Using Peer Assessment Data to Help Improve Teaching and Learning Outcomes

Zi Jin

U5541673@anu.edu.au

Supervised by Dr. Lynette Johns-Boast

November 2016

A thesis submitted in partial fulfillment of the degree of Bachelor of Advanced Computing (Honours) at The Research School of Computer Science Australian National University
I declare that, to the best of my knowledge, this thesis is my own original work and does not contain any material previously published or written by another person except where otherwise indicated.

Zi JIN
28/10/2016
Acknowledgements

I would like to express my deepest gratitude to all of the people who have helped me to achieve this individual project and report.

In particular, I would like to thank my supervisor, Dr. Lynette Johns-Boast. This was my first individual project, and Dr. Johns-Boast patiently taught me the basic knowledge about individual projects and academic writing. In addition, she helped me through very difficult times, especially in the initial phase when I did not have a clear aim, by helping me plan in greater detail. Furthermore, she is the bridge between me and the client, and organised many client meetings. She also gave me many helpful suggestions regarding presentation.

A big thank you to course convenor, Professor Weifa Liang. He organised 6 lessons which gave me many helpful suggestions, both on academic research and report writing. I also learnt many presentation skills in the mid-presentation organised by him.

A special thanks to my client, Dr. Shayne Flint for offering an opportunity to undertake and complete my first individual project. Furthermore, his assistance in the collection and preprocessing of the data used in this report was invaluable.
Abstract

Peer assessment is used in a number of courses in the ANU Research School of Computer Science. This report introduces the concept of using document analysis with qualitative peer assessment data to reveal valuable information which may help improve learning outcomes. A UML class diagram has been created in order to gain a better understanding of this theme, and the UML class diagram can also contribute to the construction of a database as a repository. The requirements and criteria for constructing a suitable database have been analysed and researched. Both the building of UML class diagram and the construction of a repository are preparatory steps for document analysis.

Text classification is at the core of the document analysis used in this project, because the foundation of extracting valuable information is the classification of quality peer feedback. Four machine learning methods: decision trees, Support Vector Machines (SVM), K-Nearest Neighbor (KNN) and Naive Bayes were evaluated in order to determine the most appropriate for use in this project. A series of experiments was conducted in order to test if text preprocessing can benefit text classification. Training data was labelled manually. The results obtained from the experiments indicated that SVM was the best method, and that text preprocessing cannot contribute to classification. This report details the analysis of information extracted from the classified quality peer feedback. In addition, it also discusses some potential applications based on the classification results.
Contents

Acknowledgements ........................................................................................................... 3
Abstract ............................................................................................................................. 4
Contents ............................................................................................................................. 5
List of tables ...................................................................................................................... 7
List of figures .................................................................................................................... 8
1. Introduction: ................................................................................................................ 9
   1.1 Document outlines ............................................................................................... 10
2. Understanding the problem space .............................................................................. 11
   2.1 UML class diagram ............................................................................................. 11
   2.2 Selection of repository ......................................................................................... 12
      2.2.1 NoSQL vs. SQL ......................................................................................... 12
   2.3 Document analysis ............................................................................................... 13
      2.3.1 Sentiment analysis ..................................................................................... 13
      2.3.2 Choice of document analysis tool .............................................................. 14
3. Methodology and implementation ............................................................................ 16
   3.1 Preparation for text classification ....................................................................... 16
      3.1.1 UML class diagram .................................................................................... 16
      3.1.2 Final repository selection ......................................................................... 18
   3.2 Text classification ............................................................................................... 19
      3.2.1 Methodology .............................................................................................. 20
      3.2.2 Implementation ......................................................................................... 21
4: Experiment results, discussion, and choice of model .............................................. 26
   4.1 Choice of metrics ................................................................................................. 26
   4.2 Experiment results .............................................................................................. 27
   4.2 Discussion ............................................................................................................. 28
5. Application .................................................................................................................. 29
   5.1 Implementation of the chosen model .................................................................. 29
   5.2 Application of the classification results .............................................................. 33
      5.2.1 The ratio of actionable and descriptive feedback ....................................... 33
      5.2.2 Identifying problematic students ............................................................... 33
      5.2.3 Identifying suggestions in a long text ........................................................ 34
      5.2.4 The relationship with quality feedback and group mark ......................... 34
      5.2.5 Common student performance-related problems ....................................... 35
6. Conclusion and recommendations for future work ........................................... 36
  6.1 Conclusion ........................................................................................................ 36
  6.2 Future work ....................................................................................................... 37
References ..................................................................................................................... 38
List of tables

Table 1: Comparison of Document Analysis Tools ......................................................... 15
Table 2: An example of a confusion matrix ................................................................. 26
Table 3: Accuracy rate for Decision Tree, SVM, KNN and Naive Bayes with three different text preprocessing approaches .......................................................... 27
Table 4: Recall rate of actionable feedback for Decision Tree, SVM, KNN and Naive Bayes with three different text preprocessing approaches ............. 27
Table 5: Precision rate of actionable feedback for Decision Tree, SVM, KNN and Naive Bayes with three different text preprocessing approaches .......... 27
Table 6: Confusion matrix of mutual check .................................................................. 32
List of figures

Figure 1: UML class diagram showing relationships among students, staff, courses, assessments and grades ................................................................. 17
Figure 2: Data import module in KNIME ................................................................. 23
Figure 3: Text preprocessing module in KNIME ......................................................... 24
Figure 4: Text preprocessing module which removing the ‘snowball stemmer’ in KNIME ......................................................................................... 24
Figure 5: Predictive modelling and scoring module in KNIME ......................... 25
Figure 6: Data import module ...................................................................................... 30
Figure 7: SVM classifier module.................................................................................... 30
Figure 8: Sample of the .xls file output by KNIME, where no student ID column and contribution column is compared with Appendix 2 ......................... 31
Figure 9: Sample of classification results processed by MATLAB ......................... 32
Figure 10: Proportion of actionable feedback and descriptive feedback in week 4 and week 6 ....................................................................................... 33
Figure 11: Example of a categorised peer assessment .............................................. 34
1. Introduction:

Nowadays, teamwork is an important skill practiced in higher education computing courses. In university, when an assessment requires that a team work together, the fairness of marking needs to be addressed. The ideal solution would be a grading system based on individual contributions by team members; however, it is extremely difficult for examiners to identify such individual contributions. The implementation of peer assessments can effectively solve this problem (Stock & Stephens, 2008). Peer assessment is a process in which students assess each other, based on a benchmark or marking rubric provided by the examiner or tutor. The final individual grade will thus be determined by the other team members, according to the benchmark provided. Apart from improving the fairness of the marking process, peer assessment can also save time for teaching staff, and improve the grading skills of students (Sadler & Good, 2006). Furthermore, some peer assessments also require team members to provide comments on the performance of each team member; for example, acknowledging and recognising good behaviour, and offering suggestions regarding overcoming any weaknesses. Such comments as part of peer assessments can help strengthen the communication skills of students (Lingard, 2010). At the ANU, peer assessments are used in some courses where group work is required.

Both Dr. Shayne Flint and Dr. Lynette Johns-Boast are lecturers at the Australian National University (ANU). Over the past couple of years, they have collected a large amount of data relating to peer and tutor assessment and grades, from students taking the various group project courses. Dr. Flint is very interested in what useful information can be extracted from this data, but there is too much data to read manually (around 600 pieces every two weeks), and presently there exists no automated process for such data analysis. Therefore, it was decided to develop some models to identify and extract valuable information from the data, with the aim of using the results to improve teaching and learning outcomes.

Course COMP3100 is based on group work mostly. For example, in the second semester of 2016, 80% of the final mark is based on peer assessment, including tutor review, group presentation and final project review, and students are asked to perform peer assessment to rate group performance and provide feedback to team members. In the tutor review and final project, each group receives a benchmark from the teacher, and the individual mark received by a student will be influenced by the aggregated performance rating. For the group presentation, students need to submit a set of peer assessments for other groups, and they subsequently receive an individual mark based on the quality of the peer assessment they provide. COMP3100 uses three different peer assessment formats: tutor review, group presentation, and final project review. Due to time limitations, the challenge of analysing all the data obtained from all three peer assessment types is too great; therefore, this report will focus on the peer assessment data specifically related to the tutor review process. The need to establish an effective method of analysis for this data is greatest due to the large quantity of data which it produces; more than 600 pieces every two weeks.
Appendix 1 shows a sample of a peer assessment submitted by a student in COMP3100. It contains basic information: team name, time, tutor name, student ID and student name, names and IDs of teammates, feedback for tutors, feedback for students, and the performance rating for students, where the total share of the performance rating is 100%. The aim of this project is to use this data to help improve learning and teaching outcomes. There are two initial approaches; firstly, determining which students require special attention, such as those who are not sufficiently engaged or lack certain skills. The details of a student, and the problem being experienced, can be shared with the tutor with the aim of enabling the tutor to intervene and help a student with their specific problem. Secondly, the quantitative feedback can be analysed and combined with the student grades. It is anticipated that there exists a relationship between student grades and the quantitative feedback. For example, there may exist a pattern of the feedbacks exit in the high mark students group or low mark students group.

1.1 Document outlines

The remainder of this paper is organised as follows:

Chapter 2: Understanding the problem space
Chapter two describes some potential approaches to achieve this project. A review of relevant literature related to each approach is presented, and the application of each approach is discussed.

Chapter 3: Methods and implementation
Chapter three presents the implementation of the UML class diagram, and the decision regarding the repository. It also provides the methodology of machine learning and the implementation of a series of experiments to determine the best method.

Chapter 4: Experiment results, discussion, and choice of model
Chapter four presents a discussion of the results obtained from the series of experiments, and the choice of the best model.

Chapter 5: Application of the model
Chapter five presents the implementation of the chosen model and some findings extracted from the modelling results. In addition, some potential applications are discussed and analysed.

Chapter 6: Conclusion and future work
Chapter six summarises the main contributions made by this project, and discusses possible areas for further research in the future, in order to expand this project.
2. Understanding the problem space

This section presents a review of existing literature relevant to UML class diagrams, the construction of databases and document analysis. This literature review is based on my understanding of this project. The first step involves understanding the data, which includes the meaning of data, and the various data relationships. The second step is the selection of an appropriate data storage repository, based on the previously mentioned understanding of the nature of the data, and one which can benefit the analysis process. Finally, an effective method of data analysis needs to be established.

2.1 UML class diagram

The data collected by Dr. Flint and Dr. Johns-Boast is complex in nature and considerable in terms of the quantity to be processed. Therefore, an effective method of understanding and analysing the data and its associated relationships is required.

One potential solution is Unified Modeling Language (UML), which is defined as an international industry standard graphical notation for describing software analysis and designs (Quatrani & Evangelist, 2003). From the definition, it is easy to see the first advantage of UML is its popularity. Quatrani and Evangelist use the term “international industry standard” to describe UML, which means that the implementation of a model using UML would be beneficial in terms of ease of communication. If UML is used to describe the data, other UML users will be able to understand the data easily, because UML is an international industry standard.

UML class diagrams are one type of UML diagram, which can be used for data modelling. In UML class diagrams, the main constituents are classes, and their association (Purchase, Colpoys & McGill, n.d.). Classes are abstracted from a set of entities which contains some behaviours or attitudes. The term ‘association’ refers to the individual relationships which exist in the UML class diagram.

The constructing of a UML class diagram can benefit the understanding of complex relationships. In class diagrams, the static structures of the system are shown visually, each class is abstracted from a set of entities logically and the associations among each class are defined (Quatrani & Evangelist, 2003). All classes and associations are named logically, and associations are labelled by a pair of numbers showing the mapping relationships.

Additionally, the construction of a UML class diagram can help to construct a database; as much of the content in the UML class diagram can be mapped to the database directly (Urban & Dietrich, 2003). For example, as a UML class diagram contains classes, each class can be regarded as a relation in a database, in which there is a one-to-one correspondence between the attributes contained in each class and the relation. Therefore, the construction
of a UML class diagram can not only help in understanding the relationships in the data, but can also help to construct a database.

In summary, UML class diagrams can be beneficial in understanding the data and the data relationships, and can contribute to the construction of an effective database. In addition, other UML users can easily understand the data, as UML is an international industry standard.

2.2 Selection of repository

As Appendix 1 shows, each peer assessment contains only a few items of feedback and the location of the cells which store the qualitative feedback are not fixed. That means any attempt to analyse all the feedback would require the extraction of each feedback item from hundreds of Excel files. However, the analysis of the marks would also require the extraction of each mark from the same Excel files. Furthermore, the volume of data is increasing. Before long, there would be a need to manage thousands of Excel files for student peer assessment data. Therefore, it is necessary to establish an appropriate data management process capable of handling such complex and large amounts of data.

A database can be constructed to store the collected data; the main benefit of which is greater efficiency (Frawley, Piatetsky-Shapiro, and Matheus, 1992). Firstly, nowadays most popular database applications are able to easily store and retrieve enormous amounts of data, with acceptable speed for inquiry and alter operations. Without such a database application, it is a difficult and slow task to perform inquiry and alter operations for thousands of Excel files. Secondly, peer assessment data also contains some information related to other peer assessments, such as the names of tutors and IDs of team members. Therefore, the construction of a database will enable the reorganisation of the raw data in a manner which illustrates the content and data relationships more clearly.

2.2.1 NoSQL vs. SQL

Nowadays, the two most popular types of database are NoSQL and SQL. Both of them can satisfy the basic requirements of this project in terms of data storage; however, each offers unique benefits and limitations, which require consideration in order to select the most appropriate type for this project. The first step is to define the selection criteria:

1. The ability to store various data formats. The database will need to be able to store thousands of pieces of data related to peer assessment, including numbers (e.g. grades) and strings (e.g. comments given by students). Potentially, the database will need to store many extra pieces of data in order to do further research; for example, submission of assessments. The format of submissions varies, and includes formats such as strings, PDF files, Word files, videos and websites, etc.
2. Scalability. This project is just starting; therefore, there is a need to consider future requirements in terms of storage capacity. Both horizontal and vertical Scalability should be considered. Vertically, the database will add 700 tuples for student comments every two weeks. Horizontally, new attributes will be added to the
database, which means importing new associations and, potentially, changing the structure of the database.

3. Easy to construct. The whole database will be constructed by the author, and this project emphasises application rather than conducting comparisons, ideally. Therefore the ability of the designer and complexity of implementation should be taken into consideration. As the sole designer on this project, the author is somewhere familiar with SQL databases but knows nothing about NoSQL databases.

4. Performance. All data must be stored correctly without any loss. The speed of the basic operations such as inquiry, alter, and delete should be acceptable. Acceptance is an intuitive judgement, depending on the test results. For example, an operation takes 0.01ms in SQL database, while a NoSQL database needs 0.1ms; 10 times slower. However, both speeds would be acceptable and in this case the fact that SQL is 10 times faster will not be regarded as a distinct advantage.

2.3 Document analysis

It was decided to implement document analysis for the peer assessment data, because all of the feedback in peer assessment is qualitative. Document analysis is a sub-task of qualitative analysis, which takes documents as the input (Bowen, 2009).

2.3.1 Sentiment analysis

The final aim of this project is to improve learning outcomes; there are two potential approaches.

- Building a pattern to recognise quality feedback. This can inform students how to improve their performance, through offering negative comments (e.g. “lacking programming skills”) or suggestions (e.g. “Learn more from online tutorials”). In this project, such feedback is termed “actionable feedback”; the aim being to identify this type of feedback and present it to tutors and the course convenor so that tutors can help those particular students.

- Analysis of the influence of feedback on the grade awarded to a team. For example, will team members give each other quality feedback to improve overall team performance? More specifically, considering the influence on individuals; if a student always gives quality feedback to others, will he benefit from it? Or if a student always receives quality feedback, will he perform better?

The foundation for the above approaches is the correct identification of all actionable feedback, which is related to sentiment analysis; referring to the task of mining opinions expressed in text and analysing the entailed sentiments and emotions (Liu, 2015). Primarily, sentiment analysis is based on text classification (Quinteiro-Gonzalez, Hernandez-Morera, and López-Rodríguez, n.d.). Text classification can be used to assign a text to a specific category. Therefore, as part of this project, it is necessary to construct a classifier able to recognise actionable feedback. Furthermore, sentiment analysis also relies on natural language processing. Before implementing text classification, text should be preprocessed. For example, Support Vector Machine (SVM) is a popular method used in text classification,
and a series of experiments show that appropriate text preprocessing can improve the SVM classification results (Isa, Lee, Kallimani, and Rajkumar, 2008).

2.3.2 Choice of document analysis tool

Based on the literature review and the understanding of the problem space, the ideal research document analysis tools should have the following three functionalities:

- **Text preprocessing**: this refers to the preprocessing of text before it becomes the input for text classification; the more functions a tool can support, the better the output results, potentially. However, there is no related research showing which text preprocessing module is most appropriate for classifying actionable feedback.

- **Text classification**: there are more than 600 items of feedback every two weeks which require classification; hence, the need for an appropriate tool. There are many methods of classification which have different features; therefore, the best strategy is a tool which can implement all of the popular classification methods in order to conduct comparisons.

- **Quantitative analysis**: the peer assessment data not only contains linguistic feedback which needs analysed qualitatively, but also contains quantitative data such as the marks students give their team members and the benchmark by teachers. One of the potential approaches is conducting quantitative analysis based on the results of qualitative analysis. In this project, some sample quantitative analysis abilities are adequate, especially the calculation of the average mark and analysing the trend of the mark.

Implementation of the tool requires cross-platform functionality as it will be installed on two different desktop computers with different operating systems: one running Linux (Dr. Flint’s) and the other running Windows (the author’s). Furthermore, the tool should be easy to implement; time limitations mean that, for example, a tool which requires learning a new programming language and programming a large amount of functions would not be desirable as an option.

Based on the above criteria, a comparison of five document analysis tools, KH Coder, tm (Text Mining Infrastructure in R), Natural Language Toolkit (NLTK), KNIME, and OpenNLP was conducted; the results of which are shown in Table 1, below.
Table 1: Comparison of Document Analysis Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Text preprocessing</th>
<th>Text classification</th>
<th>Quantitative analysis</th>
<th>Linux &amp; Windows</th>
<th>Ease of implementation (1-5) *</th>
</tr>
</thead>
<tbody>
<tr>
<td>KH Coder</td>
<td>√</td>
<td>Only supports one method</td>
<td>√</td>
<td>√</td>
<td>1</td>
</tr>
<tr>
<td>tm</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>5</td>
</tr>
<tr>
<td>NLTK</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>4</td>
</tr>
<tr>
<td>KNIME</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>2</td>
</tr>
<tr>
<td>OpenNLP</td>
<td>√</td>
<td>Only supports one method</td>
<td>√</td>
<td>√</td>
<td>4</td>
</tr>
</tbody>
</table>

Note: In the table, √ indicates it qualifies for this project.

* Values represent the level of difficulty to implement the tool (1 = easiest, 5 = most difficult).

Overall, KNIME is the most appropriate tool to complete the document analysis for this project. KH Coder and OpenNLP can only implement one specific text classification method, but in this project, there is a need to experiment with popular classification methods. The remaining three tools tm, NLTK, and KNIME can potentially achieve this project, but both tm and NLTK require the learning of new programming languages. Therefore, KNIME has been chosen as the method for document analysis in this project.
3. Methodology and implementation

This chapter presents and discusses the methodology and implementation. The first part presents the preparatory work required for text classification, which includes creating a UML class diagram and attempting to create a database to store the data. The second part details the methodology of text classification and the implementation of a series of experiments to establish the best text classification method.

3.1 Preparation for text classification

The first part includes creating a UML class diagram and attempting to create a database to store the data. That because before attempting text classification, the data and the complex relationships need to be understood and creating a database to store the data can make the text classification operation easier.

3.1.1 UML class diagram

Firstly, a UML class diagram was created because this can avoid the analysis process being limited by the collected data. In other words, it can inform and guide the data collection (Epstein, 2008). For example, a UML class diagram was created which contained all of the data relevant to an RSCS course, which suggested the need to capture data regarding the nationality and language background of students. However, if Dr. Flint does not already collect such data, a recommendation could be made regarding the need to capture this data for inclusion, with the aim of producing better results from the analysis. On the other hand, if a UML class diagram was simply created according to the data already collected by Dr. Flint, assuming the data did not already include details of nationality and language background, a more general model would be constructed as a result rather than a specific model. The inclusion of more specific data can allow for greater inspiration in terms of analytical approaches. Another reason is Dr. Flint needs a few weeks to collect and preprocess the data. Therefore, a general model was constructed to show the relationships among students, staff, courses, assessments and grades.
The UML class diagram shown in Figure 1 was created using an add-in plus Papyrus in Eclipse; an explanation of the diagram follows:

- Each block represents a class abstracted from a group of entities which have the same types of features. For example, the class ‘person’ is abstracted from all people related to a course; people include students, lecturers and tutors, while other classifications include ‘ANU staff’, ‘ANU students’ and ‘non-ANU persons’. Whatever the type of classification, all the people have similar related information, such as name, date of birth, occupation and ID. Therefore, they can be abstracted into a class, in this case assigned the tag ‘person’.
- A line with a hollow arrow represents ‘inheritance’, which refers to a child class inheriting all of the attributes contained in the parent class; in addition, the child class potentially has more attributes and behaviours. For example, the class ‘assessable’ and
‘non-assessable’ are two child classes, while the class ‘task’ is the parent class. The relationship between the child classes and parent class is that of ‘inheritance’. This means that ‘task’ has two specific types, because some tasks are assessable but some are not. Both of them inherit the same attributes from the parent class ‘task’, such as ‘task ID’, ‘task name’ and ‘due date’, etc. However, an assessable task has some specific attributes relevant to grades, and thus needs to be abstracted to a different class from non-assessable tasks.

- A solid line connecting two classes refers to association, which means a relationship exists between those two classes. The words on the line are a brief description of the association. The notation 1..* means 1 to many entities involved in the association. For example, an association exists in the class ‘person’ and ‘company’, a person can belong to 1 or more companies and a company contains 1 or more persons.

- A block connected by a dotted line refers to association class, which are used for associations that participate in an association with another class. For example, class ‘team’ is an association class which contains the properties of the association between ‘student’ and ‘group task’. Students join a team to finish group tasks together; therefore an association class containing the team information can refer to the association.

Briefly, the diagram (shown in Figure 1) is a general model representing the organisation of a course. It divides persons into clients who do not belong to ANU, staff and students. It uses two methods to classify tasks: assessable tasks and non-assessable tasks, and shows that grades only have a relationship with assessable tasks, and not with all tasks. Another method is that of dividing tasks into group tasks and individual tasks, and showing the relationship between group tasks and peer assessments; this means if the analysis object is peer assessment, only the group tasks need to be considered.

Basically, the class diagram contains most entities present during the teaching process, and their relationships are visualised, which can be used to construct a database. In addition, relying on the UML class diagram, the understanding of the data and the relationships becomes deeper. It is important to understand how the entire teaching system works, because it can help identify the core data, which in turn will allow the focus to be targeted on the most important data rather than dealing with all of the data.

### 3.1.2 Final repository selection

It was decided to analyse two organised .xlsx files directly rather than constructing a database. There are three reasons for this. Firstly, Dr. Flint preprocessed the peer assessment, extracting and organising the peer assessment data into a single .xlsx file every two weeks. Compared with the creation of a database, as mentioned in 2.3, these preprocessed .xlsx files are sufficient for document analysis, because the data is organised structurally and it is sufficiently convenient to be imported into document analysis tools. Secondly, in choosing the .xlsx files as the repository, Dr. Flint considers that the project can focus directly on the peer assessment data rather than all of the information involved in the UML model. Essentially, the construction of a database is a preparatory stage for document analysis; however, at the current stage, a database cannot add extra value to this project.
And due to time constraints, this might be considered for future work, because an organised .xlsx file cannot easily handle complex data, and the demand on a database increases with growth of data complexity. Lastly, the document tool KNIME can import and operate .xlsx files directly.

Appendix 2 shows the .xlsx files provided by Dr. Flint. Briefly, the file hides the name of students, tutors and groups for privacy issues, and instead uses a new ID such as 001, 002 to represent a student, tutor or group. The data consists of six sheets: teams, students, team feedback, individual feedback, peer assessments and peer feedback. The following provides more detail regarding the data content of each sheet:

- Teams: contains team ID and tutor ID; identifies which tutor is in charge of which team.
- Students: contains student ID and team ID; identifies team members within a team.
- Team feedback: contains team ID, benchmark for a team, and two types of textual feedback: “things done well” and “things to do”.
- Individual feedback: contains team ID, student ID and the individual feedback given by tutor to a student.
- Peer assessment: contains student ID and the comments given by a student to his/her tutor.
- Peer feedback: contains student ID, contributions and comments. A ‘contribution’ is the mark given by all team members. A ‘comment’ is the textual feedback explaining the mark given. These comments are the most important resource in terms of analysis.

### 3.2 Text classification

As Chapter two discussed, text classification is the main work in this project. All of the comments given by peers will be classified into two categories: ‘actionable’ and ‘descriptive’.

Actionable is a precise term for quality feedback, because ‘actionable’ means the text contains executable suggestions which can help students identify their weaknesses and better understand future work. For example, “If he cannot finish the job, he will tell us in advance; however, if you are going to make some changes to the database or something important, it is better if you discuss it with us in advance”. This text contains meaning that suggests this student could improve their communication with others. Furthermore, innocently negative comments can also be regarded as actionable, because they imply a suggestion that a student should fix something. For example, “your programming skill is broken”, suggests that the student should learn more about programming to catch up.

Descriptive feedback means the text describes what a person has done without any executable suggestion. For example, “An excellent team leader; makes the project clearer for us which helps in planning the whole project” and “A great team member who keeps the team moving forward”.

The primary application of the results obtained from text classification is the selection of actionable feedback, and its presentation to tutors in order to save time, and enable them to
help the students who need to improve their performance. Furthermore, the classification results can be analysed and combined with team marks and individual marks to explore whether actionable feedback can improve the performance of students and whether the well organised teams tend to offer actionable feedback to each other.

### 3.2.1 Methodology

To realise text classification, two basic steps need to be addressed: text preprocessing and machine learning (Ahonen, Heinonen, Klemettinen, and Verkamo, 1997).

**Text preprocessing**

Machine learning algorithms are based on term filtering. Term filtering refers to finding the most typical words in a text (Ahonen, Heinonen, Klemettinen, and Verkamo, 1997). Therefore, texts are required to be preprocessed in order to make the term filtering results more typical. Some issues need be discussed before training the classifier:

- **Punctuation:** most feedback contains commas and periods (full stops). Normally, commas and periods cannot express any sentiment. Therefore, they should be filtered out. Potentially, exclamation marks and question marks can express sentiment; an exclamation mark has the meaning of praise and question mark can express blame. This means a text which contains an exclamation mark tends to be classified as descriptive, while text containing a question mark is likely to be actionable.

- **Numbers:** in this project, numbers are hard to include in machine learning, because the work which students assess each other on is not distinctly relevant to numbers. Therefore, numbers should be filtered out.

- **Case sensitivity:** case sensitivity means considering a word occurring at the beginning of a sentence can carry a different meaning to similar words which appear at other positions within a sentence in the text, but in lowercase. In addition, students may write words in uppercase for emphasis or simply for stylistic reasons. However, in this project, the amount of labelled data is not large enough to support case sensitivity, since case sensitivity will decrease the frequency of terms. For example, Communication, communication and COMMUNICATION will be recognised as three different terms. If one of the word forms is low frequency, it will be filtered. Actually, the frequency of this word is not very low, which means a word that can contribute to classification will be filtered out. Therefore, limited by the amount of labelled data in the current stage it has been decided to convert all words into lowercase.

- **Low frequency word:** a low frequency word should be filtered, because this kind of word is not sufficiently typical. It can cause overfitting, which can reduce the accurate rate of classification (Yang & Pedersen, 1997).

- **General words:** general words such as ‘I’, ‘we’, ‘you’, ‘do’ and ‘what’, etc. cannot express special meaning normally, therefore they should be filtered.
Stemming: stemming is a popular technique in natural language processing but with uncertain influence (McCallum & Nigam, 1998). Intuitively, the contribution of stemming to build this classification pattern is uncertain, because one of the important tasks of stemming is the elimination of the tense. For example, an original text might contain the word ‘assigned’, which contains the meaning of description, but it loses that meaning when the word is stemmed to ‘assign’. Therefore, an experiment should be implemented to find out if stemming can improve the classification results.

Machine learning algorithms

Four popular machine learning algorithms are used for classification: k-nearest neighbor (KNN), Naive Bayes, decision tree, and support vector machines (SVM). Chen, Huang, Tian, and Qu (2009) conducted a study about improving text classification methods, and established that the four machine learning algorithms mentioned above are the most popular text classification methods. This report focuses on application rather than improving these popular methods; therefore, the four methods are not dealt with in great detail.

3.2.2 Implementation

The KNIME Analytics Platform was used to implement the text preprocessed model and machine learning model. In Chapter two, the results obtained following a comparison of the various tools established that KNIME was in fact the most appropriate document analysis tool for use in this project; the following are the main advantage of this tool:

- It is easy to learn, because KNIME uses nodes to represent functions, and for each node there are detailed descriptions including the meaning of each parameter, the inputs, and outputs. Detailed and convenient documentation makes the software easier to use.
- It is powerful; each node can implement a complex function, and, for example, an ‘xml reader’ node can load an Excel file to an executable table directly. KNIME has more than 1000 nodes and can implement the classification using four different methods without searching for and downloading other packages.
- Easy communication; each node can be renamed, and a set of nodes can be packed as a module, therefore, users can understand how a specific node or set of nodes work easily via meaningful names of nodes and modules. This will facilitate easier communication between all interested parties in this project.

A workflow is designed using KNIME to experiment how to obtain the best classification results. The experiment includes two parts:

1. Investigation into whether text preprocessing can improve the classification results, and if stemming should be removed from the text preprocessing module, indicates the need for three experiments. The first one contains all of the text preprocessing functions. The second one contains the same text preprocessing functions, but with stemming omitted. In the last one, the text will not be preprocessed. KNIME cannot be set up to retain only some specific
punctuation characters such as exclamation marks and question marks. Removing all punctuation characters will result in losing potentially useful data; however, commas and periods are useless with uncertain influence. Due to time restrictions, the experiment involving the removal of punctuation to improve classification results was not conducted; however, for the purpose of this project, it is assumed that the removal of punctuation will provide better classification results.

2. Determining which of the four machine learning algorithms (decision tree, SVM, KNN or Naive Bayes) can provide the best classification results.

A KNIME workflow can compare four machine learning methods in the same time, but the different text preprocessing groups cannot be experimented in one workflow. The idea is creating a workflow containing all of the text preprocessing functions and four machine learning methods, and output the performance. Removing the correspondent text preprocessing functions if it is not included in the experiment.

The workflow contains three modules: data import, preprocessing, and predictive modelling and scoring. Each icon represents a function called a ‘node’ in KNIME; a black triangle on the right side of a node indicates the node can output a result, while a black triangle on the left side indicates the node accepts the output of other nodes as its input. Two nodes can be connected by a line. The words above a node is the name of the node, which can be used to search the node in KNIME, while the words below a node is the description of a node.

Data import: Figure 2 shows how this module works. First of all, 2 ‘XLS reader nodes’ read the XLS files directly. The two XLS files contain the actionable and descriptive feedbacks labeled manually. The configuration can choose which sheet, rows and columns are read. The default setting is used as all XLS files are preprocessed manually. Secondly, ‘string to document nodes’ can convert the temporary data to a document with a given category. A ‘document’ is a special format in KNIME and texts must be stored in this format to enable KNIME to deal with the texts. All descriptive texts are assigned the category ‘des’ and actionable texts are assigned the category ‘act’. ‘Column filter nodes’ are used to filter the temporary data and output a table which contains only the document column. Finally, a ‘concatenate node’ can output a combined document which contains two documents and as the input for next module text preprocessing.
Text preprocessing: this module (shown in Figure 3) includes six text preprocessing nodes: punctuation erasure, N chars filter, number filter, case converter, stop word filter, and snowball stemming. Following description comes from the documentation of KNIME.

- **Punctuation erasure:** this removes all punctuation characters from terms contained in the input documents.
- **N chars filter:** this filters all the terms contained in the input documents with less than the specified number (N) of characters.
- **Number filter:** this filters all the terms contained in the input documents that consist of digits, including decimal separators ',', or '.' and possible leading '+' or '-'.
- **Case converter:** this converts all the terms contained in the input documents to lowercase or uppercase.
- **Stop word filter:** this filters for terms in the input documents which are contained in the specified stop word list. The node provides built-in stop word lists for various languages. A specific stop word list was not created for this experiment; the built-in default stop word list for English was used (Appendix 3 shows the stop word lists).
- **Snowball stemmer:** this is the stemming process which is based on the snowball stemming library, which means this function can do stemming process according to the rules set in the snowball stemming library.

The order of functions cannot influence the test results, as each function is independent of the others. The last node ‘term filtering’ is a meta-node offered by KNIME. A meta-node is collapsed by a group of nodes. For example all of the text preprocessing nodes can be collapsed into a meta-node because the whole process they achieved can be regarded as a sub-workflow. There is no actual influence on collapsing many nodes into a meta-node, while using meta-node is just for looking clean. The ‘term filtering’ meta-node can compute...
the term frequency of each term according to each document and the output of term filtering is the input of next module predictive modelling and scoring.

If a particular node is excluded from an experiment, the connecting line in question is simply eliminated, and a new one drawn. For example, in an experiment which does not contain ‘stemming’, the line between ‘stop word filter’ and ‘snowball stemmer’ and the line between ‘snowball stemmer’ and ‘term filtering’ are eliminated, and a new line connecting ‘stop word filter’ and ‘term filtering’ is drawn. Figure 4 shows the results of removing the ‘snowball stemmer’ node.

Predictive modelling and scoring: the module (shown in Figure 5) implements four machine learning methods. Firstly, the ‘transformation meta-node’ takes the output of ‘Term Filtering’ as input, this can use a vector to represent each text, which is preparation for machine learning. Then, the ‘partitioning node’ divides all of the data into either the ‘training set’ or ‘testing set’. This model was set up with a random 70% of data selected as ‘training set’ and the rest of the data as ‘testing set’. This can ensure that for each experiment the training sets and test sets are different. The training set was used to train the four machine learning methods (the three ‘leaner nodes’) and then the trained model (the three ‘predictor’ node) was used to label the test set, which results in the ‘scorer’ node being able to output a number of accurate statistics. K Nearest Neighbors is different with other three methods, a ‘KNN node’ can take the training set and testing set and output the predicted results directly. Here is not going to give a theoretical reason.
Figure 5: Predictive modelling and scoring module in KNIME
This chapter presents the choice of measurements, experiment results, and discussion of the results, as well as the rationale for choice of the best model.

In total, 252 feedback items were manually labelled (See Appendix 4 for sample), which consisted of 99 actionable feedback items and 153 descriptive feedback items. These labelled feedback items were then used to train and test the models. The experiments compared the three text preprocessing methods and the four machine learning methods.

### 4.1 Choice of metrics

Three metrics were calculated to judge the performance of the models: the accuracy rate, recall rate, and precision rate. All three metrics are commonly used in the evaluation of classification methods (Nguyen & Armitage, 2008).

**Classification accuracy rate:** this is the rate at which a model is able to successfully classify a text.

**Recall rate:** this is the fraction of relevant instances that are retrieved. Because actionable feedback can potentially yield more information, the recall rate is only applied to actionable feedback. For example, an 80% recall rate indicates that 80% of actionable feedback in the test set is recognised successfully.

**Precision rate:** this is the fraction of retrieved instances that are relevant. As in the case of the recall rate, the precision rate is only applied to actionable feedback. For example, a 90% precision rate indicates that 90% of all texts classified as actionable are indeed found to be actionable.

<table>
<thead>
<tr>
<th></th>
<th>Predicted as actionable</th>
<th>Predicted as descriptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actionable texts</td>
<td>29</td>
<td>7</td>
</tr>
<tr>
<td>Descriptive texts</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 2 shows an example of a confusion matrix; in this example, the accuracy rate is \( \frac{29 + 9}{29 + 9 + 2 + 9} = 77.5\% \). The recall rate for actionable feedback is \( \frac{9}{2 + 9} = 81.8\% \) and the precision rate for actionable feedback is \( \frac{9}{7 + 9} = 56.2\% \).
The factor of 'time consumed' refers to another commonly employed metric for the evaluation of classification methods (Szczesniak, 1963). However, for the purpose of this project, all the methods considered are able to classify 1000 pieces of text within a few seconds. As the course COMP3100 probably only generates approximately 3800 feedback items each semester, and all methods considered have the capacity to process such a volume in seconds, the factor of 'time consumed' is not taken into account as part of the selection process. But in future, this classifier will be used to classify many courses data, which means time consumed should not be ignored. Due to the limitation of time and the tool, time consumed need to be measured in future work.

4.2 Experiment results.

The following three tables present the experiment results. Each table corresponds to a particular metric. Experiments have been implemented five times, row 1-5 indicates the five experiments results respectively. The bottom row ‘AVE’ presents the average results.

Table 3: Accuracy rate for Decision Tree, SVM, KNN and Naive Bayes with three different text preprocessing approaches

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>72.40%</td>
<td>77.60%</td>
<td>75.00%</td>
<td>71.10%</td>
<td>73.00%</td>
<td>66.80%</td>
<td>69.20%</td>
<td>69.20%</td>
<td>72.40%</td>
<td>69.20%</td>
<td>66.80%</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>76.30%</td>
<td>71.10%</td>
<td>76.30%</td>
<td>77.60%</td>
<td>79.00%</td>
<td>76.80%</td>
<td>67.10%</td>
<td>64.50%</td>
<td>76.30%</td>
<td>66.00%</td>
<td>61.80%</td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>72.40%</td>
<td>77.60%</td>
<td>69.70%</td>
<td>71.10%</td>
<td>71.40%</td>
<td>77.60%</td>
<td>69.20%</td>
<td>66.00%</td>
<td>61.80%</td>
<td>71.10%</td>
<td>57.90%</td>
<td>68.40%</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>72.40%</td>
<td>77.60%</td>
<td>75.00%</td>
<td>72.40%</td>
<td>78.90%</td>
<td>77.40%</td>
<td>65.80%</td>
<td>65.80%</td>
<td>61.80%</td>
<td>69.70%</td>
<td>61.80%</td>
<td>64.50%</td>
</tr>
<tr>
<td><strong>AVE</strong></td>
<td>72.40%</td>
<td>74.20%</td>
<td>73.10%</td>
<td>74.48%</td>
<td>76.50%</td>
<td>69.50%</td>
<td>62.90%</td>
<td>62.64%</td>
<td>69.70%</td>
<td>64.74%</td>
<td>60.78%</td>
<td>67.10%</td>
</tr>
</tbody>
</table>

*TP - applying text preprocessing. NS - applying text preprocessing except stemming. NTP - text processing not applied.

Table 4: Recall rate of actionable feedback for Decision Tree, SVM, KNN and Naive Bayes with three different text preprocessing approaches

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>62.90%</td>
<td>60.60%</td>
<td>57.10%</td>
<td>54.50%</td>
<td>51.50%</td>
<td>62.10%</td>
<td>14.30%</td>
<td>10.90%</td>
<td>25.00%</td>
<td>14.30%</td>
<td>23.20%</td>
<td>25%</td>
</tr>
<tr>
<td>SVM</td>
<td>50%</td>
<td>58.60%</td>
<td>57.10%</td>
<td>53.80%</td>
<td>65.60%</td>
<td>78.60%</td>
<td>7.70%</td>
<td>6.90%</td>
<td>39.30%</td>
<td>34.60%</td>
<td>20.70%</td>
<td>42.90%</td>
</tr>
<tr>
<td>KNN</td>
<td>68.60%</td>
<td>64.90%</td>
<td>54.50%</td>
<td>62.90%</td>
<td>56.80%</td>
<td>69.60%</td>
<td>14.30%</td>
<td>10.90%</td>
<td>15.20%</td>
<td>57.10%</td>
<td>35.10%</td>
<td>39.40%</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>58.10%</td>
<td>56.70%</td>
<td>65.60%</td>
<td>67.70%</td>
<td>66.70%</td>
<td>66.80%</td>
<td>16.10%</td>
<td>20%</td>
<td>15.60%</td>
<td>38.70%</td>
<td>33.30%</td>
<td>32.20%</td>
</tr>
<tr>
<td><strong>AVE</strong></td>
<td>62.10%</td>
<td>59%</td>
<td>50%</td>
<td>65.30%</td>
<td>66.70%</td>
<td>75.80%</td>
<td>6.90%</td>
<td>16.70%</td>
<td>45.50%</td>
<td>37.60%</td>
<td>35.10%</td>
<td>30.50%</td>
</tr>
</tbody>
</table>

Table 5: Precision rate of actionable feedback for Decision Tree, SVM, KNN and Naive Bayes with three different text preprocessing approaches

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>73.30%</td>
<td>83.30%</td>
<td>69.60%</td>
<td>76%</td>
<td>85%</td>
<td>69.60%</td>
<td>83.30%</td>
<td>70%</td>
<td>100%</td>
<td>83.50%</td>
<td>57.90%</td>
<td>63.60%</td>
</tr>
<tr>
<td>SVM</td>
<td>72.20%</td>
<td>63%</td>
<td>72.70%</td>
<td>73.70%</td>
<td>67.90%</td>
<td>84.60%</td>
<td>66.70%</td>
<td>100%</td>
<td>91.70%</td>
<td>74.10%</td>
<td>54.50%</td>
<td>75%</td>
</tr>
<tr>
<td>KNN</td>
<td>70.60%</td>
<td>85.70%</td>
<td>69.20%</td>
<td>71%</td>
<td>80.80%</td>
<td>80.80%</td>
<td>83.30%</td>
<td>100%</td>
<td>85.30%</td>
<td>74.10%</td>
<td>61.90%</td>
<td>76.50%</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>69.20%</td>
<td>78.60%</td>
<td>70%</td>
<td>77.80%</td>
<td>76.90%</td>
<td>75.90%</td>
<td>100%</td>
<td>76%</td>
<td>71.40%</td>
<td>7%</td>
<td>92.60%</td>
<td>68.70%</td>
</tr>
<tr>
<td><strong>AVE</strong></td>
<td>69.20%</td>
<td>78.10%</td>
<td>69.10%</td>
<td>65.50%</td>
<td>80%</td>
<td>95.60%</td>
<td>66.70%</td>
<td>100%</td>
<td>100%</td>
<td>47.10%</td>
<td>57.90%</td>
<td>62.50%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>70.90%</td>
<td>77.10%</td>
<td>68.12%</td>
<td>72.80%</td>
<td>78.12%</td>
<td>80.70%</td>
<td>80.00%</td>
<td>90.00%</td>
<td>89.20%</td>
<td>70.72%</td>
<td>56.90%</td>
<td>68.86%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
<th>TP</th>
<th>NS</th>
<th>NTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>70.90%</td>
<td>77.10%</td>
<td>68.12%</td>
<td>72.80%</td>
<td>78.12%</td>
<td>80.70%</td>
<td>80.00%</td>
<td>90.00%</td>
<td>89.20%</td>
<td>70.72%</td>
<td>56.90%</td>
<td>68.86%</td>
</tr>
<tr>
<td>KNN</td>
<td>70.90%</td>
<td>77.10%</td>
<td>68.12%</td>
<td>72.80%</td>
<td>78.12%</td>
<td>80.70%</td>
<td>80.00%</td>
<td>90.00%</td>
<td>89.20%</td>
<td>70.72%</td>
<td>56.90%</td>
<td>68.86%</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>70.90%</td>
<td>77.10%</td>
<td>68.12%</td>
<td>72.80%</td>
<td>78.12%</td>
<td>80.70%</td>
<td>80.00%</td>
<td>90.00%</td>
<td>89.20%</td>
<td>70.72%</td>
<td>56.90%</td>
<td>68.86%</td>
</tr>
</tbody>
</table>
4.2 Discussion

Table 3 shows the highest classification accuracy rate was obtained using SVM without any text preprocessing (83.12%). It was also discovered that text preprocessing reduces the accuracy rate when using SVM, KNN and Naive Bayes. Only in the case of using Decision Tree was there any discernible improvement in performance as a result of text preprocessing (1.06%).

Table 4 shows the highest recall rate was also obtained using SVM without text preprocessing (73.78%), and the precision rate of the same model was 80.7% (see table 5); the second highest, after KNN. The precision rates obtained for KNN were 80.00%, 90.00% and 89.28%, but the recall rates were unacceptable (11.86%, 12.70% and 28.12%). Except there is an extremely high demand of high precision rate, KNN cannot be used to train a classifier for this project, because the results of other two metrics are not good enough. This indicates that SVM can be used for classification in this project. Therefore, SVM without text preprocessing is considered to be the best model for classifying the peer assessment feedback.

The fact that text preprocessing did not improve the classification results in the experiments was unexpected, because according to the literature review, text preprocessing is normally helpful for classification (Isa, Lee, Kallimani, and Rajkumar, 2008). Here are two potential reasons why this might be: firstly, the text preprocessing module offered by KNIME cannot achieve the best preprocessing results. For example, the text preprocessing module cannot only remove commas and periods and retain question and exclamation marks. In addition, the module only supports the Snowball library for stemming. This is a limitation of the experiment software used, which indicates that the experiment should be implemented using alternative and better software in the future. Secondly, the experiment did not cover all the possibilities; there are six functions in the text preprocessing module. Therefore, in total, there are 64 possibilities ($2^6 = 64$). Due to time restrictions, the experiments covered only stemming containing the most uncertain influences rather than all possibilities. Ideally, all of the remaining possibilities should be investigated in future experiments.
5. Application

This chapter presents the implementation of the chosen model and the accuracy rate of the SVM classification method by mutual check, along with some findings extracted from the modelling results. In addition, some potential applications are discussed and analysed.

5.1 Implementation of the chosen model

The chosen model has been implemented in KNIME. The workflow contains two modules: data import, and predictive modelling and scoring. The basic idea is using the manually labelled actionable feedback and descriptive feedback to train a SVM classifier, then using the trained classifier to classify the raw data. The main functions are almost the same as the implementation of the model comparison. However, there are three main differences:

1. Import data: this module is shown in Figure 6. Three documents are imported: they are labelled actionable feedback, labelled descriptive feedback, and raw data. These two labelled data are same with the data used in the experiment. The three ‘XSL Reader’ nodes also can assign categories to the data: descriptive feedbacks are given a category ‘des’, actionable feedbacks are given a category ‘act’, while the raw data are given a category ‘unlabeled’. These categories will be used in SVM classifier module. The output of this module is a document which contain all of the feedbacks.

2. Text preprocessing: no preprocessing is applied; the original texts are imported for machine learning directly, which means the output of data import module is the input of SVM classifier module.

3. SVM classifier: this module is shown in Figure 7. All of the 252 labelled feedback items are used to train an SVM classifier, which subsequently classifies the imported raw data. The ‘document vector node’ use a vector to represent each text, which is preparation for machine learning. Two ‘Row filter’ nodes can divide the data into training set and raw data according to their different categories. Finally, the workflow outputs an XLS file which contains the original text along with its category.
Figure 6: Data import module.

Figure 7: SVM classifier module

Figure 8 shows a sample of the output .xls file after implementing KNIME. It is worth noting that the file does not include certain information, lost in the process; in this case, the student ID and the mark given by other peers. KNIME is able to output the category of a feedback item, but cannot indicate that a particular feedback item belongs to a particular student. That because ‘document’ is the main format to store texts in KNIME, a series of operations in SVM classifier module cannot run if the document contains other information besides the texts which need to be classified.
Figure 8: Sample of the .xls file output by KNIME, where no student ID column and contribution column is compared with Appendix 2.

In order to solve this problem, MATLAB was implemented, adding categories as a new column in the original peer assessment data, as shown in Appendix 2. For example, in the case of a comment such as “great contribution in team”, the comment was searched for in the output file of KNIME, which returned the prediction result – “Des”, and then “Des” was added as a new column in the original peer assessment. Figure 9 shows a sample of the final results of text classification.
Choosing Matlab to solve this problem is not a good choice, because this is not a complex mathematical problem. But Matlab is the only tool installed in author’s desktop that can be used to solve this problem. Due to time restriction, other easier ways did not attempted, for example Python.

The accuracy rate of the classification results were also checked mutually, taking every 10th comment as a sample (i.e. line 10, 20, 30… 650 and 660). If a line did not contain a comment, it was passed.

Table 6: Confusion matrix of mutual check

<table>
<thead>
<tr>
<th></th>
<th>Predicted as actionable</th>
<th>Predicted as descriptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actionable texts</td>
<td>39</td>
<td>2</td>
</tr>
<tr>
<td>Descriptive texts</td>
<td>5</td>
<td>11</td>
</tr>
</tbody>
</table>

In total, 59 feedback items were selected as a sample for mutual checking. The results can be calculated according to the confusion matrix shown in Table 6. The accuracy rate
is \( \frac{39+11}{59} = 84.7\% \); the recall rate for actionable feedback is \( \frac{11}{5+11} = 68.8\% \), and the precision rate for actionable feedback is \( \frac{2}{2+11} = 84.6\% \). The results produced by the KNIME software were 83.1% for accuracy rate, 73.8% for recall rate, and 80.7% for precision rate. There is no significant deviation between the software measured results and mutual check results. Therefore, the classification results are considered to be acceptable.

5.2 Application of the classification results

Followings are some findings extracted from the modelling results. In addition, some potential applications are discussed and analysed.

5.2.1 The ratio of actionable and descriptive feedback

The classification model was used to classify week 4 and week 6 peer assessment data. The first step was the analysis of the ratio of actionable and described feedback. In week 4 data, there are 181 actionable feedback items and 419 descriptive feedback items. In week 6 data, there are 165 actionable feedback items and 399 descriptive feedback items.

With reference to Figure 10, notably, the proportion of descriptive feedback supplied by students is considerably greater than actionable feedback provided. The performance of an individual student is indicated by the individual mark assigned; however, at this moment in time, access to these individual marks is not available, therefore, it is cannot be asserted that actionable feedback can improve students’ performance. But a research shows that the feedbacks containing suggestion does improve students’ performance generally (Winstone, Nash, Rowntree and Menezes, 2015). Therefore, tutors should encourage students to provide more actionable feedback.

5.2.2 Identifying problematic students

This is a direct application of the classification model, which involves presenting the classified feedback to tutors regarding which students receive actionable feedback, to
enable tutors to work more closely with those individual students to improve their contribution. Figure 11 presents a sample of a categorised peer assessment, tutor can know which students received actionable feedback quickly.

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Contribution Comment</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student_000</td>
<td>Should arrive to tutor meetings on time</td>
<td>Act</td>
</tr>
<tr>
<td>Student_000</td>
<td>The user flow discussion was good for helping us get a clear idea</td>
<td>Act</td>
</tr>
<tr>
<td>Student_000</td>
<td>Turned up late to tutor meeting despite living on campus</td>
<td>Act</td>
</tr>
<tr>
<td>Student_001</td>
<td>Should strive to ensure we adhere to the agile process</td>
<td>Act</td>
</tr>
<tr>
<td>Student_001</td>
<td>Setting up the starterkit so it worked on our computers</td>
<td>Act</td>
</tr>
<tr>
<td>Student_001</td>
<td>Communication skills need a little work</td>
<td>Act</td>
</tr>
<tr>
<td>Student_002</td>
<td>Should consider a more consistent sleep schedule</td>
<td>Act</td>
</tr>
<tr>
<td>Student_002</td>
<td>The user flow discussion helped us get a clearer idea</td>
<td>Act</td>
</tr>
<tr>
<td>Student_002</td>
<td>Communication skills need work</td>
<td>Act</td>
</tr>
<tr>
<td>Student_002</td>
<td>Should be more involved in organizing future coding days</td>
<td>Act</td>
</tr>
<tr>
<td>Student_003</td>
<td>The previous starterkit was a bit complicated - let's take it</td>
<td>Act</td>
</tr>
<tr>
<td>Student_004</td>
<td>Lin has helped us to start the transform from the old plan</td>
<td>Act</td>
</tr>
</tbody>
</table>

Figure 11: Example of a categorised peer assessment

5.2.3 Identifying suggestions in a long text

This is a potential application; in the case of COMP3100, most feedback consists of short sentences. The model which was trained as part of this project can classify the feedback directly without splitting it into many sentences. For some courses, if students receive long textual feedback, this model can identify the sentences that contain suggestions, through the addition of a ‘splitting function’ which divides a long piece of text into its constituent sentences.

5.2.4 The relationship with quality feedback and group mark

The top 3 groups (highest grades) and the bottom 3 groups (lowest grades) were identified and their respective peer feedback data was analysed. In week 6 data, the percentage of quality feedback in the low-grade groups is 30% (9 out of 30), while in the high-grade groups it is 33% (10 out of 33). The results do not show any obvious relationship between group grades and quality feedback. Another hypothesis is that if team members gave each other actionable feedback, the team grades would increase; however, currently insufficient data exists to analyse such a trend, but it is certainly worth considering for inclusion in future studies.
5.2.5 Common student performance-related problems

The use of frequency distribution can help to identify the common problems students suffer from with regard to their performance so that lecturers can schedule special activities to address these issues. The most effective method would be through the development of an artificial intelligence based model capable of interpreting the actionable feedback items; however, due to time restrictions this is currently not feasible.
6. Conclusion and recommendations for future work

This Chapter summarises the main contributions made by this project, and discusses possible areas for further research in the future, in order to expand this project.

6.1 Conclusion

The aim of this project is to investigate the implementation of document analysis as part of peer assessment to identify valuable information which can be used to improve learning outcomes.

Firstly, a UML class diagram was created in order to gain a better understanding of the subject. In addition, the UML class diagram can also contribute to the construction of a database as a repository. The requirements and criteria for constructing a suitable database were researched and analysed; however, the preprocessed peer assessment data provided by Dr. Flint was in a format which allowed it to be used directly and, therefore, construction of a database was not required currently.

Secondly, text classification is the core of document analysis in this project. Experiments were conducted to ascertain if text preprocessing can benefit text classification, and if the stemming process in text preprocessing was required. In addition, an evaluation of four machine learning methods (decision trees, SVM, KNN and Naive Bayes) was conducted in order to establish which method would provide the best classification results.

Thirdly, 252 data items were manually labelled, divided into a training set and a test set randomly. The highest accuracy rate and precision rate, and an acceptable recall rate, were obtained using SVM. In addition, experiment results indicated that text preprocessing did not contribute to text classification in this project.

Finally, the SVM method was implemented, and the classified peer assessment feedback for a two-week period was processed. Accuracy, recall and precision rates of the classification results were checked manually. The manual check results confirmed that the text classification results produced by KNIME were good. Some useful information was extracted from the classification results, such as the fact that only 30% of feedback is quality, and the actionable feedbacks can be sent to tutor so that tutor can work more closely with those individual students to improve their contribution. Furthermore, some potential applications based on the classification were identified, such as training a pattern to understand the classified feedback at a deeper level, and tracking students that receive quality feedback to establish if quality feedback can improve student performance.
6.2 Future work

It is important to note that text preprocessing includes six functions: punctuation removal, low frequency word filtering, number filtering, case converting, general word filtering, and stemming. However, in this project, as a result of time constraints, only the stemming function was included in experiments; the other functions being treated together as a module. Therefore, any future work should focus on the testing of all six functions independently in order to obtain more comprehensive and informative results. Secondly, the time consumed of the four different classification methods was not experimented and compared. The time consumed should be measured before implementing the classifier created in this project to a large amount data (more than 100,000). Lastly, ideally, the training of a new model which would be capable of gaining a more in-depth understanding of the classified texts would lead to outputting the common problems that students are suffering from.
References


Purchase, H. C., Colpoys, L., Carrington, D., & McGill, M. *UML CLASS DIAGRAMS.*


Appendix 1. A sample of peer assessment submitted by a student in COMP 3100.

<table>
<thead>
<tr>
<th>Team Name</th>
<th>TechLauncher Peer Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review</td>
<td>2016, Semester 2, week12</td>
</tr>
<tr>
<td>Tutor</td>
<td>Kyle Maher</td>
</tr>
<tr>
<td>Student ID</td>
<td>u5541673</td>
</tr>
<tr>
<td>Student Name</td>
<td>Zi Jin</td>
</tr>
</tbody>
</table>

**Instructions:**
In the table below, enter what you think each student's contribution (including your own) was during the reporting period as a percentage. The total MUST equal 100%.

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Team Member Name</th>
<th>Contribution</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>u5541673</td>
<td>Zi Jin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>u5541596</td>
<td>Qiang Duan</td>
<td></td>
<td>You must include some constructive feedback for every member of your team. If you don't provide feedback, your peer assessment will be considered invalid.</td>
</tr>
<tr>
<td>u5542269</td>
<td>Guanzhong Wang</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Enter comments for your tutor here. They will not be seen by any of your team members or clients.

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Team Member Name</th>
<th>Contribution</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>u5541673</td>
<td>Zi Jin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>u5541596</td>
<td>Qiang Duan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>u5542269</td>
<td>Guanzhong Wang</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0% This total must equal 100%
Appendix 2. The .xlsx files offered by Dr. Flint.

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Contribution</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student_000</td>
<td>0.3</td>
<td><a href="https://www.youtube.com/watch?v=APvodget9J4">Video Link</a></td>
</tr>
<tr>
<td>Student_000</td>
<td>0.33</td>
<td>The work on APIs, adding new pages, and being a team manager has been very good – let’s keep working in this direction.</td>
</tr>
<tr>
<td>Student_000</td>
<td>0.28</td>
<td>Missed the client meeting. Remember there are lots of other important things to do/learn.</td>
</tr>
<tr>
<td>Student_001</td>
<td>0.3</td>
<td>Hey, you’re doing ok I guess.</td>
</tr>
<tr>
<td>Student_001</td>
<td>0.33</td>
<td>Eric, you’ve done really well this week. I’m happy with all the DevOps activities you’ve achieved this week. Future development will benefit from this. It is my hope you keep it up.</td>
</tr>
<tr>
<td>Student_001</td>
<td>0.31</td>
<td>The work on keeping the website stable has been going well – great work.</td>
</tr>
<tr>
<td>Student_001</td>
<td>0.28</td>
<td>Demonstrated need to improve on weaknesses. Well done. Stable at every other aspect.</td>
</tr>
<tr>
<td>Student_002</td>
<td>0.2</td>
<td>Communication skills need a little work.</td>
</tr>
<tr>
<td>Student_002</td>
<td>0.23</td>
<td>Good job on search pages. Good to see motivation at the tutor meeting. Please keep it up - I think we can make a great website!</td>
</tr>
<tr>
<td>Student_002</td>
<td>0.23</td>
<td>All good.</td>
</tr>
<tr>
<td>Student_002</td>
<td>0.2</td>
<td>Needs to be more open to other perspectives and the business/market side of things.</td>
</tr>
<tr>
<td>Student_003</td>
<td>0.2</td>
<td>Could do more outside of meet ups.</td>
</tr>
<tr>
<td>Student_003</td>
<td>0.11</td>
<td>Looking at Trello, only completed one task compared to the many others that people were doing. Please motivate yourself to completing activities outside of mandatory meetings/reviews. Check Slack more frequently as well.</td>
</tr>
<tr>
<td>Student_003</td>
<td>0.05</td>
<td>Avoid pushing directly to master, and try offering to do pages you have confidence in building.</td>
</tr>
<tr>
<td>Student_003</td>
<td>0.24</td>
<td>Great team management.</td>
</tr>
<tr>
<td>Student_004</td>
<td>0.33</td>
<td>I did pay some unnecessary attention on how to make people to work when they lack of motivation for the team as well as himself. It seems the problem has been taken care of by the whole team. Let’s see before wk8 tutor meeting.</td>
</tr>
<tr>
<td>Student_004</td>
<td>0.37</td>
<td>She does a good job on connecting the team with the client. She is the one who will gather our feedback and progress and give to the client on time. Also, if the client has new requirement, she will tell us on time.</td>
</tr>
<tr>
<td>Student_005</td>
<td>0.33</td>
<td>Great contribution in team.</td>
</tr>
<tr>
<td>Student_005</td>
<td>0.37</td>
<td>Delivering quality work with efficiency. Really reliable and trustable to work with.</td>
</tr>
<tr>
<td>Student_005</td>
<td>0.34</td>
<td>I finish all task that assigned to me on time and efficiently. Giving feedback to Lin Li on time to make sure the client know the progress and has told the client what I really want to do.</td>
</tr>
<tr>
<td>Student_005</td>
<td>0.34</td>
<td>Tasks I do have been recorded on the management software we use.</td>
</tr>
</tbody>
</table>

- Teams
- Students
- TeamFeedback
- IndividualFeedback
- PeerAssessments
- PeerFeedback
Appendix 3. The build-in stop word lists in KNIME.

<table>
<thead>
<tr>
<th>after</th>
<th>becomes</th>
<th>end</th>
<th>good</th>
<th>k</th>
<th>n</th>
<th>our</th>
<th>seemed</th>
<th>think</th>
<th>were</th>
</tr>
</thead>
<tbody>
<tr>
<td>again</td>
<td>been</td>
<td>ended</td>
<td>goods</td>
<td>keep</td>
<td>necessary</td>
<td>out</td>
<td>seeming</td>
<td>thinks</td>
<td>what</td>
</tr>
<tr>
<td>against</td>
<td>before</td>
<td>ending</td>
<td>got</td>
<td>keeps</td>
<td>need</td>
<td>over</td>
<td>seems</td>
<td>this</td>
<td>when</td>
</tr>
<tr>
<td>all</td>
<td>began</td>
<td>ends</td>
<td>great</td>
<td>kind</td>
<td>needed</td>
<td>p</td>
<td>sees</td>
<td>those</td>
<td>where</td>
</tr>
<tr>
<td>almost</td>
<td>behind</td>
<td>enough</td>
<td>greater</td>
<td>knew</td>
<td>needing</td>
<td>part</td>
<td>several</td>
<td>though</td>
<td>whether</td>
</tr>
<tr>
<td>alone</td>
<td>being</td>
<td>even</td>
<td>greatest</td>
<td>know</td>
<td>needs</td>
<td>parted</td>
<td>shall</td>
<td>thought</td>
<td>which</td>
</tr>
<tr>
<td>along</td>
<td>beings</td>
<td>evenly</td>
<td>group</td>
<td>known</td>
<td>never</td>
<td>parting</td>
<td>she</td>
<td>thoughts</td>
<td>while</td>
</tr>
<tr>
<td>already</td>
<td>best</td>
<td>ever</td>
<td>grouped</td>
<td>knows</td>
<td>new</td>
<td>parts</td>
<td>should</td>
<td>three</td>
<td>who</td>
</tr>
<tr>
<td>also</td>
<td>better</td>
<td>every</td>
<td>grouping</td>
<td>I</td>
<td>new</td>
<td>per</td>
<td>show</td>
<td>through</td>
<td>whole</td>
</tr>
<tr>
<td>although</td>
<td>between</td>
<td>everybody</td>
<td>groups</td>
<td>large</td>
<td>newer</td>
<td>perhaps</td>
<td>showed</td>
<td>thus</td>
<td>whose</td>
</tr>
<tr>
<td>always</td>
<td>big</td>
<td>everyone</td>
<td>h</td>
<td>largely</td>
<td>newest</td>
<td>place</td>
<td>showing</td>
<td>to</td>
<td>why</td>
</tr>
<tr>
<td>among</td>
<td>both</td>
<td>everything</td>
<td>had</td>
<td>last</td>
<td>next</td>
<td>places</td>
<td>shows</td>
<td>today</td>
<td>will</td>
</tr>
<tr>
<td>an</td>
<td>but</td>
<td>everywhere</td>
<td>has</td>
<td>later</td>
<td>no</td>
<td>point</td>
<td>side</td>
<td>together</td>
<td>with</td>
</tr>
<tr>
<td>and</td>
<td>by</td>
<td>f</td>
<td>have</td>
<td>latest</td>
<td>nobody</td>
<td>pointed</td>
<td>sides</td>
<td>too</td>
<td>within</td>
</tr>
<tr>
<td>another</td>
<td>c</td>
<td>face</td>
<td>having</td>
<td>least</td>
<td>non</td>
<td>pointing</td>
<td>since</td>
<td>took</td>
<td>without</td>
</tr>
<tr>
<td>any</td>
<td>came</td>
<td>faces</td>
<td>he</td>
<td>less</td>
<td>noone</td>
<td>points</td>
<td>small</td>
<td>toward</td>
<td>work</td>
</tr>
<tr>
<td>anybody</td>
<td>can</td>
<td>fact</td>
<td>her</td>
<td>let</td>
<td>not</td>
<td>possible</td>
<td>smaller</td>
<td>turn</td>
<td>worked</td>
</tr>
<tr>
<td>anyone</td>
<td>cannot</td>
<td>facts</td>
<td>here</td>
<td>lets</td>
<td>nothing</td>
<td>present</td>
<td>smallest</td>
<td>turned</td>
<td>working</td>
</tr>
<tr>
<td>anything</td>
<td>case</td>
<td>far</td>
<td>herself</td>
<td>like</td>
<td>now</td>
<td>presented</td>
<td>so</td>
<td>turning</td>
<td>works</td>
</tr>
<tr>
<td>anywhere</td>
<td>cases</td>
<td>felt</td>
<td>high</td>
<td>likely</td>
<td>nowhere</td>
<td>presenting</td>
<td>some</td>
<td>turns</td>
<td>would</td>
</tr>
<tr>
<td>are</td>
<td>certain</td>
<td>few</td>
<td>high</td>
<td>long</td>
<td>number</td>
<td>presents</td>
<td>somebody</td>
<td>two</td>
<td>x</td>
</tr>
<tr>
<td>area</td>
<td>certainly</td>
<td>find</td>
<td>high</td>
<td>longer</td>
<td>numbers</td>
<td>problem</td>
<td>someone</td>
<td>u</td>
<td>y</td>
</tr>
<tr>
<td>areas</td>
<td>clear</td>
<td>finds</td>
<td>higher</td>
<td>longest</td>
<td>o</td>
<td>problems</td>
<td>something</td>
<td>under</td>
<td>year</td>
</tr>
<tr>
<td>around</td>
<td>clearly</td>
<td>first</td>
<td>highest</td>
<td>m</td>
<td>off</td>
<td>put</td>
<td>somewhere</td>
<td>until</td>
<td>years</td>
</tr>
<tr>
<td>as</td>
<td>come</td>
<td>for</td>
<td>him</td>
<td>made</td>
<td>off</td>
<td>puts</td>
<td>state</td>
<td>up</td>
<td>yet</td>
</tr>
<tr>
<td>ask</td>
<td>could</td>
<td>four</td>
<td>himself</td>
<td>make</td>
<td>often</td>
<td>q</td>
<td>states</td>
<td>upon</td>
<td>you</td>
</tr>
<tr>
<td>asked</td>
<td>d</td>
<td>from</td>
<td>his</td>
<td>making</td>
<td>old</td>
<td>quite</td>
<td>still</td>
<td>us</td>
<td>young</td>
</tr>
<tr>
<td>asking</td>
<td>did</td>
<td>full</td>
<td>how</td>
<td>man</td>
<td>older</td>
<td>r</td>
<td>such</td>
<td>use</td>
<td>younger</td>
</tr>
<tr>
<td>asks</td>
<td>differ</td>
<td>fully</td>
<td>however</td>
<td>many</td>
<td>oldest</td>
<td>rather</td>
<td>sure</td>
<td>used</td>
<td>youngest</td>
</tr>
<tr>
<td>at</td>
<td>different</td>
<td>further</td>
<td>i</td>
<td>may</td>
<td>on</td>
<td>really</td>
<td>t</td>
<td>uses</td>
<td>your</td>
</tr>
<tr>
<td>away</td>
<td>differently</td>
<td>furthered</td>
<td>if</td>
<td>me</td>
<td>once</td>
<td>right</td>
<td>take</td>
<td>v</td>
<td>yours</td>
</tr>
<tr>
<td>b</td>
<td>do</td>
<td>do</td>
<td>furthering</td>
<td>important</td>
<td>member</td>
<td>one</td>
<td>right</td>
<td>taken</td>
<td>very</td>
</tr>
<tr>
<td>back</td>
<td>does</td>
<td>furthers</td>
<td>in</td>
<td>members</td>
<td>only</td>
<td>room</td>
<td>than</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>backed</td>
<td>done</td>
<td>g</td>
<td>interest</td>
<td>men</td>
<td>open</td>
<td>rooms</td>
<td>that</td>
<td>want</td>
<td></td>
</tr>
<tr>
<td>backing</td>
<td>down</td>
<td>gave</td>
<td>interested</td>
<td>might</td>
<td>opened</td>
<td>s</td>
<td>the</td>
<td>wanted</td>
<td></td>
</tr>
<tr>
<td>backs</td>
<td>down</td>
<td>general</td>
<td>interesting</td>
<td>more</td>
<td>opening</td>
<td>said</td>
<td>their</td>
<td>wanting</td>
<td></td>
</tr>
<tr>
<td>be</td>
<td>downed</td>
<td>generally</td>
<td>interests</td>
<td>most</td>
<td>opens</td>
<td>same</td>
<td>them</td>
<td>wants</td>
<td></td>
</tr>
<tr>
<td>became</td>
<td>downing</td>
<td>get</td>
<td>into</td>
<td>mostly</td>
<td>or</td>
<td>saw</td>
<td>then</td>
<td>was</td>
<td></td>
</tr>
<tr>
<td>because</td>
<td>downs</td>
<td>gets</td>
<td>is</td>
<td>mr</td>
<td>order</td>
<td>say</td>
<td>there</td>
<td>way</td>
<td></td>
</tr>
<tr>
<td>become</td>
<td>during</td>
<td>give</td>
<td>it</td>
<td>mrs</td>
<td>ordered</td>
<td>says</td>
<td>therefore</td>
<td>ways</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>given</td>
<td>its</td>
<td>much</td>
<td>ordering</td>
<td>second</td>
<td>these</td>
<td>we</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix 4. Two samples of descriptive feedbacks and actionable feedbacks labeled manually

Above is descriptive feedbacks sample

| A bit absent and hard to reach. Make yourself easier to contact. |
| A good leader. More patience and tolerance would be better. Break down the whole task into smaller pieces and allocate them to us with explicit goals. |
| A very kind teammate. Did a great job last semester. We may need you to work a little bit faster due to our tight schedule this semester. |
| A very kind teammate. However, you may need to put more effort on our project. |
| A very kind teammate. However, you may need to put more effort on our project. |
| Able to see some pusher to our git repository. |
| According to the plan you finished the task of html, it’s good. And in the next sprint we still have lots of work to do together, so keep struggling and try to make the best application. |
| Always contribute a lot on our project; Always attend the meeting; If he can not finish the job, he will tell us in advance; However, if you are going to make some changes to database or something important, it is better if you can discuss with us in advance. |
| Attended the meeting with Brendan Loo Ge and provided useful feedback and critical points. Should regularly check on and participate in the team’s slack channel as often as possible due to the lack of other nodes of communication. Should not hesitate in discussing project related issues with team mates. |

Above is actionable feedbacks sample
Appendix 5. The final project description as agreed upon by the student and supervisor(s), detailing tasks and expected outcomes.

Over the past couple years, Dr. Shayne Flint and Dr. Lynette Johns-Boast collected a large amount of data from students taking the various group project courses that related to peer and tutor assessment and grades. This data is stored in a variety of csv files which do not enable them to make use of it. They are not sure of the relationships in the data and what they might be telling them. The second half of the project is vague at this stage and will be determined more precisely once they have a better understanding of those relationships.

Following is the detailing tasks:
1. Developing a UML information model of the data and its relationships
2. Based on the data model, decide an appropriate repository to store all of the data
3. Load the data into the repository and verify.
4. Conduct qualitative data analysis of the qualitative data contained in the repository

The expected outcomes are:
1. Creating A UML information model can represent all of the data related with the course which contain peer assessment.
2. Constructing an appropriate repository and load all of the data into the repository.
3. Implementing qualitative data analysis of the qualitative data contained in the repository.
INDEPENDENT STUDY CONTRACT

SECTION A (Students and Supervisors)

UnitID: a5541673
Surname: Jin
First Names: Zi
Project Supervisor (may be external): Kinette Johns-Beast
Course Supervisor (if applicable): 
Course Code, Title and Unit: COMP4560 Programs and Courses
Semester: [ ] S1 [ ] S2 Year: 2016

Project Title:
Using peer assessment data to improve teaching and learning outcomes

Learning Objectives:
1. Demonstrate an understanding of data modelling in the UML
2. Develop qualitative analysis skills
3. Further develop database skills and knowledge

Project Description:
Over the past couple of years, we have collected a large amount of data from students taking the various group project courses that relates to peer and tutor assessment and grades. We wish to be able to use this data to help us improve outcomes for students taking these courses. Currently it is stored in a variety of CSV files which do not enable us to make use of it. As we are not sure of the relationships in the data and what they might be telling us, the second half of the project is vague at this stage and will be determined more precisely once we have a better understanding of those relationships.

We envisage that this project will be conducted in two stages and will require you to do some or all of the following:
Stage 1
Develop a UML information model of the data and its relationships
Based on the data model, decide an appropriate database technology and create a database that replicates the model
Load the data into the database and verify

Research School of Computer Science

Form updated Jun-12
Stage 2

Conduct qualitative data analysis of the qualitative data contained in the database

Determine an appropriate visualisation method for the data

Implement the visualisation

ASSESSMENT (as per course’s project rules web page, with the differences noted below):

<table>
<thead>
<tr>
<th>Assessed project components:</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report: name style</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e.g. research report, software description...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Artefact name kind</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e.g. software, user interface, robot...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presentation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MEETING DATES (IF KNOWN):

STUDENT DECLARATION: I agree to fulfil the above defined contract:

Signature ____________________________ Date ____________

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project.

Signature ____________________________ Date ____________

REQUIRED DEPARTMENT RESOURCES:

SECTION C (Course coordinator approval)

Research School of Computer Science
SECTION D (Projects coordinator approval)

Signature       Date

Signature       Date
Appendix 7. Description of software / artefacts produced.

I created three artefacts:

- A UML class diagram by Papyrus. (all of the work is achieved by myself)
  - PeerAssessment.di: 2016/8/16 14:29
  - PeerAssessment.jpeg: 2016/10/27 0:45
  - PeerAssessment.notation: 2016/10/26 21:39
  - PeerAssessment.uml: 2016/10/11 0:36

File PeerAssessment.di can be opened by Papyrus, which the other two files .notation and .uml should be stored with .di file in same folder.

Peerassessment.jpeg is the image of this class diagram.

- Two workflows created by KNIME Analytics Platform.
  - Actionable.xlsx: 2018/10/9 14:56
  - descriptive.xlsx: 2018/10/9 14:56
  - Document_Classification.knwf: 2018/10/27 0:00
  - Model_comparsion.knwf: 2018/10/27 0:01
  - readme.txt: 2018/10/27 0:31
  - results of week 4 data.xlsx: 2016/10/26 23:51
  - results of week 6 data.xlsx: 2016/10/26 23:48
  - week04_export.xlsx: 2016/10/9 22:40
  - week06_export.xlsx: 2016/9/27 20:53

Actionable.xlsx and descriptive.xlsx are 252 data manually labeled by myself. These data are used to train and compare the different classification methods.

Model_comparsion.knwf is the workflow used to implement the experiment in chapter 3. Most work is achieve by myself, only two mate-nodes and one module in this workflow are offered by KNIME community (I cited that in my thesis and readme file.)

Document_Classification.knwf is the workflow used to implement the selected SVM classifier. Most work is achieve by myself, only two mate-nodes in this workflow are offered by KNIME community (I cited that in my thesis and readme file.)

Results of week 4 data.xlsx and results of week 6 data.xlsx are the output of Document_Classification.knwf

Week04_export.xlsx and week06_export.xlsx are the raw data offered by Dr. Flint.

- A .m file created by Matlab. (all of the work is achieved by myself)
  - output.m: 2016/10/27 0:39
  - Readme.txt: 2016/10/27 0:36
  - results of week 4 data.xlsx: 2016/10/26 23:51
  - results of week 6 data.xlsx: 2016/10/26 23:48
  - week 4 final.xlsx: 2016/10/27 0:18
  - week 6 final.xlsx: 2016/10/27 0:20

Output.m is the code to improve the output results of KNIME workflow.

Results of week 4 data.xlsx and results of week 6 data.xlsx are same with above.

Week 4 final.xlsx and week 6 final.xlsx is the final output of this project.

There are four readme files, one is overview for all artefacts, and other three focus on the operations.

- **Overview**
  I declare that, to the best of my knowledge, this thesis is my own original work and does not contain any material previously published or written by another person except where otherwise indicated

The artefacts contain three files. There is a readme in each file to let you know how to implement the artefacts.

**KNIME**: Contains two workflows and the labeled data. The workflow is used to compared different classification methods and implement a trained classifier to classify raw data.

**Papyrus**: Contains two UML class diagrams.

**Matlab**: A Matlab code to process the KNIME output to get a full output.

- **Papyrus**
  This file contains two UML class diagrams I created. If you just want to see the diagram, you can open those two images Peerassessment.jpeg and Survey.jpeg

  If you want to run those two UML class diagrams, you need to install Papyrus firstly, https://eclipse.org/papyrus/download.html.

  After finishing the installation, just open the .di file by Papyrus. The details of the peer assessment diagram is presented in my report.

- **KNIME**
  1. **Model_comparision** (if you are not going to repeat my experiments, just ignore this workflow) // In this workflow the modules Term-filtering, transformation and text preprocessing are offered by KNIME community. Also use node scorer to get confusion matrix is also offered by KNIME community.

  Running this workflow need to install KNIME analysts platform firstly. https://www.knime.org/downloads/overview?quicktabs_knimed=2#quicktabs-knimed

  I suggest to install the "KNIME Analytics Platform + all free extensions” version

  This workflow is used to compare four classification methods: KNNs, SVM, Decision tree and Naive Bayes.

  How to use it: 1. installing KNIME analysts platform

    2. Click file- Import KNIME workflow, click Browse to choose this workflow and click finish.

    3. Double-click XLS Reader "import the descriptive feedbacks"
4. In the top of the configuration, select file to read: use Browse to choose the file "descriptive.xls" then click OK

5. Double-click XLS Reader "import the actionable feedbacks"

6. In the top of the configuration, select file to read: use Browse to choose the file "actionable.xls" then click OK

7. Shift + f7 to run

8. Right-click on the four "Scorer", you can read the "confusion matrix" and "accuracy statistic"

PS: this workflow contains all of the text preprocessing functions. If you do not want to use a function, for example "snowball Stemmer", the line between 'stop word filter' and 'snowball stemmer' and the line between 'snowball stemmer' and 'term filtering' are eliminated, and a new line connecting 'stop word filter' and 'term filtering' is drawn.

2. Document_Classification • // In this workflow the modules Term-filtering and transformation are offered by KNIME community

Running this workflow need to install KNIME analysts platform firstly. https://www.knime.org/downloads/overview?quicktabs_knimed=2#quicktabs-knimed
I suggest to install the "KNIME Analytics Platform + all free extensions” version

This workflow is used to implement the trained SVM classifier to classify a raw data

How to use it: 1. installing KNIME analysts platform

2. Click file- Import KNIME workflow, click Browse to choose this workflow and click finish.

3. Double-click XLS Reader "import the descriptive data"

4. In the top of the configuration, select file to read: use Browse to choose the file "descriptive.xls" then click OK

5. Double-click XLS Reader "import the actionable data"

6. In the top of the configuration, select file to read: use Browse to choose the file "actionable.xls" then click OK

7. Double-click XLS Reader "import the raw data"
8. In the top of the configuration, select file to read: use Browse to choose the file which contains the raw data, then in the "select the sheet to read" select the sheet which contain the peer feedbacks. The default name of that sheet is "PeerFeedback"

9. Double - click XLS Writer. In "Select File" you can choose the output path. Then click remove all in the bottom of the configuration

10. Shift + f7 to run //the node SVM leaner will give a WARN, this won't influence the results.

3. Actionable.xlsx and Descriptive.xlsx

These two files are manually labelled data, used to train the classifier. You can add new data into those files if you want

4. Week04_export.xlsx and week06_export.xlsx

These two files are raw data.

5. Results of week 6 data.xlsx and results of week 4 data.xlsx are two samples of running Document_Classification

- Matlab

This Matlab code is used to get the students ID and contribution two columns back if you want.

Prepare the raw data (please put the peer feedback sheet into first), and the KNIME output data

Put this Matlab code and these two files into the workshop of Matlab to ensure Matlab can call this function

Open Matlab, input output ('file01.xlsx','file02.xlsx','file03.xlsx') in the command line

File01 is the raw data which first sheet is the peer assessment data
File02 is the KNIME output data
File03 is the final results which contain the predicted categories

Examples: output ('week04_export.xlsx','results of week 4 data.xlsx','week 4 final.xlsx')
output ('week06_export.xlsx','results of week 6 data.xlsx','week 6 final.xlsx')
These two files: week 4 final.xlsx and week 6 final.xlsx are two sample outputs of this Matlab code