Predicting Molecular Structures
using SMILES strings and Bayesian Networks

Student: Jacob Rivett
Supervisor: Simon Dobson
Date: 7th April 2017

This thesis is submitted in partial fulfilment for the degree of Bsc at the
The School Of Computer Science
University of St Andrews

Abstract

The vast amount of data being generated today by researchers has caused a great need for automatic data management techniques that allow people to search and find data efficiently. The Schools of Chemistry and Computer Science in St Andrews created a system called NOMAD to manages NMR data and allow users of the system to search for experimental data using chemical structure formulae. Drawing a chemical structure formula can be a slow, error prone process and a search for systems that attempt to help increase efficacy to the drawing process, by making predictions to what the user might be attempting to draw failed. The aim of this project was to generate predictions for molecular structure drawing. To do this a Bayesian network was created from a graph of past structures users had drawn. From evaluating the system using a user study, the application saw improvements in the time taken to draw structures if the user opted to use a prediction and the study saw the percentage of accuracy of structures correctly drawn increase when users used predictions. There are many new areas to explore in terms of molecular structure predication but from the findings in this work, the use of predictions when drawing a molecular structure could improve the speed of data acquisition in NOMAD.

0.1 Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 16393 words long, including project specification and plan but excluding appendices.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

0.2 Acknowledgement

Thanks goes to Professor Simon Dobson for supervising the project. I would like to thank Dr Tomas Lebl who acted as a consultant throughout the project. Thanks goes to Simone Conte who gave me the initial idea for the project and was very willing to guide me and give constant positive feedback to my work. The NOMAD team, who gave access to code their repository and allowed me to join their team. Thanks goes to Matthew Holden who pointed me in the correct direction for the statistical evaluation of the study. Finally thanks goes to the St Andrews School of Computer Science for Technical support and the use of their resources.

Contents

1 Introduction

1.1 Background

In all areas of science, researchers create large amounts of data on a regular basis through experiment and data gathering techniques such as human focus groups, interviews and use of web analytical tools. Managing research data can be very difficult due to its large quantity. Many researchers see the quick release of publications far more important than long preservation of findings and because of this there is a high need for easy, automatic data management in research [8].
The term big data has come to light in recent times because of the rapid pace at which data is being created in the world today. Big data can be described using four V's: Volume, Velocity, Variety and Veracity [13]. For research, volume, variety and veracity normally apply. Volume describes the vast amount of data that is created on a daily basis not only in research but everywhere in the world. Variety describes the different forms of data. The variety of data can create a major problem when creating automated systems for data management, as they have to be able to deal with the differing forms of data that a researcher might find valuable. Finally, veracity is the data that is actually useful, out of the large amount that is produced. This is a very important factor in research because researchers need to be able to locate the most accurate data that is not biased towards a certain result. There is no one simple solution to managing big data due to the individualised needs of science departments and individuals when creating and storing their own data.
The School of Chemistry at the University of St Andrews is no different with active research that generates large quantities of data daily. One type of data gathering technique uses Nuclear Magnetic Resonance(NMR) spectroscopy. NRM spectroscopy is considered one of the most useful technique in organic chemistry for determining a structure because it can display which functional groups (groups of atoms and bonds that are responsible for certain reactions in the molecules they make up) are present in the structure and their arrangement [3]. In order to address the problem of having to store this data, an online control system named NMR Online Management And Datastore (NOMAD) was created as a collaboration between the Schools of Chemistry and Computer Science.
The main objective of NOMAD initially was to organize the NMR spectrum data produced so that it could be located quickly and easily. Before its creation, data was disorganised and hard to locate. These kinds of issue could cause hours or even days to be wasted. For instances, before NOMAD was introduced data was logged on sheets of paper and scattered across many storage units. Searching or finding this data quickly was directly reliant on human organisation, allowing for a high chance of human error which ultimately lead to loss of data [5].
The NOMAD system has now been in operation in the department since 2012 where it has grown from one NRM machine to as many as six with over 600

users to date. Further development of version 1.0 stopped in 2016 with the first version of NOMAD having seen to have be a success.
Through this success NOMAD 2.0 went into development and now is in beta testing. This version was designed with the addition of a number of small new features. One important feature was the creation of more decoupling components inside the system. This was partly done by focusing on creating more generic machine portals for the NMR machines sample output. This gives more opportunity for the system to be moved to other Chemistry departments in the future.
This project focuses mainly on the added feature to the system of multiple NMR experiments of the same compound together and attaching a molecular structure formula describing the compound. This allows for users of the system to search the database by the molecular structure formula to locate experimental data. With this extra functionality in mind, the project will try to increase efficiency in locating structures with concern to the molecular structure formula being drawn on the system.

1.2 Project Aim

The NOMAD 2.0 allows users to associate molecular structure formulae to compounds, where a compound is a group of experiments for the same sample. The structure is attached to better describe what the experimental sample is. With this functionality came the ability to located experimental data by searching using molecular structures. The search consists of trying to match the likeness of the structures drawn to others stored in the database and also sub structure search exists. Users were previously only able to search data using metadata about experiments such as date of experiment, user or group, before this was introduced.
The drawing tool that was introduced for drawing structures in NOMAD can be slow and error prone. It offers the user little assistance towards the process of drawing. The library used does allow for the loading and saving of molecular structures but this function is local to a machine and does not match the NOMAD online distribution capabilities, with many users of the system using many machines rather than one.
The project's aim was to explore the possibility of a tool that could wrap around the drawing tool, and which could then give predictions about what the user might be trying to create, using their past sessions of activity on NOMAD. This meant prediction not only of the structure the user tried to draw but also the next steps he needs to use before he can complete the drawing. This was an open ended point to the project but ultimately the goal was to create a solution that could speed up the drawing process. If the drawing process could be sped up, then this would directly allow users an increase in speed and accuracy when trying to locate the experimental data they required from NOMAD.
The project would also have to learn from and cater for users using the system, improving predictions as the system is used. To fulfill this aim, it would have to have data collection techniques for structures drawn in the system. This is

vital for how successful the predictions will be in the future.

1.3 Hypothesis

This dissertation will test the following hypotheses:

$H1$. Will the introduction of predictions make any significant reduction in time for users drawing molecular structures compared to when they do not have prediction? There should be a noticeable difference for the time taken to draw a structure for a participant when they click on a prediction compared to when they do not.

$H2$. Is there a significant difference between using different metrics within the predictions? The application will be using the user and group as prediction data. The expected result is that the user and group combined should give a shorter time compared to just either the user or group.

1.4 Initial Requirements

1.4.1 Primary

- Design and implement a tool that makes suggestions about what a user is attempting to create whilst they draw a molecular Structure Formula. These predictions should occur with as little delay as possible, when a user makes a change to the drawing
- The tool created should be able to make correct predictions, when the user is one step away from completing the structure he is trying to draw, around 50% of the time, given that the structure is in the database
- A simple user friendly interface for testing and running the user study should be implemented to access the tool

1.4.2 Secondary

- The tool created should be able to make correct predictions, when the user is one step away from completing the structure he is trying to draw, nearly 100% of the time, given that the structure is in the database
- The tool should be able to make good educated suggestions when making predictions using the data set it has.
- The tool should be decoupled in concern with the interface and database. This is so the tool can be easily used with any interface or any database

1.4.3 Tertiary

- The tool should be able to learn from users' structures that it does not get correct or has not seen yet. This should directly make the tool improve in its suggestions as its data set grows

1.5 Approach

The decision from the start of this project was to approach it using computer science solutions, because I have minimal chemistry knowledge. This did not mean that the project would not benefit from any chemistry domain experts input, but the low level workings would not look to understand the structures the program was to deal with. The prediction tool will treat the structures as objects without taking into account some of the attributes or rules that could be associated with the objects. It was important however, for the project to use as much information as possible from the School of Chemistry at the university of St Andrews as possible.
Meetings took place on a number of occasions with the product owner of NOMAD, Dr Tomas Lebl, to try and gain information on what a Chemistry student using NOMAD would need from a predictive tool. Without this input, it would have been impossible to understand if the project was going in the correct direction. One example of this was at the beginning of the project where I asked Dr Tomas Lebl to explain the thought process of a chemist when drawing structures and this was key to how I would attempt to model the prediction tool.
At first however, the idea of the prediction tool was not something that came naturally to Dr Tomas Lebl, because he had never seen something like it before. His main concerns were whether the system could comprehend the data quickly enough when a structure was being drawn to display meaningful predictions. As the project was exploratory this was a factor that could have hindered it, but with much discussion, he saw the possibility of the tool working and improving NOMAD.

1.6 Thesis Outline

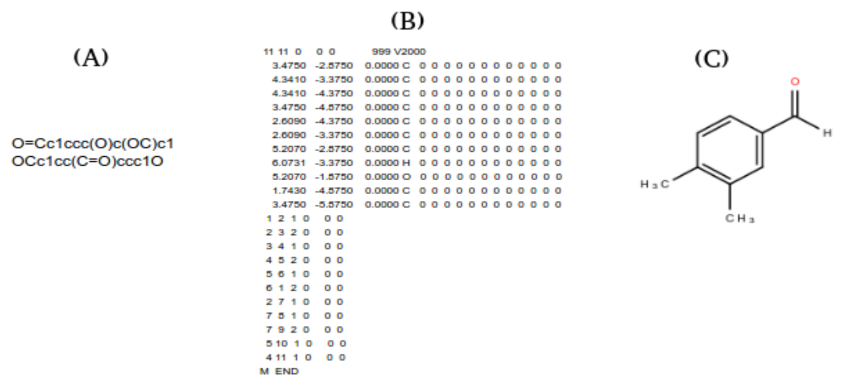
The Remainder of this dissertation is structured in the following format. Chapter 2 introduces related work and the literature required to understand the overall application. Chapter 3 considers possible ethical issues when implementing and evaluating the project. Chapter 4 talks about the software Engineering processes followed and the technologies used to create and help maintain the project. Chapter 5 introduces the system and how it was designed. Chapter 6 then talks about the exact implementation details of what is desribed in the design section. Chapter 7 describes the user study used to try and evaluate the effects of the tool for the two hypothesis explained in section 1.3. Chapter 8 talks about work that could be done to improve the system in the future. Finally, chapter 9 concludes the project.

2 Literature Review

This section will introduce the concepts that are required to understand why the overall prediction system was built the way it was. It introduces the underlining data representation of molecular structures in the form of SMILES and molfiles. Some examples of the drawing tools for molecular structures that were considered and their key features. This section also disuses machine learning and Bayesian network modeling to introduce how predictions were generated. Graphical data is discussed as later this is the underlining representation of the the structures within the system. It looks also at some similar systems that this prediction engine took inspiration from. Finally it talks about some of the statistical tests and models that needed to be used to evaluate the study.

2.1 SMILES and molfiles

Two important data formats within the system that will be discussed are Simplified molecular-input line-entry system (SMILES) strings and molfiles.
SMILES strings are an ASCII string representation of a molecular structure and represent molecular structure in a compact size. They hold no positional data on atoms and bonds geometry within a structure and because of this it is not possible to directly display a structural formula using a SMILES. The SMILES can however be converted into a molecular structural formula easily using took kits. Nonetheless, not all conversion tools will give the same output because the SMILES are missing geometric information, and took kits will differ in the coordinates of atoms and bonds they output. SMILES within this application are mostly used for comparison but it worth noting that in some cases multiple SMILES can convert to the same structure. A canonical SMILES is required to make exact comparison of structure but this project looked to focus on users drawing styles more. consequently, this application uses SMILES over canonical SMILES because SMILES retain more information on exact connectivity of atoms and bonds unlike canonical SMILES.
molfile are a type of chemical table file format that represents a molecular structure but unlike SMILES, it holds the coordinates of atoms and bonds. Connectivity between these are also held similarly to SMILES strings. Using this information, it is possible to save and display molecule structure. Molfiles are larger in size than SMILES and cannot be used accurately for comparisons because the same molecular structures can differ in their position of components in a molfile. The use of molfiles are necessary because SMILES can not directly be used to represent the structure to a user and they act as the intermediate level between the SMILES and what the user will see and edit within a drawing tool.
In this application, SMILES are used for comparison and molfiles are used to represent and display structures created. Figure 1 shows three representations of Vanillin using a SMILES string(A), a molfile(C) and structural formula(C).



```
O=Cc1ccc(O)c(OC)c1
OCc1cc(C=O)ccc1O
```

```
 11 11  0    0  0         999 V2000
   3.4750  -2.5750   0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   4.3410  -3.3750   0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   4.3410  -4.3750   0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   3.4750  -4.5750   0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   2.6090  -4.3750   0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   2.6090  -3.3750   0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   5.2070  -2.5750   0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   6.0731  -3.3750   0.0000 H   0  0  0  0  0  0  0  0  0  0  0  0
   5.2070  -1.5750   0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
   1.7430  -4.5750   0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   3.4750  -5.5750   0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0     0  0
  2  3  2  0     0  0
  3  4  1  0     0  0
  4  5  2  0     0  0
  5  6  1  0     0  0
  6  1  2  0     0  0
  2  7  1  0     0  0
  7  8  1  0     0  0
  7  9  2  0     0  0
  5 10  1  0     0  0
  4 11  1  0     0  0
M  END
```

Figure 1: SMILES(A), Moflie(B) and structural formula(C) for Vanillin

2.2 Molecular Structure Drawing Tools

Molecular Structure Drawing tools are used to create and modify chemical structures using structural formulae. This can be both in 2-D and 3-D form but this project will focus on 2-D representations. There are a range of different tools available but most tools researched for the purposes of this project use the concept of sprout drawing to place atoms and bonds onto the screen. Sprout drawing means that when a user adds an atom onto their structure, it will *sprout* out from where the user clicked; this is in comparison with being placed directly at the location that was clicked [10]. Figure 2 shows an example of sprout drawing within a tool called MarvinJs [1].

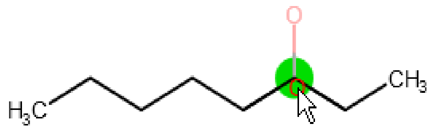


Figure 2: Example of sprout drawing

The following subsections will discuss a number of the drawing tools that were considered for this project and later in section 4.3 we will discuss the decision

2.2.1 ChemDraw

ChemDraw[2] is the most popular software tool used by chemists for drawing molecular structures. It is part of the ChemOffice suite of programs and is

[1]MarvinJs `https://marvinjs-demo.chemaxon.com/latest/docs/manual/Drawing-Chemical-Structures_17236086.html` [Online;last checked:4/3/2017

[2]ChemOffice `www.cambridgesoft.com/Ensemble_for_Chemistry/ChemOffice/` [Online;last checked:4/3/2017

available for Microsoft Windows and Macintosh computers. The ChemDraw application is inside the ChemOffice package is on its 16th version. It supports a wide range of file formats for reading and writing such as SKC, molfiles, v3000 and SDF. Some of the useful user features include hotkeys for quick operations, structural warnings, structure cleanup. There are many more features which are beyond the scope of this project.
It is relevant to talk about this drawing package, even though it is not a web-based tool because it is what most Chemists are familiar with. Most of their expectations of how a drawing tool should operate has come from this software. Any system chosen for this project should support the core features of ChemDraw.

2.2.2 Ketcher

EPAM Life Sciences [3] develop organic chemistry, open source free software under GNU affero general public license. One solution they developed is a web-based chemical structure drawing tool called Ketcher. The software is open source but the developers ask users of the library not to changed or alter the code base of the library. It is written in pure Javascript, giving it high performance and good portability.
Ketcher supports Mol and Rxnfile file formats. From a user perspective, it offers a range of different tools such as hotkeys for quick operations, chemical warnings, sprout drawing, custom atom labelling with the keyboard, and selection tools that can deal with parts of the structures drawn, rather than the entire structure. It has an overall smooth operation of atoms and bonds onto the display but does not support the preview of placement before the actual placement, which can cause unwanted placement of components.

2.2.3 MarvinJs

MarvinJs[4] is a web-based version of MarvinSketch, a desktop application, written in java-script and it requires a license key to be used. This tool has taken many of the features from the desktop version but not all. It supports quick key actions, a clean up function, sprout drawing, support for drawing chemical reactions and selection and rotation of structures. It also suggested to have the ability to easily integrate new custom buttons to the toolbar.
There are some interesting features from the desktop version that the developers did not move across to the web application, two of which are the support for 3-D drawing and the ability to search their online database of structures. This search supports abilities such as substructure search and similarity.

[3] Ketcher `http://lifescience.opensource.epam.com/ketcher/` [Online;last checked:1/3/2017

[4] MarvinJs `https://marvinjs-demo.chemaxon.com/latest/` [Online;last checked:2/3/2017

2.3 Machine Learning

Machine learning is an area of computing where you use data to make a generalizable model that give accurate predictions, or to find patterns within data [4]. This pattern most of the time needs to be adaptable to new data. The data that these models use normally consist of the observation and a number of features describing that observation. Several models exits under this area in computing such as Bayesian Networks, neural networks, support vector machines, general linear, polynomial models and many more. The majority of these models use small calculations to generate their output and because of this simplicity the are very scalable. All these different models use the data given to them in a different ways to model a prediction. In this project it was important to find a model that could be used for prior information and events that were connceted to one another.

2.3.1 Bayesian Network

A Bayesian network is a probabilistic predictive model which attempts to organise a joint probability distribution of random variables in a structured way. It is made up of a directed acyclic graph where nodes in the graph are random variables. Directed edges are used to connect nodes together and each edge represents a dependence between the nodes. Nodes that are unconnected to one another in the graph are said to be conditionally independent. Informally, in a network of one parent node and two child nodes we can state that knowing the outcome of the child nodes does not give us any more information to the parents outcome and you can state that the child nodes are conditionally independent of one another given the parent [12].
Importantly, probability tables associated to each node in the graph also exist, along with the Bayesian graph. These tables are used to estimate the probabilities of events happening, given the knowledge of other events. These probability tables are calculated using Bayes rules. The basic rule of Conditional Probability is represented by the following equation:

$$P(A|B) = \frac{P(A \cup B)}{P(B)} \; where \; P(B) \neq 0 \tag{1}$$

This means that the probability of event A happening, given that event B happened, is the probability of event A happening and event B happening, divided by the probability of event B. This can then be mapped to the following equation:

$$\frac{P(A \cup B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)} \; where \; P(B) \neq 0 \tag{2}$$

Using the rule that the probability of event A and B happening is the probability of event B happening, given that event A happened, multiplied by the probability of event A then (1) becomes (2). Using both equation (1) and (2) we can state Bayes' Rule:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \; where \; P(B) \neq 0 \tag{3}$$

2.3.2 Naive Bayes

Naive Bayesian networks are a type of Bayesian network that are regularly used in machine learning. They make the key assumption that all nodes are conditionally independent of one another. In a real life situation, having two events that are completely independent of one another is rare, but a Naive Bayesian model has shown many times using real life data that it can outperform other prediction methods [9]. Its simplicity helps to handle large data sets well, because calculations are small and so using more complex models calculations would affect performance.
To calculate the probability of two independent events occurring, we multiply the probability of the two events together (4). Given C is a dependant event on X and X is n number of events independent of one another, we can work out the probability of an event happening using the independence rule and the assumptions made about Naive Bayes using equation (5):

$$P(A|B) = P(A)P(B) \tag{4}$$

$$P(C|X1, ...Xn) = \alpha \prod P(Xi|C)P(C) \tag{5}$$

The α within the formula is used for smoothing and makes sure that the product of all the probabilities is normalized to one.
One problem that can occur with Naive Bayes is when dealing with data sets where a certain category has an unobserved event. This will cause the overall probability of a certain event occurring to be zero. This means that in the formula (5) if any of the Xi values are zero in the data-set then the overall probability $P(C|X1, ...Xn)$ will be as zero. In most practical cases, this is undesirable and in fact almost always it is wanted for the model to show the chance of an event occurring, even if the current data set might never have seen it before. One solution to this is the Laplacian correction, which is sometimes called additive smoothing. In Bayesian terms, the Laplacian correction refers to using the prior distribution of random events occurring to correct the probability of events. If our prior knowledge tells us that there is a chance of an event happening then this means that there is no chance of the probability ever being zero. So simply, to do the Laplacian correction, the operation is to increase the occurrence of all events by one.

2.4 Graphs

Graphs in real life are used to describe relationships between objects in a visual way. Graphs consist of nodes connected together by lines or arrows normally. An example of a graph in real life is a family tree, where nodes are used to represent people and arrows indicate the relationship of who is the parent of

whom in the graph.
In mathematics, a graph is made up of three components - a non empty set of vertices V, a set of edges E and a incidence function F. The incidence function F takes an edge e, where $e \in E$, and two vertices v1 and v2, where $v1, v2 \in V$, such that v1 and v2 are the the ends of e. The graph can be unidirectional or directional [2].

2.4.1 Graphical Databases

Graphical databases have become increasingly popular in recent times. They use graph theory by joining data together using direct connections between data, unlike in traditional relational databases where data is stored in multiple tables that then can be joined using a join query, where two tables are matched together by their column values. Graphical databases consists of nodes that make up entities, edges and properties. A node entity is a type of object that contains the properties of an object, the properties being the data itself that is associated with the node, for example a person node has the property name. Finally they have edges between the nodes that represent relationships between things in the database. For example, a human can be related to a car by owning it.

2.5 Similar Systems

Trying to locate a system that already existed and that did exactly what our prediction tool was attempting to do was not possible. A number of chemistry students and lecturers were asked if they knew of a system that attempted to predict structural drawing but they indicated that they were not aware of such an existence system.
There are many online open source chemical structure databases but none that attempted use their data to aid users in drawing structures. By carrying out searches online and within the School Of Chemistry in St Andrews it seemed to be a rare occurrence where user metadata was stored along with the chemical structure data they produced. This project strongly takes advantage of this factor.
For these reasons, more general comparisons were located.

2.5.1 Text Prediction

Text prediction works by using prior information about users past input of sentences, words or letters and using rules about the language to make predictions. For example, within the English language, in almost all cases, the letter ‘q’ is followed by the letter ‘u’ in a word. With this information one can say it is highly probable that a user will enter the letter ‘u’ next when they have initially entered a ‘q’.
A paper written by Proksch et al. [18] introduces a form of code compilation,

called Pattern-Based Bayesian Network (PBN), that makes suggestions for application interfaces (API) for IDE editors. It creates prior statistical data on an important library by initially looking at existing code repositories that use that library. PBN then can generate a Bayesian network using this prior knowledge and context information, such as method calls that have already been observed to predict what method call a user might type next. Code completion also uses the rules of the programming language to increase the probability of what the user will type next as discussed similarly to the 'q' and 'u' rule example earlier. Meaning if a user types a variable of type String and begins to assign its value by calling a method, it is far more likely that method call will return a String than not because it might cause a type error.
Text prediction in general is very similar to what this project uses as its underling prediction model, because text prediction uses a data-set of prior knowledge it has built up to then make predictions, and uses new data to improve its data set. The paper by Proksch et al also uses the same model of a Bayesian network to create predictions within an environment where the system can learn from users' personal actions, using the context of what the user has already done to improve the predictions. The biggest difference in our project, to the PBN, is that the strings are translated into molecular structure formula that is then shown to the user in an editable form. They edit a structure rather than directly the text but importantly, the underlining decision making process is very similar.

2.5.2 NOMAD 2.0 Search Functionality

Structural search techniques can use a number of different ways to compare structures to one another; structures can be matched through identical, substructure or similarity search techniques, though the term 'identical search' can vary in its definition. A common approach for likeness search measures two structures using an equation called Tanimoto and binary fingerprint. A binary fingerprint consists of an ordered list of binary keys stating the presence or absence of a particular substructure [20]. A substructure is a fragment of a chemical structure [19]. Using theses fingerprints you can search for structures with a threshold, meaning you can search for structure with a percentage of likeness to a structure you have drawn. 100% threshold would equate to an identical search and 0% would return all chemical structures.
Within NOMAD 2.0 the functionality to add similarity search was introduced and a proposed idea at the start of the project was to use this structural search for predictions, by using the likeness search on the database to match the structure a user is attempting to draw. Reasons for it not being used in this project are discussed in the Design chapter, section 5.1.

2.6 Statistical tests and Models

2.6.1 T-test

A student t-test, sometimes referred to simply as a t-test, measures two data sets means against one another to see if there is enough of a statistical difference to reject the null hypothesis - the hypothesis being there is no difference between the two sets of data. A student t-test must have normal populations with unknown, but equal variances between the groups. A t-test outputs a t-value that indicates the significances between the two groups means and the further away the t-value is from zero the more likely it is to be significant [21]. The p-value is then calculated using the t-value and the p-value states the chance that the null hypothesis is true. A Welch's t-test is similar to a student t-test but for undistributed values and can work for unequal sample sizes of data.
A t-test is very well known test for researchers who are looking to see significant difference between groups when running experiments and this is where this applies within this project.

2.6.2 Generalized Linear Model (GLM)

A Generalized Linear Model (GLM) is a type of linear prediction model that extends a general linear regression model to deal with values that are not evenly distributed. The phrase not evenly distributed, normally means the range of y values in the model are restricted [23]. The GLM is made up of a linear predictor (6), A link function that describes how the mean depends on the linear predictor and a variance function that describes how the variance depends on the mean[23].

$$E(y) = B0 + B1x1 + B2x2 + Bnxn \tag{6}$$

Where E is the error distribution function, y is the expected value returned that has a normal distribution and x is the explanatory variables [15].
There are many different versions of a generalized linear model that use different linear prediction functions such as binomial, gaussian, gamma and a many others. Each of these models are good for different purposes. For example, a poison GLM is good for when you are trying to predict a number of times something has occured [11] and a fitted gamma GLM works well when dealing with positively skewed mean distribution of data. A fitted gamma GLM alters the linear predictor, equation 6 to an inverse link function instead (7).

$$\frac{1}{E(y)} = B0 + B1x1 + B2x2 + Bnxn \tag{7}$$

By generating a GLM, it is possible to see the statistical significance of each variable in the model towards predicting the model itself. It does this by using a t-test between the model with and without the each variable and you can then view the t-value and p-value of each variable.

3 Ethical Considerations

The development phase of the project, did not require any ethical considerations as all data generated was random and no participants were involved.
The evaluation phase required ethical approval in order to gather molecular structure data from users and then to record information on those using the system. The little personal data gathered was anonymised immediately after the user had completed the study. This data was stored for a limited time in a safe place on the relevant personal computer behind password protection. All data about the molecular structures was held securely in the database of the School of Computer Science St Andrews. All participants were given an information sheet outlining the risks, and a participation form to sign before beginning of the study.
The ethical approval letter can be found in the appendix section A.

3.1 Collaborations

This project involved collaborations with a number of people throughout the project:

- Simon Dobson was the supervisor of the project and gave advice on the direction of the project in out weekly meetings
- Simone Conte, a developer heavily involved in the development of NOMAD, initially proposed the idea of the project. He also supervised the project when Professor Simon Dobson was out of office. He helped design the experiment and also looked at drafts of the report, giving general comments.
- Dr. Tomas Lebl acted as a consultant initially, regarding whether the project was plausible and gave useful input from a chemistry domain background. This is discussed in the Approach section 1.5.
- Finally, near the start of the project, the NOMAD team allowed access to the code repository for NOMAD. No code from the repository but it acted as a starting point to generate ideas.

4 Software Engineering Process

4.1 Methodology

This section discusses the software engineering practices which were followed through the life-cycle of the project. It was important to find the correct approach on building parts of the software because the duration of the project was a year, meaning that code written would be required to be understandable long after the project had ended.
When doing exploration it can also be difficult to set strict iterative steps to follow in order to reach the end goal of a project because the situation changes rapidly. Parts of the code had to be completely rewritten or discarded. Hence agile practices to develop the project were essential. Agile means one is adaptive to changes to the software; causing parts of the design stage to become heavily integrated into the implementation stage. Feedback given by the product owner became very important because of constant changes.

4.1.1 Version Control

Because the project was seen as exploration, the system was built out with the NOMAD repository. This also avoided potential clashes of activity while the development of NOMAD 2.0 was ongoing.
Mercurial was used for version control software because it is supported by the School of Computer Science In St Andrews and was most familiar to the researcher.
At the start, truck based development was used in order to quickly achieve a working system. Later on in the project however, feature branches were used in order to maintain a working solution and merging was done once a new feature was completed. This also allowed the trial of features that later did not work. Commit messages were important as parts of the program were left for a long time and needed to be understood later on in the project.

4.1.2 Build Tools

For the front-end development, no build tools were used for the libraries but they were added to the repository directly. With no testing on the front-end, it made sense to have the minimal use of libraries
For the back-end, Gradle [5] was used as the build tool . The project needed to be packaged up and ran in multiple places and having a built process was vital towards this. It also ensured that the test written for the project could be run atomically when building the project.
Finally for the database, all scripts for creating the database were stored in the rest-API folder. This meant that at all times, a database scheme was available, and could be used to recreate the database. The database was also versioned

[5]Gradle `https://gradle.org/` [Online;last checked:5/3/2017

as it was built up, getting dump files from the database at every stage of the sudy. This helped with avoiding the possibility of the data being lost.

4.2 Tools and Technologies

4.2.1 Java

Java is an object oriented, class based programming language and it is one of the most popular in the world today, used mostly for running web applications [16]. Java requires to be translated into byte code before it can be run on the java virtual machine (JVM). This makes it portable as any machine that has a JVM can run java code, leading to the use of the phrase, 'write once, run anywhere'. It is the language chosen for the main server of the prediction application in this project, using a library called Spring Boot. This decision was made mainly because of my familiarity with it, and the fact it is a tried and tested way of creating web servers.

4.2.2 Spring Boot

Spring Boot[6] is a Java Library that is considered to be an 'out of the box' application when building restful APIs. Restful APIs are a form of web service that uses a restful architecture (which refers to the way client and server exchange and make use of information in a state less way). Spring Boot uses HTTP or HTTPS requests with the data exchanged being in the form of JSON.
Using Spring boot meant the project could very quickly get an application running with minimal setup and without having to deal with lower level concepts such as sockets. Configurations were simple using an application.properties file.

4.2.3 SMILE

SMILE[7] is a platform independent library of C++ classes for probabilistic models, such as Bayesian networks and influence diagrams [1]. It allows one to build graphical models that can be edited and saved. It is purely used for structural assistance and does not make any of the probabilistic calculations. It also has a Java wrapper available called JSmile which was used in this project's application for building up the Bayesian network structure.

4.2.4 RDkit

RDkit[8] is a library written in C++ and python for cheminformatics and machine learning. This project only uses one function of it to generate molfies from SMILES strings. For java to access the library it using a Java wrapper.

[6] Spring Boot `https://projects.spring.io/spring-boot/` [Online;last checked:2/3/2017
[7] SMILE `https://download.bayesfusion.com/files.html?category=Academia` [Online;last checked:2/3/2017
[8] RDKit `http://www.rdkit.org/` [Online;last checked:2/3/2017

4.2.5 Java-Script

Java-Script is a scripting language used in all modern web browsers to manipulate HTML. In this project, a library called JQuery was used to handle most events on the test web-page front-end and JQuery's simple functions were used to handle all the communication with the server application.

4.2.6 Bootstrap

Bootstrap[9] is an open source front-end web-framework, which mainly contains CSS and HTML but also has JavaScript extensions. In this application it was used for layout of the front-end testing component. Explicitly, it was used for styling of the form for the metadata, styling of buttons, display panels and Modals, which are a type of pop-up window.

4.2.7 Mariadb

Mariadb[10] is an open source, relational database management system that uses SQL query language. It was initially forked from MySQL's source code and so almost all of its operations are similar to that of MySQL. A database is any structural data and MySQL is just a management system for structured relational data, which stores data in many tables rather than all in one place [14].

4.2.8 R

R [11] is an open source programing language for statistical calculations. This package is used in the evaluation phase of the study.

4.3 Structure Drawing Library

The decision was to use the Ketcher library within this application and the key reason for this was its availability. It is an open source library, unlike MarvinJs which requires a license. The inability to change the Ketchers source code however was a stumbling block at times in the project. MarvinJs, instead, might have had a better API and abilities for adding features to the drawing tool.
The drawing style of both the Ketcher and MarvinJs are similar to the one used by ChemDraw and use sprouting. A major benefit to MarvinJs is its preview of component placements. This gives users of the tool less chance of errors such as placing the wrong components on the structure or placing components in incorrect positions. However, in the opinion of the researcher, the Ketcher tool feels smoother and less clunky in movement of structures compared to MarvinJS.
When considering performance, the Ketcher tool is written in pure java script, giving it a strong case to be used. This is in contrast to MarvinJs, where features

[9]Bootstrap `http://getbootstrap.com/` [Online;last checked:2/3/2017

[10] Mariadb `https://mariadb.com/` [Online;last checked:2/3/2017

[11] R `https://cran.r-project.org/` [Online;last checked:2/3/2017

have been copied from the desktop version rather than being fundamentally focused on the development of the application for the web.
If the main concern is with users familiarity, then ChemDraw would have been the best application to use but it is not a web-based editor. A plug in has been created to access some web capabilities but it is outdated and still requires a desktop runnable version of the software. Both Ketcher and MarvinJs share important functionality with ChemDraw, such as hot keys and chemical warnings on rule breaking structures, which are both important requirements for this application. Thus, either Ketcher or MarvinJs would have done the job required but because of the licensing issue mentioned, Ketcher was the best option.

5 Design

This section describes the process and reasoning taken when designing the system. It describes at a high-level, how data is structured in the database, how structures are created to populate the database, how predictions are generated from the database and finally gives a simple example of how each part works together.

5.1 Representing Molecular Structure Data

The decision on how the application would represent the molecular structure formula data was fundamental because predictions would be directly generated from the representational data. A simple approach to storing this data would have been to simply save the end structures produced by users, with their SMILES and Molfies. However, doing this would mean losing data on the intermediate steps taken by users when drawing structures. The data associated with intermediate steps seemed too valuable if the tool was to explore predicting the steps involved in drawing a structure rather than the just the end goal. Only storing the end structure would mean that the predictor would be unable to suggest intermediate steps. So, it was required to came up with a more complex storage technique.
One solution was to create a graph of all the steps taken by users when drawing structures in the application. Nodes in the graph are SMILES of either intermediate steps or end structures themselves. An edge between nodes would represent a user drawing a component in the drawing tool and moving from an intermediate step to the next structure. This structure could be the end goal or it could be the next intermediate step. A high-level representation of this can be found in figure 3. Properties of the data are then attached to the edges. Each edge then holds the number of times each user in a certain research group has taken that step in the past when drawing structures.
A key area that could be used in the graph when making predictions was the repetition of the steps user took at a certain point of the drawing. This could be done by finding the node in the graph that represents what point the user is currently at and then using the data of users' choices in the past to make predictions.

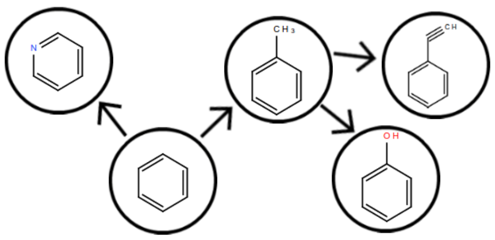
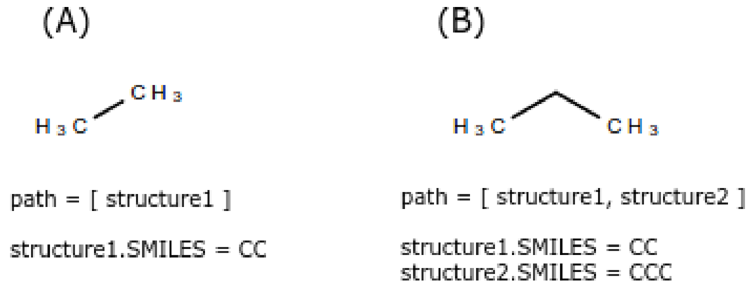


Figure 3: High-level representation of database

5.2 Populating the Database

The next step was to work out a way to populate the database given that a way to represent the data had be designed. Without being able to generate data for the database, making predictions would not be possible. One of the initial requirements was to create a testing front-end component. The main tasks of the front-end would be to display the predictions and create structures to populate the database. In order to populate the database, the front-end has to be able to save the steps in the drawing process. This would be represented by a path of structure steps that would then be added to the database graph.
The path, therefore would be represented by list of structure objects containing a SMILES string and a molfile. The adjacent elements in the list represent the next step in the drawing process. For example, if a user starts by drawing a bond, the first element in the list is a structure with the SMILES and molfile of a single bond, figure 4(A). The user then draws a hydrogen atom onto the end of that bond and the path then becomes a list of two elements. The first element still containing the SMILES and molfile of a single bond and then the second element contains the SMILES and molfile of a combination of the Hydrogen and the Bond figure 4(B).



path = [structure1]

structure1.SMILES = CC

path = [structure1, structure2]

structure1.SMILES = CC
structure2.SMILES = CCC

Figure 4: Process of building up the drawing path.

From an early stage, it was obvious that the accuracy of these paths would be vital because otherwise the database graph would contain useless repeated circles. To avoid this, there needed to be some form of validation on the paths which would look for loops, removing unnecessary data. Some of these types of loops would be produced by users undoing their actions and then redoing them. A solution to this problem was to build up a history of users actions on the front end. Luckily enough something similar to this already existed, with the path list of structures which is sent to the application to save the user structure. An alteration was made to fix the undo and redo problem by storing an index value into the path list. This index now represents the end of the path. If the user clicks the undo button, instead of removing the end element in the path list, the index is moved back one in the list. If the user now clicks redo, the algorithm

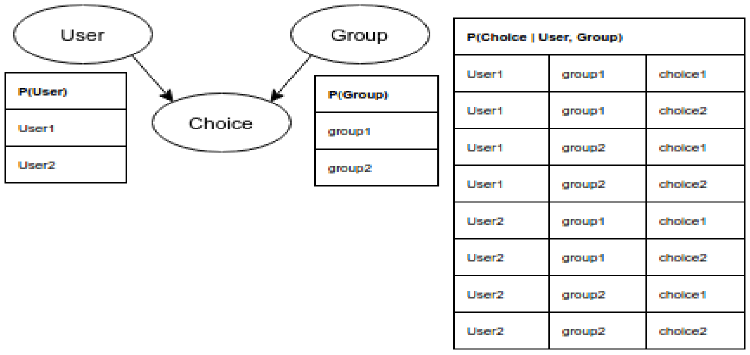
moves the index forward one instead of creating a new element into the path. Once this is done, when the system sends the structure to the database it must just remove all the elements after the index.
Another solution is when the path is sent to the database, the application loops over it and for each element i, it compares i with i + 2. If element i and element i + 2 are equal then element i and element i + 1 are removed from the list. If they are not equal, the algorithm moves to the next element in the list. This is done iteratively until i + 3 is larger than the size of the path.
In fact, both of these solutions were later used in the application.

5.3 Predictions

Once the method for populating the database was designed, the next step was to design a way of generating a list of predictions with the data.
One approach would have been to use the 'likeness search' developed in NOMAD 2.0 to try to match a user's drawing with data in the database and then populate the results to the user as predictions. This was explained in section 2.5.2. However, the decision was not to explore and look at more into the drawing style of users, and the process of drawing structures in the system one step at a time. Likeness would have been a better approach if only the end structures were to be stored in the database. This is because with step structures in the database it would always suggest only one step away as being the most alike structure and might also suggest steps backwards. Also, trying to take into account the most frequently used actions taken by users, would become increasingly more difficult.
To begin with, the prediction method that the application used was incredibly simple. It would take the current SMILES string the user had drawn in the drawing tool and query the database for the node representing that structure plus all of the outgoing edges of that node. The returned structures at the end of the edge became the predictions of what the user might do next. Once this was done, the prediction model was added to, taking into account the statistical data associated with each edge by developing a model.
The first model that came to mind using statistical data was a naive Bayesian network. Another possible model was a neural network but the decision was not to use it because of the inner structure of a neural network being described as a black box. It is described as this because there is no direct understanding of how a neural network gets to its predictions and this makes it harder to understand when developing. A neural network would not be able to take advantage of the statistical data as much as a Bayesian network, because Bayesian networks work with strict mathematical rules. These rules are simple in a naive Bayesian network because of its assumptions and makes it easier to understand the inner workings of how Bayesian Network get to their predictions.

5.4 Building The Bayesian Network

The precise Bayesian network created by this application consists of a main node called ‘choice’ which represents the next structures available from the graph. It then has two parent nodes called ‘user’ and ‘group’. The user group represents the times each user of the system has picked a *choice* and the group represents the number of times all the individuals within a certain research group have picked an edge. An example of this is in figure 5, where there are two users, two groups and at a given point, two *choices*. This makes a total of eight possible combinations of values and probabilities that would require to be calculated in the choice table.



P(User)
User1
User2

P(Group)
group1
group2

P(Choice | User, Group)		
User1	group1	choice1
User1	group1	choice2
User1	group2	choice1
User1	group2	choice2
User2	group1	choice1
User2	group1	choice2
User2	group2	choice1
User2	group2	choice2

Figure 5: High-level Bayesian Network.

To use a Bayesian network in this application however, the application was required to transform the data from the database into a form that could be inputted into the Bayesian network. Even though the relational database is organised in a graph form, the graph is very different to that of a Bayesian network. The application would not be able to query the database exactly once to give back the data in the perfect form that could then be mapped directly into the Bayesian network.
One of the reasons for this is that the database does not hold information on each edge for every user and every research group in the system. For example, if a user is new to the system, he/she has no data held about them drawing any structures before. So when they come to draw their first structure, the prediction tool will still have to make predictions based on that activity but when the database is queried it will return no information for that user. This means that the missing data for that user has to be created to populate the probability table in the Bayesian network rather than taking it directly from the database information. This also applies to the research groups.
The obvious decision here would be to give that user a zero value for the number of times he has visited each *choice*, since that is what has happened. The user has never taken that *choice* before but as explained in section 2.3.2, this

would cause the user to get back a probability of zero for each of these choices. However, just because he has never made that *choice* in the past does not mean that there is no probability he might make it now. The decided was to use the Laplacian correction to fix this problem. This worked by adding a value of one to all edges that the database returned - the number of times every user made a *choice* and every time a group made a *choice*. This meant that the default value would be one time for every possible *choice*, instead of zero. By doing this at this point, a user who had never taken a *choice* before would get the probability of the general population for that choice rather than zero.

5.5 Examples

This section, given all the information already explained in the design section, will give a simple example of the overall application working. The data used within this example is fabricated and it refers to fake users for the purpose of understanding how it would work in a real life situation.

5.5.1 Example

Figure 6 shows two structures - structure (A) was drawn by user 1 within group 1 and structure (B) was drawn by user 2 within group 1.

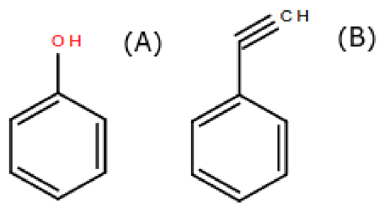


Figure 6: Two data structures added to the database of structures.

Given that these two structures are drawn and saved into the database, a high level representation of what the database would look like is given in figure 7. The Assumption is that the database was empty before these two structures were drawn. The data associated with each edge can then be found in figure 8.

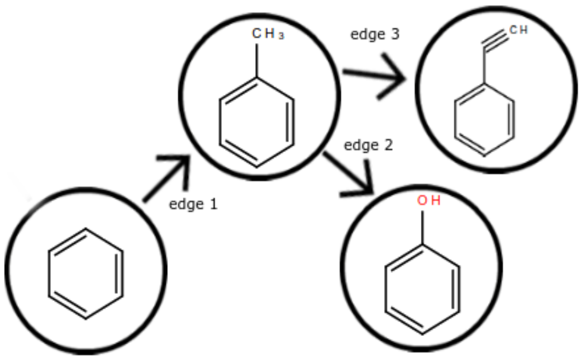
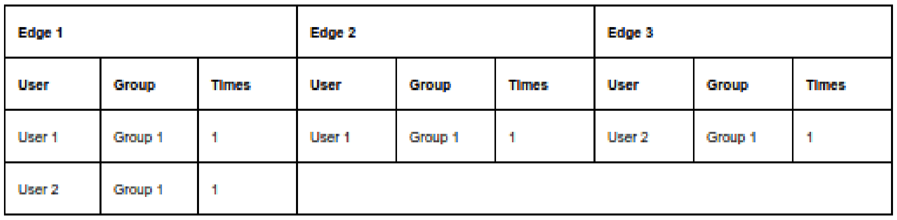


Figure 7: High level representation of database



Edge 1			Edge 2			Edge 3		
User	Group	Times	User	Group	Times	User	Group	Times
User 1	Group 1	1	User 1	Group 1	1	User 2	Group 1	1
User 2	Group 1	1						

Figure 8: The data associated with each edge in the database

With this data now in the database, user 1 still in research group 1 begins to draw a new structure in the tool. He draws until the step shown in figure 9.

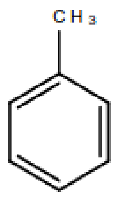
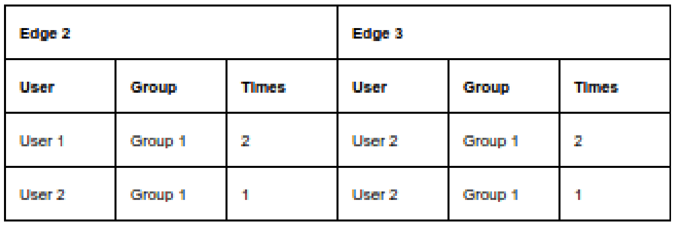


Figure 9: User 1 draws this structure in his second session.

The application takes the SMILES generated from this structure and queries the application for information on this structure and all that structure's outgoing edges. Edges 2 and Edge 3 are returned shown in figure 8. Using this data the application generates a Bayesian Network to give back structures the user might be attempting to draw. First it has to generate the missing rows needed to create the network. From the returned edges, User 1 has never drawn edge 3 before and user 2 has never drawn edge 2 before. Without using the Laplacians correction both would see a probability value of zero for either structures. The Laplacians correction simply alters the data returned to look like something shown in Figure 10.



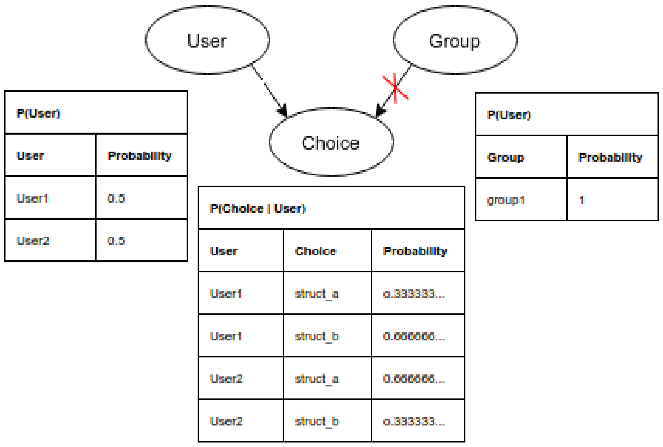
Edge 2			Edge 3		
User	Group	Times	User	Group	Times
User 1	Group 1	2	User 2	Group 1	2
User 2	Group 1	1	User 2	Group 1	1

Figure 10: The data edges returned and then altered using Laplacians Correction.

The application takes the updated data from figure 10 and creates a Bayesian network but because only one group is involved in this calculation, meaning the probability of group 1 is one, the group node is emitted from the network. The network created is described in listing 1 and a high-level diagram of the network can be seen in figure 11. Listing 1 is just outputted by the Bayesian network library used but it gives a clear output of what the Bayesian network consists of. The XML file is actually not used anywhere within the application other than for testing purposes.

Listing 1: XML Describing the Bayesian network created in example 1

```
<nodes>
    <node id="user">
        <state id="user_1"/>
        <state id="user_2"/>
        <probabilities>
            0.5 0.5
        </probabilities>
    </node>
    <node id="structureChoice">
        <state id="struct_A"/>
        <state id="struct_B"/>
        <parents>user</parents>
        <probabilities>
            0.3333333333333333 0.6666666666666666
            0.6666666666666666 0.3333333333333333
        </probabilities>
    </node>
</nodes>
```



P(User)	
User	Probability
User1	0.5
User2	0.5

P(Choice | User)		
User	Choice	Probability
User1	struct_a	0.333333...
User1	struct_b	0.666666...
User2	struct_a	0.666666...
User2	struct_b	0.333333...

P(User)	
Group	Probability
group1	1

Figure 11: Graphical View Of the Bayesian Network for example 2

So our concern is with the calculations for the probabilities of structure A and structure B given it is user one that is currently drawing. The formula required is shown in equation 8.

$$P(struct\_A|user\_1) = \frac{P(struct\_A|user\_1)P(user\_1)}{P(strcut\_A)} \tag{8}$$

The application uses the data from figure 10 and creates the required probabilities to insert into the equation. The calculation is shown in equation 9.

$$P(struct\_A|user\_1) = \frac{\frac{2}{3}\frac{1}{2}}{\frac{3}{6}} = \frac{2}{3} \tag{9}$$

This calculation is then also done for the second structure giving the result $P(struct\_A|user\_1) = \frac{1}{3}$. Figure 12 shows the structures and probabilities that would be then shown to the end user.

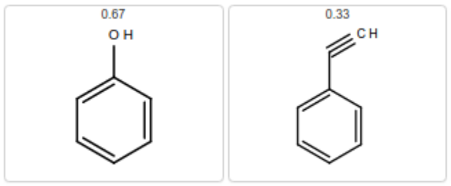


Figure 12: The structures returned for user 1 from the example explained

6 Implementation

This chapter is an overview of how the system described in section 5 was then implemented. The overall architecture of the system is client-server. This gave the ability for any front-end component to call the server. Also it allowed us to cleanly defined a simple API for the interaction between the server and front-end.
In the following sub-sections, the exact workings of back-end will be explicitly explained, how the front-end displays the predictions, and the exact database implementation. Figure 13 shows an overview of the libraries and what languages are used.

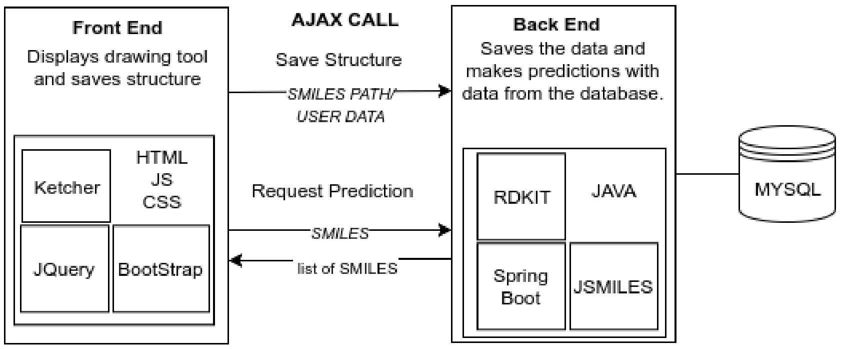
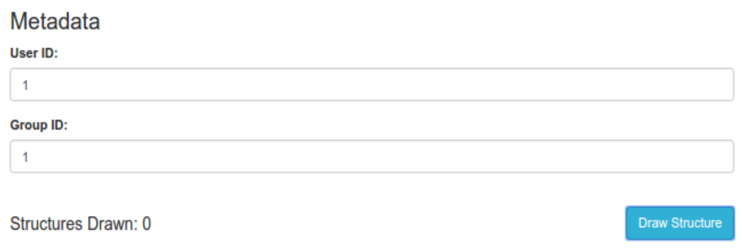


Figure 13: Overview of Architecture

6.1 Front-end

6.1.1 Metadata

The first thing seen on the test page is a web form with a user identification and group identification number, see figure 14. This form is used to mimic which data would be associated with which users in the system. Without this data, the predictions would be based only on molecular structures created in the system without information on who created it. This was discussed in the design of the Bayesian network and this form is simply replicating what data would be associated with the user in the NOMAD system.


Metadata

User ID:

1

Group ID:

1

Structures Drawn: 0

Figure 14: Metadata Form

6.1.2 Ketcher

Ketcher, mentioned in the section 2.2.2 and section 4.3, was the chosen tool for drawing the molecular structure formula. The structure formula created in the Ketcher is then translated into the SMILES and molfiles that represent the formula, and sent to the server. The back-end deals only with structures in these forms so it is important for any front-end which requires to communicate with the server to be able to generate these.
The Ketcher is placed in a bootstrap model, copying where the Ketcher was placed in NOMAD and can be found by pressing the 'draw structure' button on the main web page. It has buttons at the bottom right of the modal called 'clean structure', 'close' and 'save', with each name describing the functionality.

6.1.3 Interaction with the Server

AJAX calls were used to communicate with the back-end server. This was simply using in-build JQuery methods for Ajax calls. Ajax calls allow for web pages to make requests to web resources asynchronously, which meant that the application could update the predictions as the user added to the structure he was drawing, without the reloading of the page. There were only two main Ajax calls to the server.
The first important call sends the path of the structure drawn in the Ketcher once the user is finished and presses the save button. The placement of this button can be seen in figure 15. Both the front-end and the back-end both try to clean the structure path when they have repeated structures. Unclean paths are mostly caused by the user using the undos and redos tools within the Ketcher. However, the prediction tool does struggle to handle users using the rubber tool to remove components because the Ketcher API does not support the understanding of the user's activity in the Ketcher. This is something that is discussed in future work.
The second call is a function call to get a list of predictions generated by the server, given the current SMILES string the user creates in the Ketcher. The metadata of the user is also sent along with the SMILES; the result of this is then dealt with and displayed by the prediction viewer.

6.1.4 Prediction Viewer

The predictions created on the back-end are sent in a list, with each element of the list containing a structure object, like in the save request, but this time there is also a list of structures that represents the path from the structure in the Ketcher, at the time the request was called to the end prediction. This is so when the 'save' request is called later, there are no gaps in the path structure . If the extra path information was not sent with the predictions, the structure would lose steps and information.
Each element in the prediction list is displayed using a method in the Ketcher tool that takes the molfile and creates a molecular structure formula image. This is different from what is created in the Ketcher because the images are non editable. The method also takes as a parameter an HTML element to attach the image to within the web page. In this application, the images generated from the prediction list are placed into one of four boxes underneath the Ketcher drawing tool. The four bootstrap panel boxes display the predictions in order of the probability that the user might want that structure. The user can then click on one of these boxes displaying the prediction and forward their current drawing to that structure.
Only four prediction boxes are used which can be seen in figure 15. This is due to spatial issues of trying to fit as many structures onto the screen without the need for a scroll bar or making them too small to recognise. However the predictions after four don't normally have a high probabilistic score and are statistically not what the user is searching for. It is possible to display all predictions if wanted, as the back-end returns more than just four.
The front-end design for the prediction was based on the Google search drop down menu used for displaying typing predictions. It was considered having the predicted structure appear faded in the background of the Ketcher when a user hovered over a structure prediction panel or just in general as the user drew but the restrictions of the library did not allow for this.

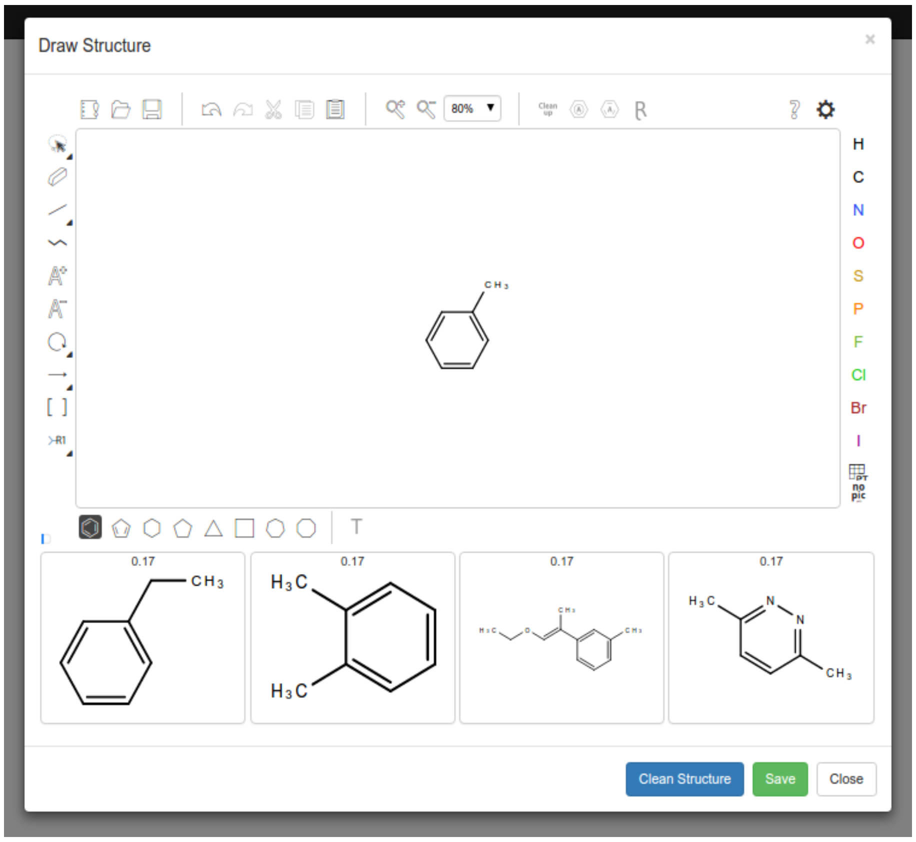


Figure 15: Prediction Tool

6.2 Database Implementation

The initial plan for implementing the database was to use a graphical database (see chapter 2.4.1) but the main disadvantage of a graphical database over a relational one is the fact they have not had the same amount of research time invested in them. This means that creating efficient queries in a graphical database can be hard, especially if there is a lack of experience in working with graphical databases. However, if we can formulate an efficient query, then queries that use associative relationships are considered to be much quicker than relational databases queries, which rely on many costly table join operations
After consideration, the decision was to use Mariadb as the database because it was something that was available from the School of Computer Science. In addition, SQL databases and how they operate were familiar. Another fact considered was the size of the project, meaning that using a relational database would not effect performance enough to cause noticeable speed differences. Performance is something that might need to be considered for the application

in the future. Looking at the quires within the application however, only one would benefit from using associative relationship queries compared to relational table joins.
To ensure the database could store a graph-like structure, a table of nodes was required. The nodes were represented in rows within a table called 'structure' with a SMILES string as the primary key. Also required was an edges table that had a composite primary key value of columns called 'smiles_to' and 'smiles_from'. Both of these columns were then foreign key values to the 'structure' table. Finally a table called 'edge_metadata' was created, which stored the statistical data describing each edge. This can be seen in the E-R diagram in figure 16.

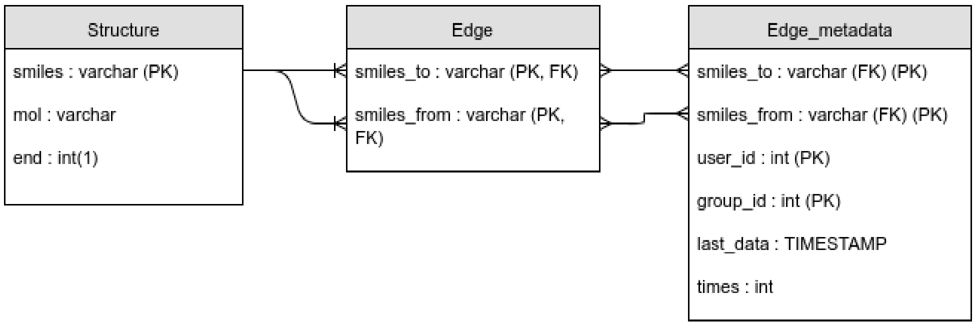


Figure 16: Database Details

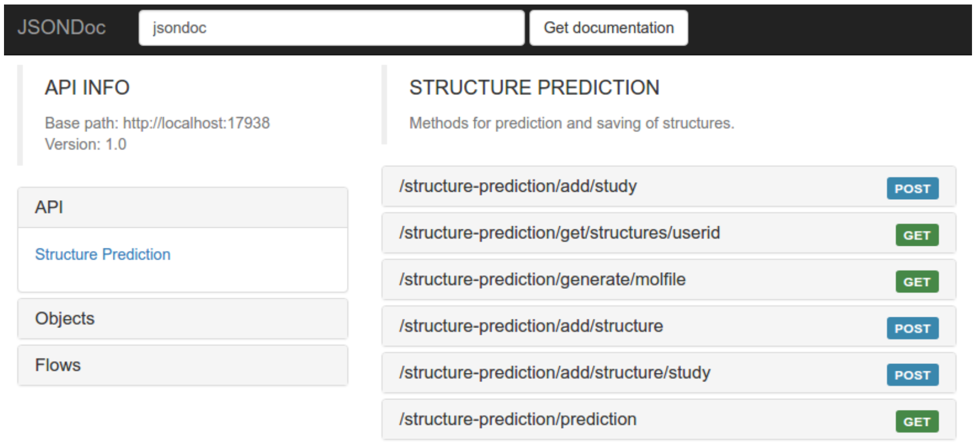
The plan was initially not to store the molfiles in the database because of their length in size; as the database grows, there is the possibility of speed issues being created. Instead the application was going to use the RDKit to generate molfiles for structures given the SMILES. However, it was chose not to do this as the coordinates for structures were change by the RDkit generator, resulting in a user having less chance of recognising the structure in the prediction list, because of the alteration to the structure. It seemed more important to ensure users might recognise structures and use the tool rather than reducing the size of the database due to the risk that it might cause performance issues into the future.

6.3 Back-end

6.3.1 Rest API

As mentioned in 4.2.2, Spring Boot is used to build the restful API. One massive benefit to doing so is the Jackson library it uses to automatically parse JSON objects sent to it into JAVA instances. This meant that difficulties of dealing with parameter checking of requests could be ignored. The two main requests that can be made to the application are the saving of a structure and the generation of a list of predictions. Both were mentioned in the front-end section

also.
The saving of the structure is a simple request for the application. The front-end must generate a POST request with the body being the path of the structure drawn, along with the user ID and group ID as header parameters. It sends these parameters to the IP address of the server, along with 'add/structure/' address. The application takes the structure path and for each element in the list, it starts by saving the SMILES and molfile into the structures table if they do not already exist. It then takes each adjacent element in the list and adds them into the Edge table if they do not already exist. Finally, it adds a row in the 'edge_metadata' table, again if that does not already exist. If it does already exist, it adds onto the count for the number of times that a user within that group has taken that step. If it does not exist then it enters a new row into the table sets the times count to one.
The second request, which is the generation of predictions, starts with the front-end formulating a GET request to the server using the address path "/prediction". The application then deals with the request by generating a Bayesian network, and returns a list of predictions based on the Bayesian network model. For more information on the API you can refer to the documentation. Appendix section D describes how it can be obtained. Figure 17 shows a screen shot of the documentation. This was important to create if the application was going to be moved over to NOMAD, giving a clear, precise format of communicating with the server.



Base path: http://localhost:17938
Version: 1.0

Methods for prediction and saving of structures.

/structure-prediction/add/study POST
/structure-prediction/get/structures/userid GET
/structure-prediction/generate/molfile GET
/structure-prediction/add/structure POST
/structure-prediction/add/structure/study POST
/structure-prediction/prediction GET

Figure 17: Screen shot of API documentation

6.3.2 Building The Bayesian Network

To generate the Bayesian Network, the application started by getting the data in the correct format to generate the probabilities for the network. Three main linked-listed hash-maps were then created by querying the database for the relevant data associated with user, groups and choices. These maps store infor-

mation on which users, group and structures picked have been done altogether in the past. This data is required when calculating the probabilities.
Once this data was set up, the application uses the SMILES library to create the network outline generating the user, group and choice nodes and adding the vertices between them. It then iterates over the maps and uses the Naive Bayes formula to input the correct probabilities for each row of data in the tables.
Finally it sets the user and group within the Bayesian network, given to it from the initial request and orders all the possible prediction choices the network found by their probability.

7 Evaluation

This chapter will try to evaluate whether the application built can answer the hypothesis proposed in section 1.3. It will measure how participants use the system with and without the tool and consider whether there is any significant difference in speed and errors. This section also explores whether the use of different metrics such as the user ID or group ID have different effects on the speed of users using the system.

7.1 Participants

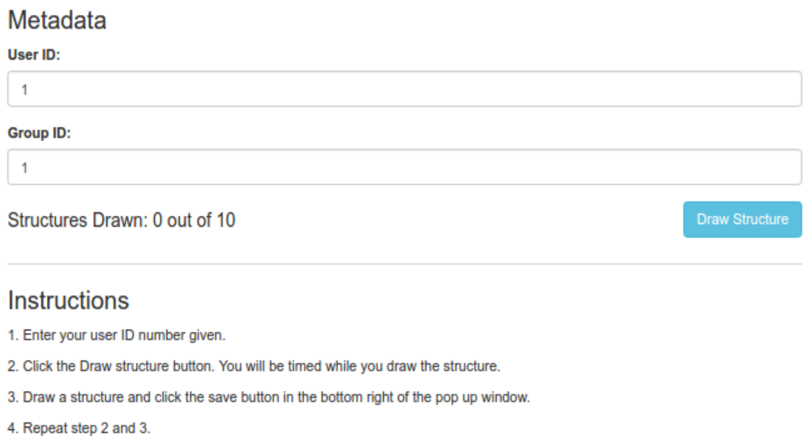
The study required Chemistry students or lecturers that were familiar with chemical structure editors, and comfortable with most of the universal operations of chemical structure editors. This requirement meant students had to be in their 3rd year or above. The majority of participants located were PhD students or those in their fourth or third year of study. In the appendix C is the data associated with the anonymous users, including their specific level.
The result of this requirement meant that it was difficult limited to locate participants easily and necessitated advertisement in the school of Chemistry department in St Andrews. Dr. Tomas Lebl was instrumental in helping to locate participants and also allowed the use of the NRM lab to carry out the study.
For the first part of the study, 25 Chemistry students were involved; 14 of that group completed the second part.

7.2 Process

The study was split into two parts: data acquisition and the measurement of the users performance when drawing structures. Only once the first part was completed by all users could the second part begin, because the study needed to insure that all users had the same fixed database on which predictions were based. Before starting, each user was given the opportunity to make themselves familiar with the drawing tool. Some of the main differences between the Ketcher and other drawing tools was explained to users, such as placement of tools in the tool bar.
Each user was given a user ID and a group ID number to fill in to the metadata form before the study began. This was required so the predictions could be catered to the participant.
The intention of the group ID number was to associate users to their research group but limitations with gathering enough participants of certain research groups restricted this. Instead the group IDs were split into the levels of study of each participant. More specifically, group one was made up of one 1st year PhD student, two 2nd year PhD student, one 3rd year PhD student and one postdoc. The second group, group 2 was made up of four 4th year undergraduates and three 3rd year undergraduate students.

7.2.1 Data Acquisition

This part of the study required to populate the database with structures and metadata associated with the structures. This was done by participants being asked to draw ten structures with which they were familiar or on which they were currently working in their studies. The viewing of predictions was turned off for this part of the study and participants drew structures from memory.
The time users took to draw each structure was recorded, along with the number of times the user used the undo, redo or rubber tool. This information however, was not used in the result. Figure 18 shows the interface given to the user to complete the first part of the study. In total from part one of the study, 241 structures from 26 participants were collected and from the 241 structures, 2704 user steps were recorded.


Metadata

User ID: 1

Group ID: 1

Structures Drawn: 0 out of 10

Instructions

1. Enter your user ID number given.
2. Click the Draw structure button. You will be timed while you draw the structure.
3. Draw a structure and click the save button in the bottom right of the pop up window.
4. Repeat step 2 and 3.

Figure 18: Web page for the data acquisition

7.2.2 Record User Performance

The second part of the study attempted to record data on whether the system could give useful predictions to the user using the data gathered from part one. The learning of the system was frozen, meaning any structure drawn by the user was not saved in structure, edge or edge_metadata table and only the data required for the study in the study_data table.
The participant was shown a structure on screen and asked to draw it using the tool. In this case, they were asked to copy the structures rather than being asked to draw the structure from memory, as they did in part one.. The user copying the structures meant the study could focus on the speed of the tool and not the speed of the user's memory. The user was asked to draw 28 structures. This involved drawing four that the user had drawn previously in part 1 and three that another user had drawn. The 7 structures were then repeated 4 times each, making up the 28 structures.

Every time a structure was repeated it would have a different type of prediction, meaning there were four different types of prediction. The different types of prediction applied were given an ID number and were as follows:

0. No prediction shown
1. The general populations data plus user data
2. The general populations data plus the group data
3. The general populations data plus the user data and the group data

The combination in which the user was given these structures and the type of prediction for each were randomly generated before they started. Figure 19 shows an example of an order of structures and prediction types that might have been given to a user.

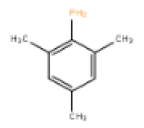
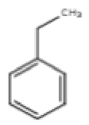
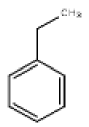
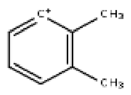
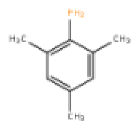
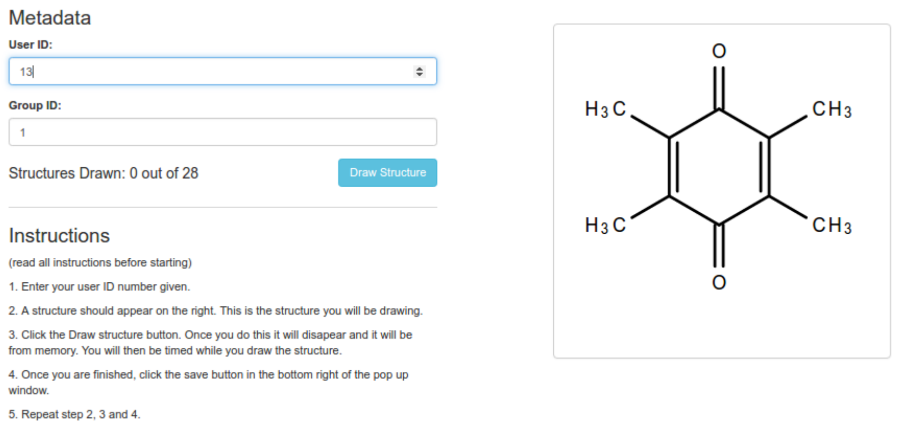
Total Drawn:	1	2	3	4	...	28
Structure:					...	
Prediction Type:	3	1	0	3	...	2

Figure 19: Example of format of structure and predictions.

To record the data a request to the applications API was created that took data associated with the drawing process of the structure. The application then saved this into a separate study_data table in the database. The data consisted of:

1. A timestamp of when the user starting drawing the structure
2. A timestamp of when the user finished drawing the structure
3. The type of prediction offered to the user
4. The number of times the user used the undo button
5. The number of times the user used the rubber button
6. The SMILES the user drew
7. The SMILES the user was meant to draw
8. The number of predictions used

This data was created on the front-end and sent to the server to save once the user had finished a structure. This was in case of a crash, allowing for all data structures to be saved up until the point of the crash. Figure 20 shows the interface given to the user to complete for the second part of the study.



Metadata

User ID:

13

Group ID:

1

Structures Drawn: 0 out of 28

Instructions

(read all instructions before starting)

1. Enter your user ID number given.
2. A structure should appear on the right. This is the structure you will be drawing.
3. Click the Draw structure button. Once you do this it will disapear and it will be from memory. You will then be timed while you draw the structure.
4. Once you are finished, click the save button in the bottom right of the pop up window.
5. Repeat step 2, 3 and 4.

Figure 20: Web page for recording user performance

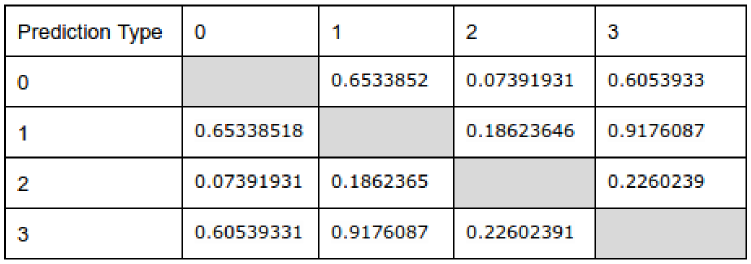
7.3 Limitations of Evaluation

Similar to most user studies, there are a number of limitations to the study in regard to users' actions that are uncontrollable towards effecting the result. One of the biggest issues within this study was users' ability to learn to use the application better as the study progressed. A random order of structures and prediction types were introduced in part 2, to try and stop structures shown to users at the end of the study being significantly faster than others. The random order would hopefully avoid a patterns being introduce for certain structure because of the same ordering for every user. However, there was also an issue when users were asked to draw the same structure more than once as the user would most likely be much quicker at drawing that structure on the last occasion compared to the first.
Another issue that could be minor but is worth discussion, is that the participant population are are not widely different. All participants share the fact they attend St Andrews university within reasonably small research groups. It is possible because of this they share abilities and habits when drawing structures or evaluating structures that people from outside the School of Chemistry do not have.

7.4 Student T-Test

The simplest thing to start with when beginning to evaluate the data was to run a paired student t-test between each prediction type to see if there was any

statistical difference between the types. The paired just means that I made sure that each data set was sorted and in the same order before being inputted to the t-test.
To start the student t-test, the data was split into four groups by their prediction type - user, group, user plus group and no prediction. On the splits the student t-test was run in every combination. This was unconvincing towards the H2 hypothesis that the prediction type, user plus group would give the users the best predictions. In truth, most of the P-values were extremely high and far away from the desired value of 0.05. Figure 21 gives a table breaking down the p values for each prediction type comparison.



Prediction Type	0	1	2	3
0	▨	0.6533852	0.07391931	0.6053933
1	0.65338518	▨	0.18623646	0.9176087
2	0.07391931	0.1862365	▨	0.2260239
3	0.60539331	0.9176087	0.22602391	▨

Figure 21: The p-values for each prediction group compared against one another

Out of the values in figure 21 the best significant difference was found between the group and no prediction type where the p-value was calculated to be 0.07. This is still not near the desired value of lower than 0.05 but is better than the rest of the comparisons. This is more than likely down to random chance because the types of prediction were not drastically different in the way they made predictions and the expectation was to have similar p-values.

7.5 Generalized Linear Model (GLM)

The t-tests results gave the need to explore other options and exploring all variables in predicting the time it took a user to draw a structure seemed like the next step. A Generalized Linear Model (GLM) seemed appropriate because it can model the possibility of more than one variable being important. In this case a number of variables can contribute to the time taken by a user to draw a structure.
To begin with the default GLM used by R seemed like it would allow us to see the variables importance but plotting the density of time taken variables showed that the distribution was infact skewed. The solution was to change to using a fitted gamma GLM representation rather than a normal GLM. Figure 22 shows the skew of data to the left which forced the change of model type.

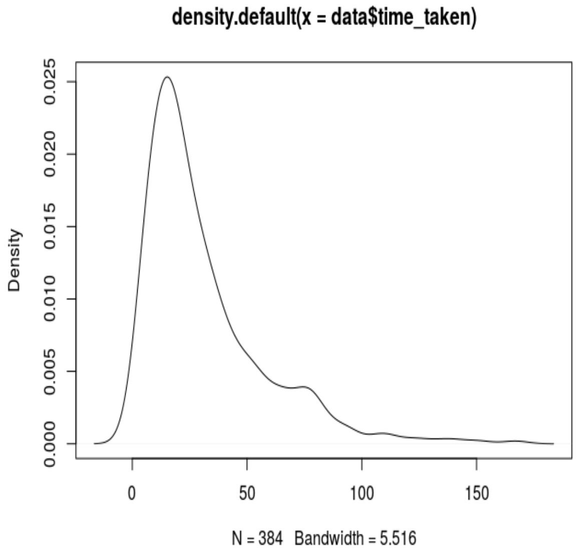
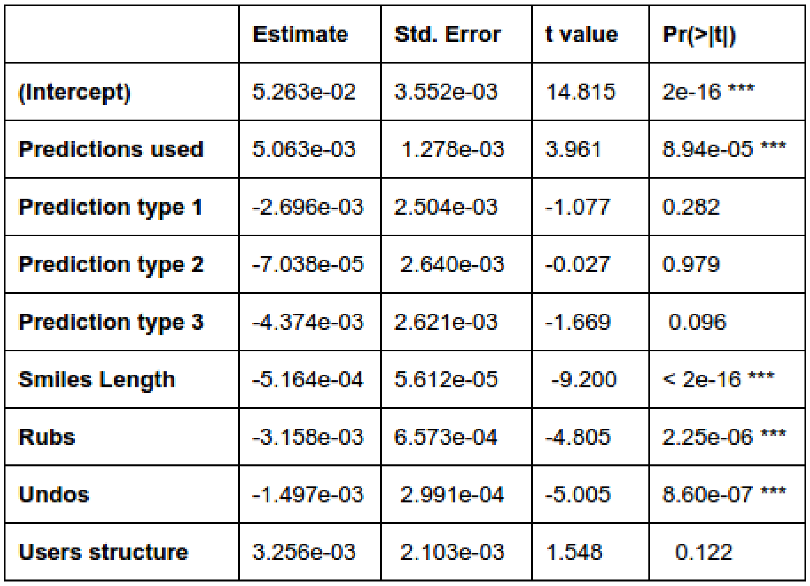


Figure 22: The plotted density of GLM

Using the summary function on the GLM, it outputs statistical data on each variable contained in the model and their contribution to predicting the time taken by a user. It does this by doing a t-test on the model with and without each variable. The p-value is outputted by the function and this refers to the significance of a variable in predicting the time taken for a user to complete a structure within the model. Figure 23 shows the p-values for the initial model using all the variables from the data-set that might be significant.



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	5.263e-02	3.552e-03	14.815	2e-16 ***
Predictions used	5.063e-03	1.278e-03	3.961	8.94e-05 ***
Prediction type 1	-2.696e-03	2.504e-03	-1.077	0.282
Prediction type 2	-7.038e-05	2.640e-03	-0.027	0.979
Prediction type 3	-4.374e-03	2.621e-03	-1.669	0.096
Smiles Length	-5.164e-04	5.612e-05	-9.200	< 2e-16 ***
Rubs	-3.158e-03	6.573e-04	-4.805	2.25e-06 ***
Undos	-1.497e-03	2.991e-04	-5.005	8.60e-07 ***
Users structure	3.256e-03	2.103e-03	1.548	0.122

Figure 23: The summary of the GLM model results, showing the significance of each variable when creating a model to predict the time taken for users to draw a structure. 'Predictions used' means the number of predictions a user clicked and users structure is a Boolean stating if the structure drawn was initially that user's structure from part one.

The stars, in the $Pr(> \|t\|)$ column, give us a quick indication of the P-values. In this case the three stars means the p-value is lower then 0.0001 and nothing indicates a value higher than 0.05.

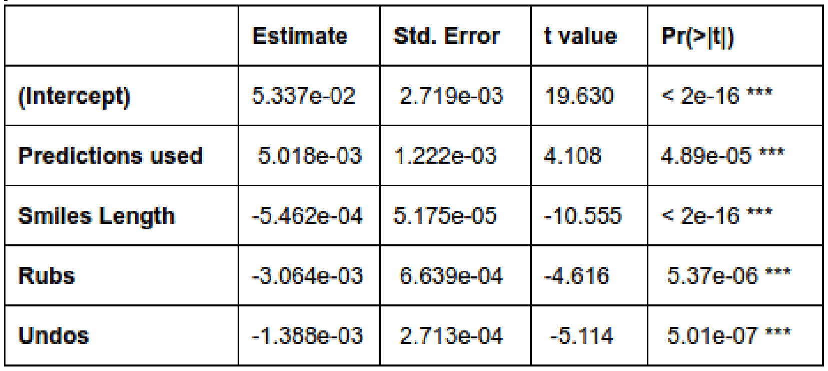
7.5.1 P-Values of Variables in GLM

Using this model's data it is possible to look at obvious effects some of the variables might have before looking deeply into the prediction tool.
The first variable looked at was the length of the structure and it seems obvious to say that a bigger structure, on average, takes longer to draw than a smaller structure. The variable SMILES length variable was added as a rough estimate of the size of a structure and this works because the SMILE strings normally grow given each atom or bond added. It is worth noting that, because the Gamma model uses an inverse linker function, the values that come out are inverted. This means looking at the t-value of the smile length, it suggests that as the smiles length grows, the variance of time increases by a t-value of 9.2. The p-value is far less than 0.0001 meaning there is a very large chance that the SMILES' length directly effects the time taken to draw a structure.
Another effect then considered from the data is that if a user is forced to use the rubber or undo tool, due to a mistake, they will most likely be far slower in

completing the structure. From the data, it can be seen that the t-value affects the time by 4.8 when the undo button was used and 5 when the rubber was used. Both these are said to be highly significant by their p-values and gives us a high level of confidence that these contribute to predicting the time.
More importantly, the variable predictions used within the GLM can contribute to answering our Hypothesis H1, that when a user uses a prediction there is a direct reduction in time. Predictions used seems to follow the same trend as variables mentioned already, SMILES length, rubs and undos. With a p-value lower than 0.0001, we can have high confidence that the model says there is a reduction in time when a user clicks the prediction. This is very positive to our H1 hypothesis statement.
The GLM gives us the same sort of results for the prediction type that was seen when running the paired student t-test. The p-values are higher than 0.5 and indicates there is no real significance towards the model when predicting the time taken to draw a structure. Possible reasons for this are discussed in section 7.6.

7.5.2 Removing unnecessary Variables from the GLM

GLM suggested that the variables, user structure, prediction type 1, 2 and 3 give no contribution towards the model. These values needed to be removed from the model because they may have been affecting the contribution of other variables. They were removed one at a time, to make sure that removal of one did not cause the other variable to become more important. This is called backwards selection.
The final models' values are shown in figure 24 and show an increase in the variables' contribution to predicting the time taken by a user to draw a structure.



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	5.337e-02	2.719e-03	19.630	< 2e-16 ***
Predictions used	5.018e-03	1.222e-03	4.108	4.89e-05 ***
Smiles Length	-5.462e-04	5.175e-05	-10.555	< 2e-16 ***
Rubs	-3.064e-03	6.639e-04	-4.616	5.37e-06 ***
Undos	-1.388e-03	2.713e-04	-5.114	5.01e-07 ***

Figure 24: The summary of the GLM model results, once the prediction type and user structure was removed.

7.5.3 Plot of GLM

Throughout the earlier steps, the plot function in R was used on the glm model to understand if the model was correct for the variables it was using. Figure 25 shows the plotting function being called on the GLM.

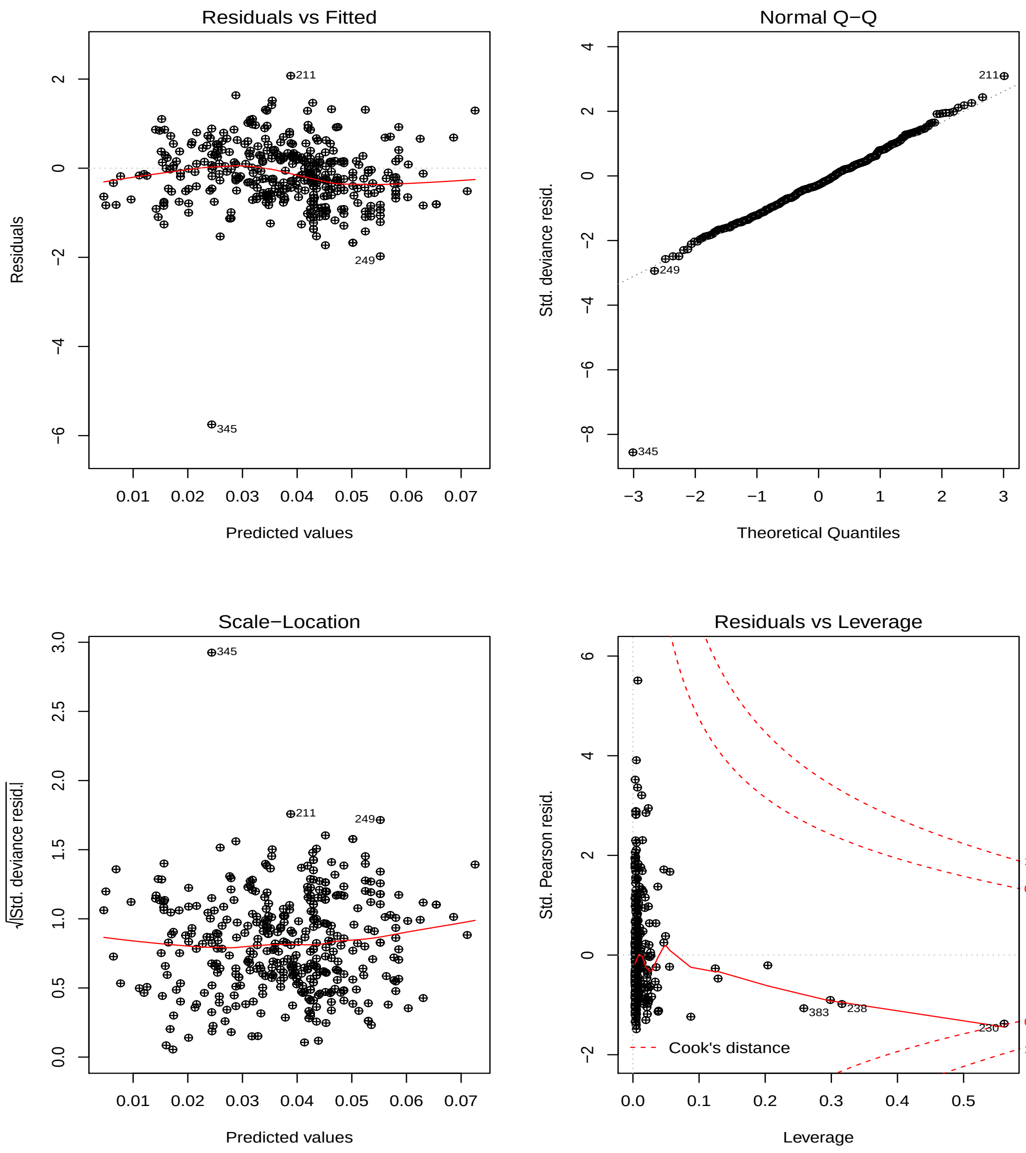


Figure 25: Graphs that correspond to the GLM

The following graphs are useful in stating what residuals are not following the correct distribution and identifying outliers. The first graph, Residual VS Fitted puts the residual values against the fitted generalised linear prediction model, and the values that are drastically far from the prediction line are called outliers. The Normal Q-Q graph is used to see if it is plausible to say that the values all come from the same theoretical distribution and in this case the type of distribution is normal [7]. The Scale-Location is the same as the Residual Vs Fitted plot but uses the square root for the residuals instead. Finally the Residuals vs Leverage uses Cook's distance to try to identify points which have more influence than other points [22] and are said to have high leverage. None of these graphs should have any discernible patterns or any points that do not agree with the other data points.
In this case, from looking at the graph's Residual Vs fitted, the Normal Q-Q and Scale-Location, point 345 is an outlier. This point does not agree with the rest of the plotted residuals. Figure 26 shows the generated molecular structure given the SMILES string from data point 345.

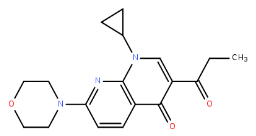


Figure 26: Molecular Structure of Outlier Data Point

Looking at the data associated with structure 345, it took 0 seconds in time to complete. This is of course an error and does not seem normal in comparison to the rest of the data times. This value can only be put down to an error within the code or a glitch within the response time of requests. Removing this data point from the data set is the only course of action. Figure 27 shows the graphs once structure 345 was removed.

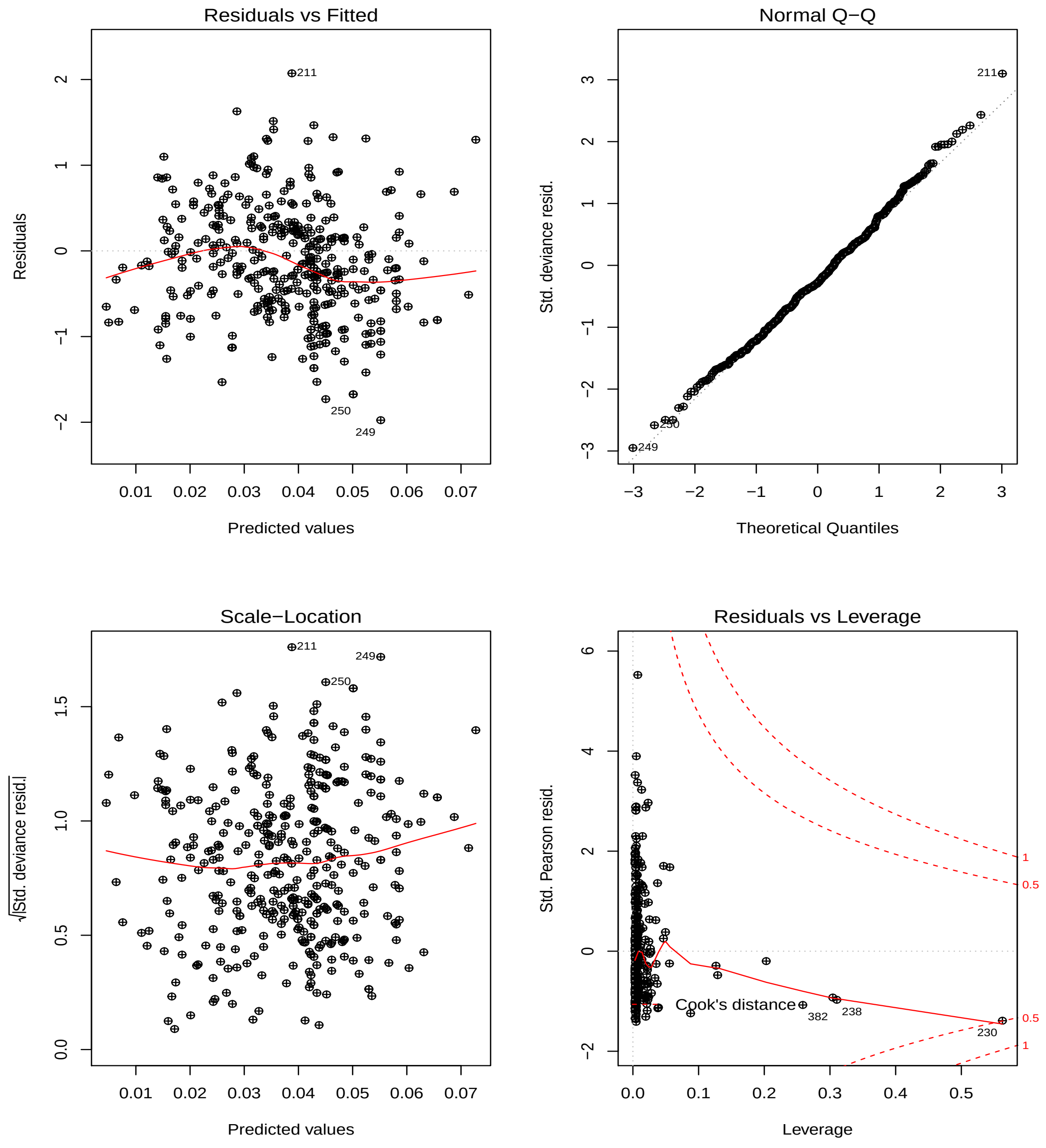
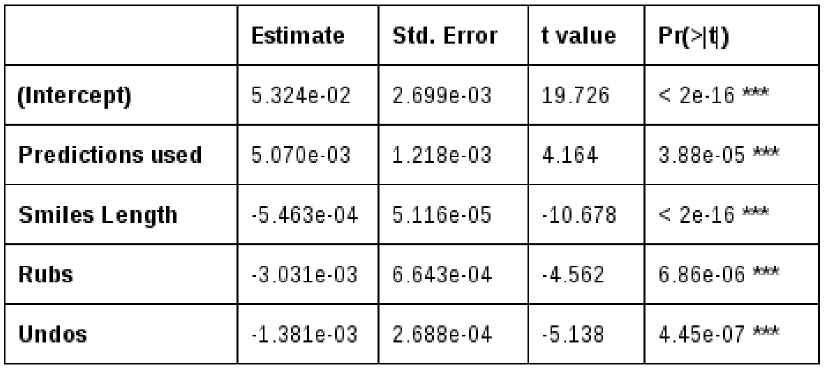


Figure 27: Graphs that correspond to the GLM

The p-values from figure 24 were changed because of the removal of data point 345. Figure 28 shows the new values, and in fact the significance of the predictions used was decreased and the significance of the rubbers used has increased. This change from the initial model is because the structure removed used two rubbers. However, these changes do not change the underlining meaning of what was said in section 7.5.1.



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	5.324e-02	2.699e-03	19.726	< 2e-16 ***
Predictions used	5.070e-03	1.218e-03	4.164	3.88e-05 ***
Smiles Length	-5.463e-04	5.116e-05	-10.678	< 2e-16 ***
Rubs	-3.031e-03	6.643e-04	-4.562	6.86e-06 ***
Undos	-1.381e-03	2.688e-04	-5.138	4.45e-07 ***

Figure 28: The summary of the GLM model results, once the prediction type and user structure was removed, and the outlier has been removed.

7.6 Prediction Type Result

There are a number of possible reasons for the prediction types giving no significant differences between one another. The reasons are most likely down to the data acquisition and the data sampling size but also to the fact that a prediction tool was not something the user had seen before.
A problem with the group metric was the fact they were artificially created, where users were grouped by their academic level and not research group. The research group was suggested by Dr. Tomas Lebl and Simone Conte to be a responsible variable to group people by, because they are far more likely to be drawing structures similar than to, say, people in the same year group. Finding a real life sample of research groups was difficult but doing so might improve the results between the prediction groups.
The user suggestions might also have had no contribution to the overall prediction because the data-set was not large enough for users to have to pick between a large number of differing structures. For example, a user may be drawing a structure and is returned a list of predictions; if the database is large then the number of possible structures he can pick from will be large and they might not fit within the screen. Without the user metric in the larger data-set, the structure could be pushed to the back of the suggestions, off screen because there is no user preference to push the probability higher for the user's structures. In a smaller data set this is the opposite: users will more likely only get one or two predictions displayed and the order does not matter because they will all fit on screen. In the case of the study, this happened often and meant the difference in prediction for users was no different from any of the other types. This could also be said for group.
The biggest issues though was the tool struggling with users drawing structures differently, for example, placing bonds earlier in places they might not have

before. If a user was to alter his drawing, meaning he did not meet a step containing a SMILES string which they or another user has used before, the tool could fail to make any prediction towards the structures the user was trying to draw. This meant that even though predictions were switched on, the users could not get predictions. This could be a fundamental reason for there not being any significant difference between the groups. A solution might have been to make users draw the same structure more than once in the data acquisition stage. Another solution would have been to have a greater number of structures in the database because a larger training set would mean more predictions to the user.

7.7 User Errors

Another area worth exploring is whether there is a significance difference in the number of errors made by each user when they used the prediction and when they did not. To do this, the data sets were split into two groups; group one, when the user used the prediction tool and group two, when they did not. A new row first needed to be generated called correctly_drawn. The correctly_drawn column indicates if the structure drawn, found in column smile_drawn, matched what the user was asked to draw, found in column smiles. To generate this row was difficult because SMILE strings are not unique for all Molecular structures and this meant that, to be able to compare SMILES, they had be converted to a canonical SMILE. To do this a python script was written that used a library called Indigo Toolkit [12] to translate the SMILES into canonical SMILES. The script read each row in the data set and converted the column values smile and smiles_drawn to canonical SMILES. The script then compared the columns and outputted the result of this comparison, true or false, into the new column correctly_drawn.
Once the new column was generated by the python script, the next step was to look at the total times each group got the structure correct. When predictions were used by users they managed to get 148 correct out of 154 structures, $\approx 96\%$ and when the users did not use prediction they got 185 out of 230 correct, $\approx 80\%$. A Welch's t-test was then run on each group. This applied because the two groups, prediction not used and prediction used were split into sizes of 230 and 140 respectively. The p-value was calculated as 4.72e-07 which is highly significant at a value lower than 0.0001. This gives high confidence that when predictions were used by users, the structure was far more often correct.
We then decided to do another Welch's t-test using the same groups but using the total number of errors made by users. To do this, a new column had to be generated again but this time it was simply adding the number of times a user used both the rubber and undo. Overall when users used prediction, they made 119 errors compared to when users did not use prediction, when they made 304 errors; considering however, there were far more structures within the group

[12]Indigo ToolKit `http://lifescience.opensource.epam.com/indigo/` [Online;last accessed:4/5/2017]

prediction node used, proportionally this is not massive. Another effect that needed to be considered was the size of structures, because the likelihood is that the bigger the structure, the more likely a user is to make an error. The average length of structures contained in both sets was calculated for this reason and the results were that when predictions were used, they gave an average SMILES length of 30.94805 and when prediction was not used the average was 32.26957. However the t-test did give a t-value of 2.9864 and a p-value of 0.003, which can be said to be significant. Furthermore, taking the mean of both data sets on total errors showed on average the rubber tool or the undo button was used 0.7727273 time when prediction was used, compared to on average 1.321739 of the time when prediction was not used.

7.8 Feedback from Users

To gather qualitative feedback, an email was sent out to the users on the day each of them finished the study, containing a link to the form. The written feedback was used to evaluate users' thoughts about the application and give some information on how user friendly the tool's interface was. Out of the 14 people that completed the study, 9 of them gave feedback. There were two yes and no questions, 6 rating questions between agree and strongly agree and two optional long answer questions.
100% who voted said they saw the point of having a predictive tool for molecular structures. 77.8% said they found this application useful and the other 22.2% said they did not know. 77.7% voted between 4-5 out of 5 that the prediction tool aided them in producing a structure and this was backed up by the statistical model talked about in section 7.5.1. Interestingly though 22.2% gave a value of 3-4 out of 5 that they found the tool distracting. This could be down to the nature of the application being something a user has never seen before.
The written feedback was the most useful and gave a real indication of what users thought needed to be done to improve the application. Mostly they said they found the application idea useful and when predictions were available to the user, they saw a vast improvement in their speed. The main problem they found was its lack of predictions at times. This is directly down to the data in the database not being large enough. The result most likely would have improved, had different users drawn the same structures because this would allow the application to learn more than one way to draw certain structures. There were more complaints about the Ketcher drawing tool rather than the predictions. One user who gave written feedback directly stated they found the placement of the predictions on screen in the ideal position.
The overall feedback form can be found in the appendices B.

7.9 Conclusion of Evaluation

Overall, the GLM showed that when predictions were used, the time taken to draw a structure was decreased with a p-value lower than 0.001. The study did fail to state however that having the predictions in place overall improved the speed of users drawing structures and failed to define what prediction type is the most useful. This is most likely down to the dataset not being large enough and the tool's limitations when dealing with differing structures. It is possible to state though that when predictions were used we saw a noticeable difference between when users drew the correct structure and the number of errors made when prediction was used and when it was not. Not only was the quantitative data important, but the qualitative written feedback given was helpful and positive and mostly confirmed what the statistical data told us.

8 Future Work

8.1 Improvement of Data Acquisition

This is perhaps the most fundamental part of improving the overall predictions of this system. Currently, the application uses raw data created from what each user is typing with no real validation of whether or not what they have just created is useful information to the overall database. At present, the only validation is stopping repeated data with the undo and redo tools. There is nothing stopping the user from creating and drawing unrealistic structures that will not contribute to the overall predictions of the population.
One way to stop this might be to use only the drawing sessions of individual users in the predictions. Doing this would produce far fewer suggestions for each user and result in no means of predicting structures the user has never drawn before, but the predictions it would make would always be relevant to the individual user. This is a very simple approach.
Another way would be to allow the data to build up over a long period, letting the population validate it slowly. Thus, the more users using the system, the quicker the predictions would improve, given that the majority of the users create structures correctly.

8.2 Explore Other Drawing Tools

The Ketcher drawing tool restricted this application in how much the prediction tool could interact with the user actions in the Ketcher. This means that it could not deal with the user using the rubber tool effectively. This at times caused unrealistic structures for the database.
From the study, it was noticed that users tended to make mistakes by placing the incorrect components onto their structure because the Ketcher did not support a preview on hover. This is something MarvinJs supports and may improve the usability of the system. An example of this can be seen in figure 29, where Ketcher is component (A) and MarvinJs is component (B).

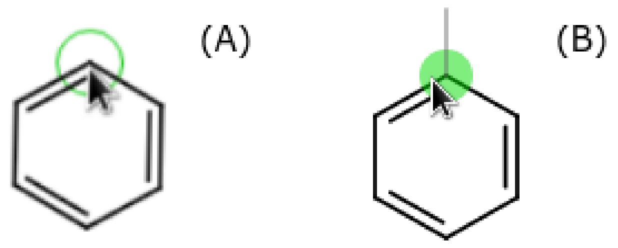


Figure 29: The Preview In MavinJs(B) and not in Ketcher(A)

With this preview in mind, the prediction tool might be more likely to be adopted by users if the prediction itself was suggested with a preview in grey. This was something consider at the start of the project but because of API issues with the Ketcher could not be implemented.

8.3 Improve Prediction Display

From the user feedback sheet, 22.2% scored the prediction tool a three or four out of ten for the level of distraction whilst they used it.
One reason for this could be the unfamiliarity of the tool for the users, but improvements could be made to the display, for example resizing the prediction preview, alterations to colour or altering the number of predictions shown. These changes could be made and tested using A/B testing. This would work best if the application was launched on NOMAD because the system would have an easily accessible pool of users. This would simply mean giving one group of people the tool where one visual variation to the prediction tool has been added and one with a different visual variation added. Then for each group, the number of times the users chose to use the prediction tool would be recorded, and compared between both groups. If one of the visual changes made users use the tool statistically significantly more than the other, the interface could be changed permanently to that setting [17].
The distraction mentioned by users could also have been caused by the constant recalling of the predictions for the slightest change to the drawing. For example, when a user drags a bond on screen, in the Ketcher, each small movement calls the server to give predictions. This could easily be stopped by waiting until the user releases the mouse before calling the request but the Ketchers API restricted this.

8.4 Improve Predictions

8.4.1 Understand SMILES

This application does not attempt to understand the internal meaning of the SMILES strings and these are just treated as strings, and used just for comparisons of structures. If the program was to use the rules of the SMILES strings it could make better educated predictions or even rule out parts of the database graph.
An example where predictions could potentially improve would be to categorise the SMILES strings in the database using common functional groups. Figure 30, shows two structures contained in the Vanillyl group [6]. When the structures are saved, the system look at the SMILES strings and attaches a list of strings naming the functional groups in which the structure is contained. The system could then make predictions with an added node in the Bayesian Network that takes into account functional groups. This means that if the user is drawing a structure and that structure shares one functional group with suggestion A, then there will be a greater probability that the user takes that step, than suggestion B (as it shares zero functional groups with suggestion B).

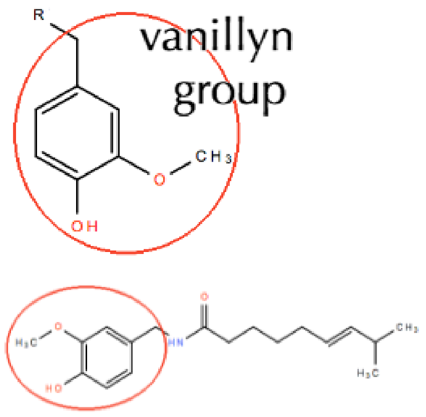


Figure 30: Vanillyl functional group in two structures [6]

To recognise these functional groups, the system would require to have to have a database of the SMILES and to search the incoming structures for the functional groups. For example, Vanillyl is contained within the database of functional groups. If an incoming structure contains the string representation of Vanillyl within it, the system can then add the functional group of Vanillyl into the data associated with that structure.

8.4.2 Understand NMR Spectrum

There were suggestions at the start of the project of the possibility of incorporating the NMR spectrum into the prediction. This would involve considering a user's sessions in the NOMAD application and looking directly at that user's experimental NMR spectrum data. This would be an extremely difficult feature to add and would require high domain expert knowledge when trying to understand the spectrum data; this is something that even a human expert can struggle with. The impurity of spectrum data can cause problems for mapping them to structures.
However predictions tools for these spectra do exist. Using one of these tools, it is possible that the output of one of these tools come aid predictions. A possibility would be that when a user in the NOMAD system starts creating the molecular structure formula to attach to the spectrum data, the prediction tool for the spectrum is run. The structure output of prediction is then incorporated into the molecular structure prediction by doing a likeness comparison with all the suggested next step structures. The system then adds an extra node in the Bayesian network that says if the structure is between a certain percentage of likeness, the probability is increased.

8.5 Other Approaches

8.5.1 NOMAD Likeness Search

An idea that was proposed from the start of the project was to use the similarity search, added into NOMAD 2.0 to generate predictions. Its operation is discussed in section 2.5.2 and the reasons for it not being used in this project, are detailed in section 5.1.

8.6 Integrate the Prediction into NOMAD

Some of the ideas when designing this project came directly from NOMAD, for example, NOMAD's architecture, client-server and the decision to use a Java rest API for the back-end server. This was to allow for a possible clean integration of the application at a later date. One of the main objectives of the project was to have all components of the system decoupled. Each component should be easily removed from one another. One reason for this is that if the application was to be imported into the main NOMAD project at a later date, it would be far easier to do so with a decoupled system.
Figure 31 shows the overall architecture of NOMAD 2.0; this project, if integrated, would be placed next to the experimental search within the user portal.

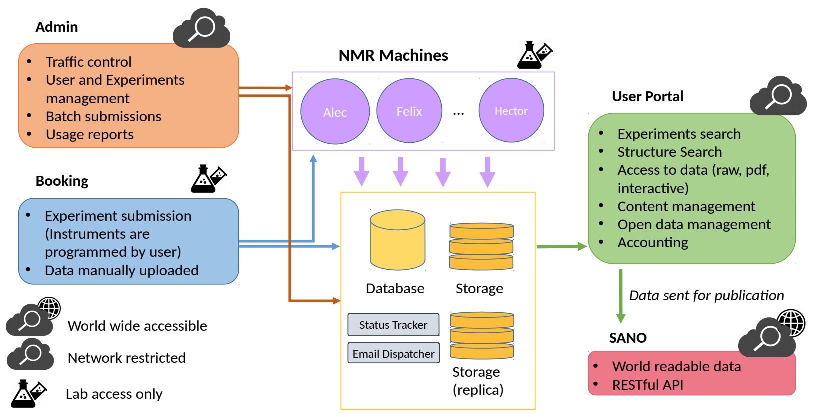


Figure 31: NOMAD 2.0 Architecture [5]

This project has proved that this type of predication can have positive results for increasing users' efficiency for drawing molecular structures and the NOMAD developers were very happy with the progress of the system; however many other improvements can still be made to it. This type of application has not been fully explored, and with little evidence to suggest it has been attempted before, more exploration is required before NOMAD would consider fully adopting it into their system.

9 Conclusion

The processing of big data within scientific research will continue for the foreseeable future and the amount of data that can be stored and gathered will continue to grow rapidly. The effective automatic management of this data will only become more important as the ability to gather bigger volumes and varieties of data grows.
NOMAD is a success story within the School of Chemistry for managing large amounts of NMR data. However the system is not perfect and the Schools of Computer Science and Chemistry in St Andrews continue to locate areas to improve it. Their underlying goal is to make it as efficient as possible for researchers to li ocate and store their data within the system.
This dissertation explored and evaluated a molecular structure prediction tool that could decrease the time it took users to draw molecular structures on NOMAD to allow users to quickly search for experimental data. It used machine learning to build up a graph of structures that could then supply the data to generate predictions using a Bayesian network.
Overall, the project met the requirements that were set out from the start of the project. The tool can make predictions, educating itself with past user data. The system can even make the correct prediction 100% of the time, if users follow the same pattern of drawing from a previous structure. The front-end is simple but also fits the purpose of the project and this is backed up by user's mostly positive answers within the study feedback sheet. Components of the system are as decoupled from one another as possible, with the ability to switch between any SQL database with a suitable schema or front-end that complies to the applications API.
Furthermore, from the evaluation carried out on user performance, results showed promise in creating an effective solution when the users opted to use the prediction to assisted when drawing structures, compared to when they did not. The study saw a reduction in time taken to draw the structure when users clicked on predictions compared to when they did not, given from a GLM. The GLM showed a p-value of less than 0.0001 between the model with and without the variable measuring the number of times a user clicked a prediction. On top of this, it can be stated with a high level of confidence, that users drew the correct structure more often with they used the tool, compared to when they did not. On the other hand, the evaluation failed to show that the application had a large enough significance difference in time between when the prediction was turned on and when it was not. A number of areas can be improved with the system to do this but large the handling of structures such as gathering a larger data-set to deal with structures being drawn in slightly different ways.
Along with this, the application could still be improved in areas such as data acquisition and the user interface. This was always going to be the case due to the difficulty in finding systems comparative to this project. A product shippable to NOMAD was always going to be difficult to create, but with this level of results a prediction tool , if introduced, will make a difference to efficiency of locating NMR data using molecular structures search.

References

[1] BayesFusion. Smile: Introduction. https://dslpitt.org/genie/wiki/SMILE:_Introduction, 2015. Last accessed 5 April 2017.

[2] Bondy, J. A., and Murty, U. S. R. *Graph theory with applications*, vol. 290. Citeseer, 1976.

[3] Burrows, A., Holman, J., Parsons, A., Pilling, G., and Price, G. *Chemistry^3: Introducing Inorganic, Organic and Physical Chemistry*. Oxford University Press, 2013.

[4] Castrounis, A. Machine learning: An in-depth guide - overview, goals, learning types, and algorithms. http://www.innoarchitech.com/machine-learning-an-in-depth-non-technical-guide/, 2016. Last accessed 6 April 2017.

[5] Conte, S., Fina, F., Psalios, M., Reyal, S., Lebl, T., and Clements, A. Integration of an active research data system with a data repository to streamline the research data lifecycle: Pure- nomad case study.

[6] Dalke, A. Molecular fragments, r-groups, and functional groups. http://www.dalkescientific.com/writings/diary/archive/2016/08/08/molecular_fragments_and_groups.html, 2013. Last accessed 5 April 2017.

[7] Ford, C. A residuals vs. fits plot. http://data.library.virginia.edu/understanding-q-q-plots/, 2015. Last accessed 5 April 2017.

[8] Jahnke, L., and Asher, A. The problem of data: Data management and curation practices among university researchers.

[9] John, G. H., and Langley, P. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (San Francisco, CA, USA, 1995), UAI'95, Morgan Kaufmann Publishers Inc., pp. 338–345.

[10] JS, M. Marvin js : Drawing chemical structures. https://marvinjs-demo.chemaxon.com/latest/docs/manual/Drawing-Chemical-Structures_17236086.html, 2016. Last accessed 5 April 2017.

[11] Kabacoff, R. I. Generalized linear models. http://www.statmethods.net/advstats/glm.html, 2017. Last accessed 6 April 2017.

[12] Lathrop, R. H. Bayesian networks. https://www.ics.uci.edu/~rickl/courses/cs-171/cs171-lecture-slides/cs-171-17-BayesianNetworks.pdf, 2017. Last accessed 5 April 2017.

[13] Marr, B. Why only one of the 5 vs of big data really matters. , 2015. Last accessed 5 April 2017.

[14] MySQL. What is mysql? `https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html`, 2016. Last accessed 5 April 2017.

[15] of Science, P. E. C. 6.1 - introduction to generalized linear models. `https://onlinecourses.science.psu.edu/stat504/node/216`, 2017. Last accessed 5 April 2017.

[16] Oracle. Java takes on the internet of things. `http://www.oracle.com/technetwork/articles/java/afterglow2013-2030343.html`, 2013. Last accessed 5 April 2017.

[17] Perekupa, M. How a/b testing works (for non-mathematicians). `https://blog.kissmetrics.com/how-ab-testing-works/`, 2016. Last accessed 5 April 2017.

[18] Proksch, S., Lerch, J., and Mezini, M. Intelligent code completion with bayesian networks. *ACM Trans. Softw. Eng. Methodol. 25*, 1 (Dec. 2015), 3:1–3:31.

[19] PubChem. Pubchem structure search. `https://pubchem.ncbi.nlm.nih.gov/search/help_search.html#IdSimi`, 2016. Last accessed 5 April 2017.

[20] PubChem. Pubchem substructure fingerprint. `ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem_fingerprints.pdf`, 2016. Last accessed 5 April 2017.

[21] Runkel, P. What are t values and p values in statistics? `http://blog.minitab.com/blog/statistics-and-quality-data-analysis/what-are-t-values-and-p-values-in-statistics`, 2016. Last accessed 5 April 2017.

[22] Spector, P. Linear regression. `https://www.stat.berkeley.edu/classes/s133/Lr0.html`, 2011. Last accessed 5 April 2017.

[23] Turner, H. Introduction to generalized linear models. `http://statmath.wu.ac.at/courses/heather_turner/glmCourse_001.pdf`, 2017. Last accessed 5 April 2017.

Appendices

Within the code repository submitted with this report you can the appendix files contained within the folder documentation.

A Ethical Approval Letter

Ethical approval letter is contained in a file, named ‘Ethics_Approval.pdf’

B User Feedback Results

The User Feedback results can be found in a file is named ‘feedback.html’.

C Anonymous Users Data

The data associated with the anonymous users that carried the study can be found in a file is named ‘participantsData.csv’

D Rest API Documentation

Documentation on the API is generated using a library called JSONDoc [13]. To access the documentation you must run the application (refer to the README file in the root directory of the project) and visted the link `http://localhost:17938/jsondoc-ui.html` given you are running the application on the default port 17938. You then search for the documentation by typeing jsonDoc into the search bar.

[13]JSONDoc `http://jsondoc.org/` [Online; last accessed:6/4/2017