0.214     0.162      0.552     theft
                 0         0.144     0         0         0          0.609     conspiracy
                 0         0.007     0         0         0          0.603     property damage
                 0         0.003     0         0         0          0.447     misconduct
Weighted Avg.    0.412     0.125     0.495     0.412     0.44       0.625
```

Figure D.6: The detailed results of the Decision Tree classifier on the noun attributes table