Acknowledgements

Firstly, I would like to express my most gratitude to my supervisor, Dr. Robert Laramee, for his guidance and support. I am really appreciate the patience, motivation, enthusiasm and immense knowledge he provides during this project. Furthermore, I would like to thank him for his efforts in encouraging me to improve my research and programming skills. I would like to thank my family for their support and understanding for my study. I also would like to thank Mohammed Alharbi who assisted me in developing Tf-Idf features for the project. I thank Rhodri Fabbro for his valuable comments in proofreading my thesis. Finally, special thanks go to the Davais for their continued support.

Contents

List of Figures

1 Introduction and Motivation

Data sets have risen dramatically over the past few years, and these data sets have become increasingly complicated to analyse. How to deal with large amounts of data has become a challenge in certain fields [Laramee, b]. According to [Ward et al., 2015], when receiving large volumes of information, people tend to use sight as the main sense to understand it. Data visualization, as a mechanism using graphics to represent data [Ward et al., 2015], provides a good solution for exploring huge sets of complicated data.
As stated by [Williams et al., 1995], data visualization is defined as ”the visual representation of a domain space using graphics, images, animated sequences, and sound augmentation to present the data, structure, and dynamic behaviour of large, complex data sets that represent systems, events, processes, objects and concepts” [Williams et al., 1995]. By applying techniques of data visualization, more information can be explored.
Text data emerges in large quantities every day in newspapers, blogs, and social media. Hence, extracting information from text data is becoming highly needed. In some certain study areas, studying the relationship between words, sentences and texts' structure may help researchers to understand important information hiding in the text. For example, in an archaeological laboratory, analysing the text they found from a historic site may help them understand the dates of files, antecedent events, or the host of the grave, even without knowing the meaning of the ancient language. Similarly in the archaeological industry, techniques in text data analysis is fundamental and significant in translation study. Many institutes rely on knowledge of text data analysis to explore the variation of language in history, style of authors, as well as the social status of people in a particular period.
The ways to analyse and present text data have become a popular topic as the volume of the text data is often huge and complicated in format, genre, and morphology. For instance, languages inherited from

different roots may lead to different expressions when translating from one to another. Authors of different eras or regions may use different words to express the same things. The same contents may appear in different styles of expressions according to the purpose of the texts. Also, to deal with these problems, text data can be analysed and represented from lexical, syntactic and semantic perspectives [Ward et al., 2015], so that the unstructured text can be converted to structured data. Calculating frequency and weights of words can help to explore the information of content. There have been plentiful tools to visualize the structure of text data, such as Word Clouds, Word Tree, Tex Arc, etc. And for different research purpose, text data are often analysed separately in a single document and a collection of documents. One such collection of documents is *Othello*.

Othello, as one of the greatest tragedies of Shakespeare's plays, has been translated more than 60 times in German [Geng et al., 2011].The College of Arts and Humanities at Swansea University has a collection of 55 different German translations of *Othello*. The time span of these translations range from 1766 to 2010. And there are also different genres such as poems and prose, as well as plays. Applying data visualization techniques to help represent these text data will contribute to new research in the study of Shakespeare's work, and explorations into visualization. More concretely, the aims of this project are as follows:

- To develop an interactive visualization system that enable the researchers in the College of Art and Humanities to explore detailed translation information of different versions.
- To design a software of textual data visualization to display more information by compare different versions of translations, such as time span, genre, interpretation.
- To explore potential solutions in textual data visualization for difficulties in translation comparison, such as parallel text and data filtering.

Using textual data visualization as an aid to explore the text data of *Othello*'s translations will benefit for researchers to understand the changes, interactions, and impacts of these translation versions and cultures, time span, and styles [Alrehiely, 2014]. Based on the work of [Geng et al., 2015], [Alrehiely, 2014], and [Tom et al., 2012],we attempt to develop an interactive visualization system aimed to allow our users to view, compare, and analyse tokens in each version. The visualization tool will be designed to assist in viewing the variation of tokens in different translation versions, and in comparing the varieties of tokens after applying different methods to process the text data. Apart from the essential information about each version, such as the author and data of publication, there are three unique fields the data provides: the frequency of tokens, weight of tokens, and results from lemmatization for tokens.

The outcomes of the visualization system should be helpful in understanding the variation of word morphologies, varieties of text styles, and the complex features of the German language. It also facilitates improved comprehension of literature dynamics, the differences between languages, and the perception of translating cultures. Moreover, this project will provide a visualization tool for books, articles, newspapers, etc., to represent large sets of text data.

The German Shakespeare text data in this project, several special problems are caused by antiquated language, and poetic orthography. The former means that some words used in the 18th or 19th century may not be in the lexis of training corpora, if these are based on 20th/21st century sources. And by using the poetic orthography, take "verloren" (meaning: lost) for example, the word is normally written as "verloren", though can also be spelled verlor'n, or verlorn in some places in Shakespeare texts (the word normally has 3 syllables, pronounced VER-LOR-RUN, but the writer wants it to be spoken as 2 syllables, VER-LORN). This kind of situation happens a lot. Yet there are no effective algorithms to recognise these forms. To find a solution to these problems, some methods from Natural Language Processing

may be applied, such as lemmatization.
Choosing this project for my dissertation was on account of my interest in the field of data visualization and language analysis. The background of programming and language study will further my comprehension of data analysis and processing. Developing a project such as Translation Visualisation is becoming a significant topic for language studying and text data processing.
Following [Laramee, 2011], the rest of this thesis is structured as follows: Section 1 to section 4 are modified versions of work previously presented by the author in [Liu,]. Section 2 details the background research, along with the literature review, introduction to existing systems, and data characteristics. Section 3 details the specifications of the project, which includes the features specification of software and technology choices. Section 4 presents the approach of the project, time arrangement and potential risks. Section 5 provides an overview of project design. Section 6 describes how the project is implemented. In section 7, we provide the performance and feedback from a domain expert as an evaluation. Section 8 draws a conclusion of this project, and section 9 discusses potential further work.

2 Background Research

In this section, a literature review is first introduced to present the most relevant works to this project. In the second part, previous systems in a similar project are introduced. The third part provides a detailed analysis of the data characteristics.

2.1 Literature Review

This section defines principles and techniques for data preprocessing and text data visualisation.
Text Visualisation Browser [Kucher, 2014] is an online tool which provides the most comprehensive summary of published text visualisation [Cao and Cui, 2016]. According to Text Visualisation Browser, from 1976 to 2017, there exist 400 published text visualisation papers in total, in which 396 publications are aimed to analyze text alignment. By searching 'Word', there shows 20 publications, and 'Translation' gets 16 results. Whereas when typing 'Frequency' and 'Weighting', each keyword gets one results. Also, keywords such as 'Machine learning, 'Data Mining', 'Natural Language Processing' got no publication collected. The results indicate that in text visualisation domain, most researchers focus on presenting alignment of texts. There are certain amounts of research focus on the topic such as 'word analysis' and 'translation', which is similar to this project. However, applying more specific techniques such as 'Natural Language Processing' haven't been applied in text visualisation widely.

Interactive Exploration of Versions across Multiple Documents

In the work of *Interactive Exploration of Versions across Multiple Documents*, [Jong et al., 2008] provide an interactive visualisation tool, MultiVersioner, to address the issues of comparing several versions of texts. The MultiVersioner enables users to search for items such as

words, phrases and lines, along with the analysis of the frequency patterns of these items. In addition, methods such as colour-coded highlighting and overview are also rendered in this tool. Figure 1 serves as an example of overview for many versions and documents in this software. In the overview, terms are denoted by blocks. If user mouses over a single block, a tooltip with the relevant sentence will be popped up.

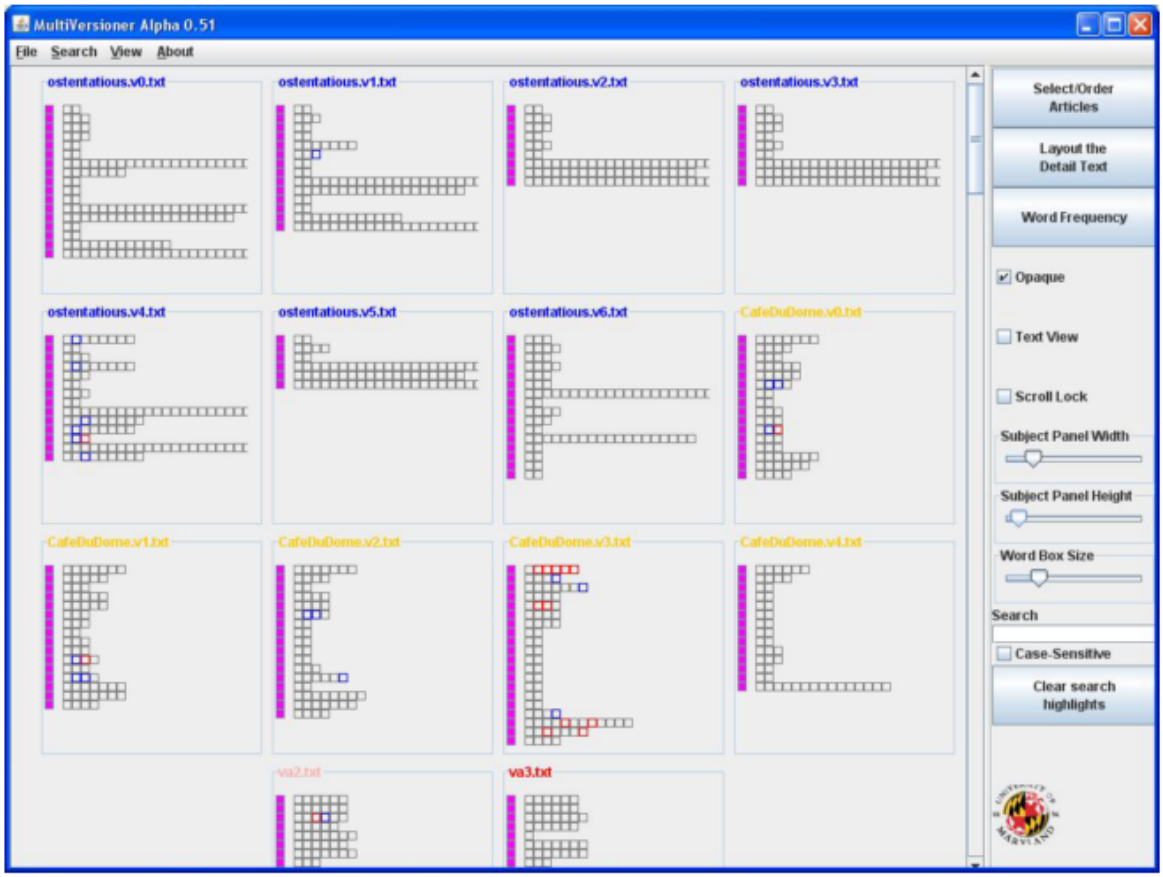


Figure 1: An overview of many versions and documents in MultiVersioner ([Jong et al., 2008]).

The work of MultiVersioner allows users to compare multiple documents. Meanwhile, it provides a helpful feature to search for entities such as words and lines. Moreover, it can be served as a tool to analyse the frequency patterns of the words. However, there are only limited features provided in this software. Some helpful functions such as alignments between versions, or version turning on and off need to be explored.

Interactive Visual Alignment of Medieval Text Versions

[Stefan and Wrisley, 2017] discussed novel methods to compare text versions in the work *Interactive Visual Alignment of Medieval Text Versions*. They provide a visual analytics system which enables computationally align complex textual differences such as orally inflected text. The data they deal with is a group of medieval poetries with complex text forms. Their works include three basic visualisations:

- **A visualisation of text alignment** This is a visualisation in which highly unstable text versions are identified and aligned. This feature is accomplished applying parameter-driven approach. Moreover, a visual analytic process is rendered which accept tweaking parameters for iterative improvement of the alignment.
- **Multi-level alignment visualisation** In this view, various visualisations are presented which enables users to analyse texts alignments on different hierarchy levels.
- **Meso Reading** This is a visualisation to interpret texts in parallel. Similarly, the connections are demonstrated among text versions. This feature is considered as a novel feature which provides an intermediate perspective to display complex variance in texts.

  Figure 2 is an interpretation of this project. The screenshot in [Stefan and Wrisley, 2017] displays different views of the distant reading, the meso reading, the close reading, and full-line matches. In part one, distant reading results of high similarity on words are displayed in the form of parallel lines. In addition, diagonal lines represent several repetitions. In part two, meso reading results are shown in the verse line 'Qui me dist que li ange sont'. In part three, a close reading feature is explored to show false positive alignment while part four are the matches for numerous full-line text.

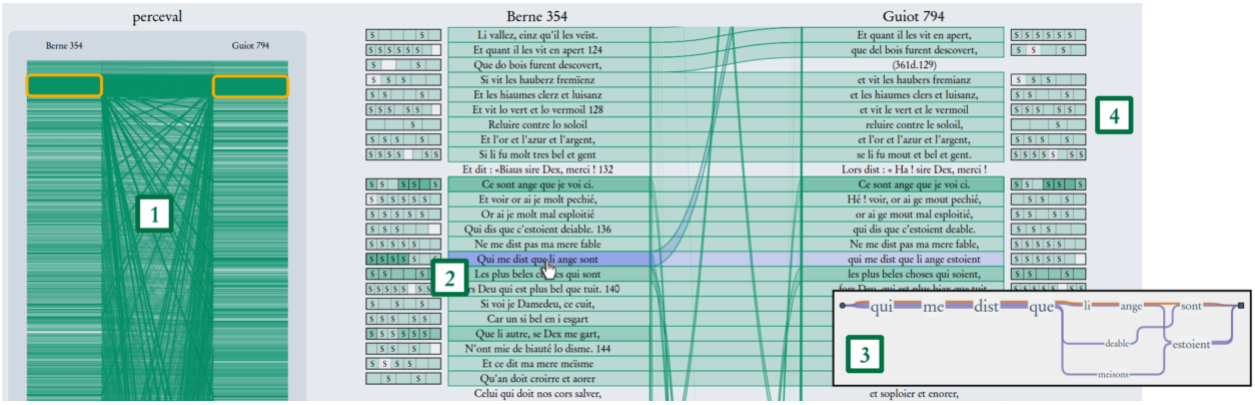


Figure 2: (1) distant reading (2) meso reading (3) close reading (4) full-line matches ([Stefan and Wrisley, 2017]).

However, this tool is more suitable for French context, which limited the rage of data text can be processed applying methods in this work.

Visualizations Translation Variation of Shakespeare's *Othello*: A Survey of Text Visualisation and Analysis Tools

In their work *Visualizations Translation Variation of Shakespeare's* Othello*: A Survey of Text Visualisation and Analysis Tools*, [Geng et al., 2011] developed a visualisation system which can be used to view and analyse variations between translation and based text. The data of this project is a collection of German translations of Shakespeare's *Othello*. In this project, several techniques are applied to get an interactive visualisation system. These techniques include parallel coordinate, Treemap, and DOI-tree. The tools which they developed provides features that enable users to brush words so that these words can be displayed in a parallel tag cloud. Figure 3 is a screen shot which illustrates the outcome of applying the TagCrowd feature into one of *Othello* translation version. In this view, the stop words are identified and removed manually from the original text.

Figure 3: The TagCrow ([Tag,]) visualisation of a passage from *Othello*.

ShakerVis: Visual Analysis of Segment Variation of German Translations of Shakespeare's *Othello*

ShakerVis [Geng et al., 2015] is a special visualisation tool which is designed to provide an interactive visualisation system to display version variations. In this system, [Geng et al., 2015] applies following visualisation techniques:

- **Parallel Coordinate View** provides outcomes by using Eddy value. The description of Eddy value can be seen in Previous System chapter.
- **Scatter Plot View** presents an average similarity value for each translation across multiple segments.
- **Term-document Frequency Heat Map** renders a way to analyse differences between pairs of versions in details, including a measurement of character-string similarities.

2.2 Previous Systems

The Version Variation Visualization (VVV) project was introduced by Dr Tom Cheesman from Modern Language Centre at Swansea University. It aims to create interactive data visualisation system to build cross-cultural exploration networks. The VVV project focus on developing digital tools which can help to compare and analyze different versions of translation [Cheesman et al., 2012]. So far, the tools developed in the project is Ebla, Prism and ShakerVis. Ebla served as the corpus, is a software to stock the text data and detailed information about them. Prism provides the interface for separating texts into segments and processing the segments as alignment. Based on the idea of this two software, ShakerVis provides an interactive interface for visualizing the information of the translation versions [Geng et al., 2015].
There are three types of data visualisation in this project: Time-Map, Alignment Maps, Parallel view and Eddy and Viv view.

Time-Map

Figure 4 supplies a screen shot of Time Map, which shows the location of the authors and the year of translation versions published. From this view, we can tell that some particular places such as Berlin and Dresden in which more authors were born this places and more translaitons versions were published from here. This can be deemed as that these two cities are popular places with large scale of city.

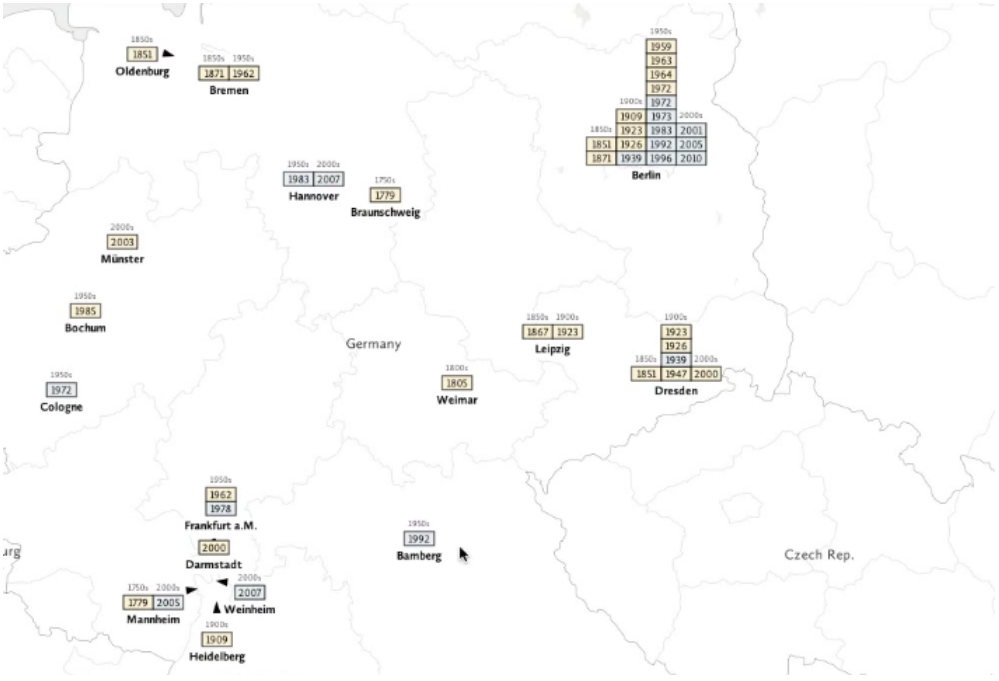


Figure 4: A demo screen shot for Time-Map ([Cheesman et al., 2012]).

Alignment Map

Figure 5 exhibits a parallel visualisation which provides an alignment from the segments in th base texts to the translation versions. The yellow parts highlighted the whole segment selected while the blue line serves as the alignments. By comparing these texts, one can tell the general differences between the base text and translations. For example, if one segment of certain translation is longer than that of the base text, it is possible that a particular expression in German is appeared.

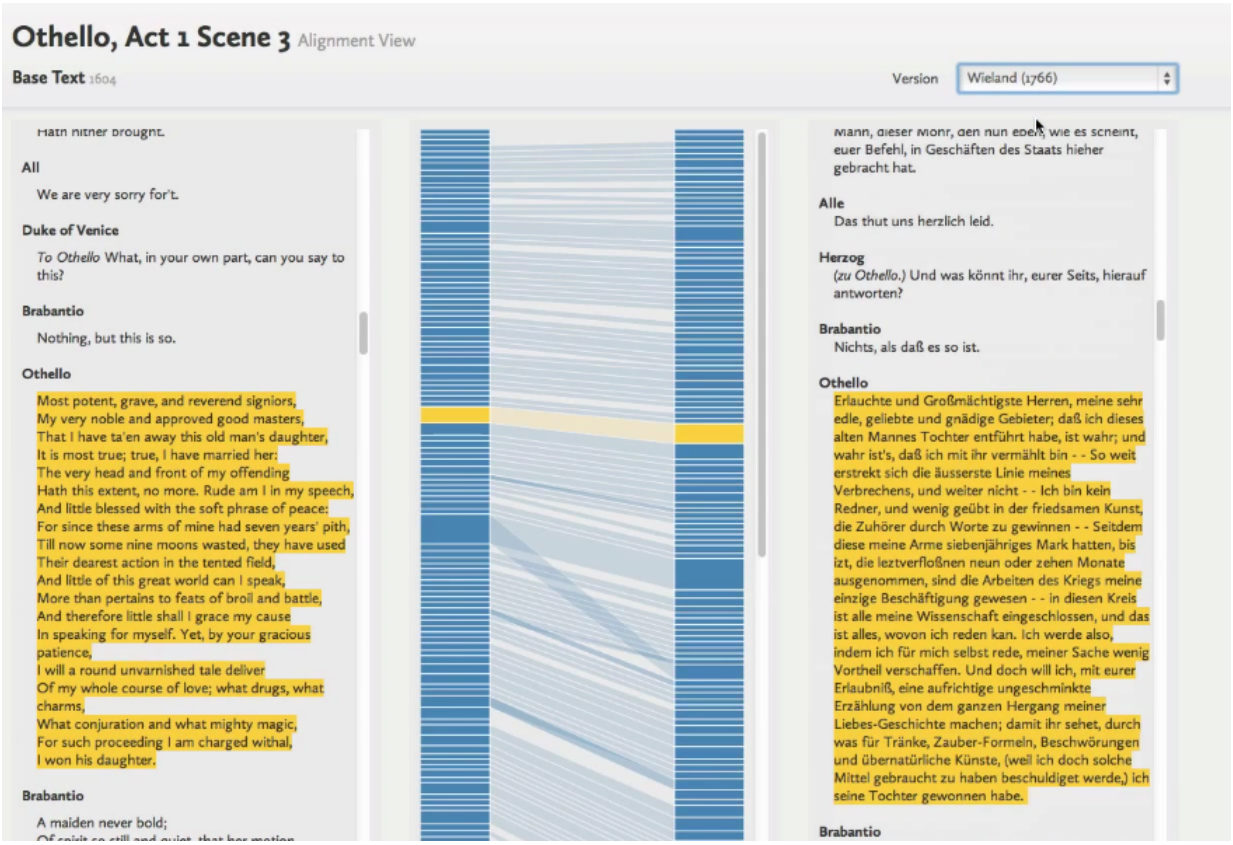


Figure 5: A screen shot of the demo of Alignment Maps ([Cheesman et al., 2012]).

Parallel View

Parallel View provides an explicit view between the base text and selected version. Figure 6 shows a straightforward view of base text and selected translations. In this visualisation, segments are more explicit to find.

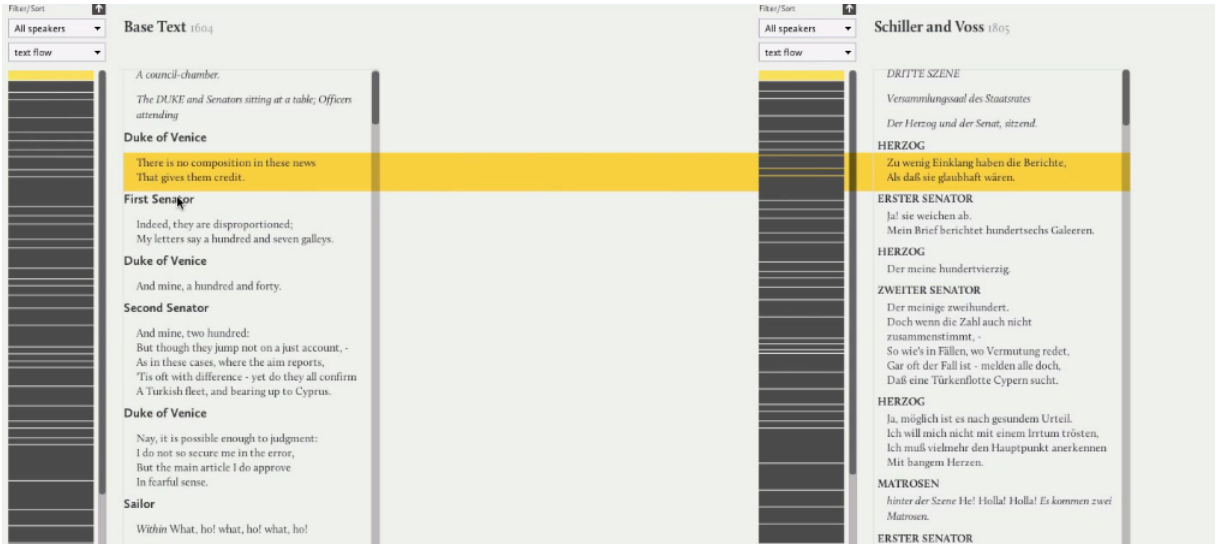


Figure 6: A screen shot for the demo of Parallel View ([Cheesman et al., 2012]).

Eddy and Viv View

Eddy and Vis view enable researchers to understand more details of vocabulary. Figure 7 demonstrates Eddy and Viv view, which provides more information of the translation comparison. From the sort bar, we can tell that there are four types can be visualized. Eddy value shows the variation of words used in the segment. Relatively, Viv value provides the changes or rivalries for some segments in translation. If we choose version name, segment length or reference date as the order of sorting, there will be other information on translation variations. Also, there are back-translation based on machine translation provided, which is another powerful function for comparing the text data.

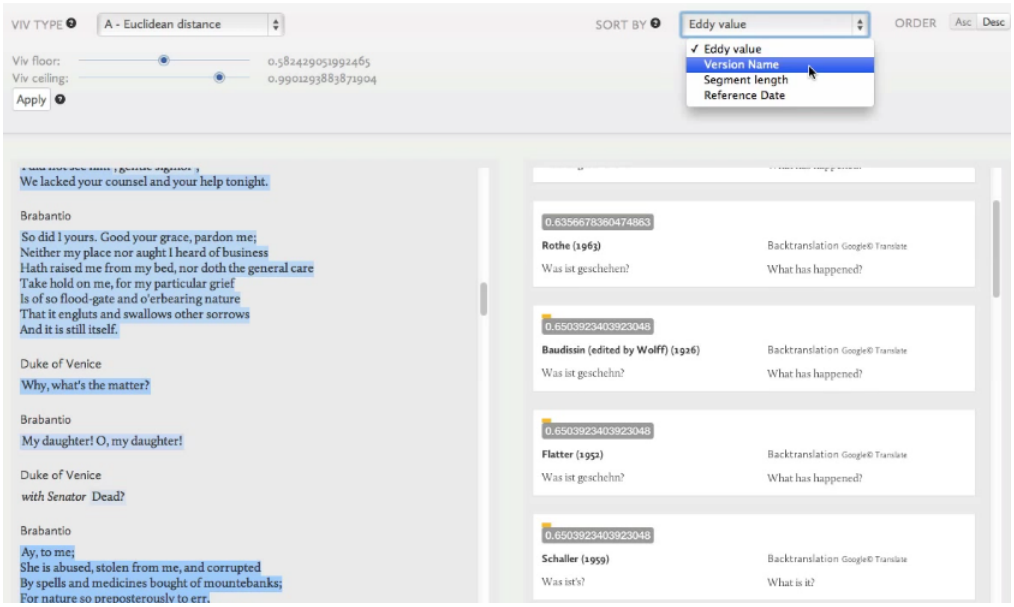


Figure 7: A screen shot of Eddy and Vis view ([Cheesman et al., 2012]).

3 Project Specification

This part is the specification of the project which includes the features specification and technology choices to the software. The user features the system will also be stated. The project specification discussed in the initial document is modified and updated here as a section of the final dissertation.

3.1 Data Characteristics

Data is a major part of all visualisation, which along with user experience play an important role as ”driving factor” with respect to the choice and attributes of the visualization method [Laramee, a]. In this chapter, the data relevant to this project is analysed, including the type, size, format, and characteristics of data. Also, a description of data preprocessing will be discussed.
The data sets used in this project come from a collection of 57 different German translations of *Othello*, which is contributed by Dr. Tom Cheesman, from College of Arts and Humanities at Swansea University, working on a project in [Tom et al., 2012]. To develop analytic tools and probe the translations in this corpus, the team digitalized 32 translation versions, with the formats being normalized, texts being segmented, speech by speech and line by line. The content of these 32 texts corresponds to Act1, Scene 3 of the English version of *Othello* (1604) play as the base text. Based on this corpus, we are given 15 text files of German translation versions by Dr. Tom Cheesman for this project. And together with the base text in English, these 16 text sets are read and processed when implementing the project.All these files are encoded as UTF-8 when converting from .docx format to .txt format. The number of words in each document are different according to the genres of text data (327 words at maximum, and 214 words at minimum). Figure 8 is a screen shot of the text data used in the project. All data has been segmented and cleaned

```
011 Gundolf 1920 (1909); no copyright

DES.
Edler Vater,
Ich sehe hier eine getrennte Pflicht:
Euch danke ich mein Leben und Erziehung,
Und Leben wie Erziehung lehren mich
Euch ehren. Ihr seid Herr der Pflicht, ich bin
Insoweit eure Tochter. Doch hier ist mein Gatte,
Und soviel Pflicht als meine Mutter euch
Erfüllt, da sie euch ihrem Vater vorzog,
Soviel begehr ich zugestehn zu dürfen
Dem Mohren, meinem Herrn.

BRA.
Gott mit euch! Ich bin fertig.
Beliebts eur Gnaden, zu den Staatsgeschäften! ...
Besser ein Kind annehmen als eins zeugen ...
Komm hierher, Mohr
Hier gebe ich dir das von ganzem Herzen
Was ich, hättst dus nicht schon, von ganzem Herzen
Dir vorenthielte ... Eurethalben, Schatz,
Bin herzlich froh kein zweites Kind zu haben:
Mich würde deine Flucht Gewalttat lehren,
Ich legte ihm Klötze an ... Herr, ich bin fertig.

DOGE.
So red ich denn wie ihr und fäll ein Urteil,
Das, Tritt und Staffel, diesen Liebenden
In eure Gunst verhelfe.

BRA.
Ich bitt euch untertänig, gehn wir an die Staatsgeschäfte.

```

Figure 8: A screen shot of text data used in this project.

The text file is commonly used to store plain texts data. It is a simple text file format which can be worked with many programming languages, including Java. Choosing .txt file as the data set is owing to following reasons:

- The aim of this project focuses on word processing, which requires computers to read text literally, without applying complicated data processing techniques.
- Since the text data sets in the corpus are stored in .docx format which is difficult to read directly from Java, it is easier and safer to convert the .docx format into .txt format.
- There exist methods in Java.io, a Java API, used to read .txt data directly from files.
- Apart from the basic and simple information (publication year and author) of each version, there is no need to obtain more information from the text. Additionally, because the data set in each

version is not large, the computer can calculate the essential features of the data, in a short time, every time the program is ran.

3.2 Feature Specification

This project is aimed at developing an interactive visualisation for a group of different text documents. The result of this visualisation should assist users in identifying and exploring the variations between these translation versions. The software has following features:

- Provide an interactive visualisation system.
- Develop a user interface serves as a tool for users to select options.
- Read and store data from .txt files.
- Provide a parallel visualisation for comparing terms in different translation versions.
- Generate concordance view with frequency bars
- Add author and publish year as the title of each concordance.
- Provide a visualisation with scroll bars.
- Connect same words in each concordance applying coloured edge.
- Provide user option for scaling the visualisation.
- Scale the size of window.
- Generate a colour mapping view, and the colour represents the frequency of words.
- Render a user option for turning translations on and off.
- Create an English-German word translation index.

- Add user option: highlight the bar and connection when clicking single bar.
- Provide user option: highlight bars with same frequency when clicking a block in colour legend.
- Generate a Lemma and Frequency visualisation.
- Generate a Tf-Idf Visualisation.
- Generate a Lemma and Tf-Idf Visualisation.

3.3 Technology Choices

According to the project specification and required features presented previously, the demonstration of technology choices is made in the following chapter.

3.3.1 Programming language

For the implementation of the software in this project, Java programming language is selected to develop the software. Java has known that it is an object-oriented language and class-based [Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad; Buckley, 2014]. It is also simple enough to understand fast. With years of upgrading and improvement, it has been growing into a mature programming language. This also means using Java to develop software will have fewer mistakes and bugs when programming. There are also active communities on the internet, in which lots of people share useful ideas and resources of Java. In addition, due to the limitation of background which is not Computer Science, the author is more familiar with Java programming language.

3.3.2 Java Library

Java Swing Library

The Java Swing Library is the tool we used in this project to generate GUI of the software. This is a free, cross-platform resource which is appropriate for using Java in implementing this project.

Stanford NLP Library

The Stanford NLP Library is attempted during we generate the lemma for English version text [Sta,]. This is a free and open source for Natural Language Processing. However, because this library has not provided German lemmatisation function, we adopted other solutions in this project.

3.3.3 Other Techniques

TreeTagger

TreeTagger, developed by Helmut Schmid at the Institute for Computational Linguistics of the University of Stuttgart [Tre,], is a tool for annotating text data and lemma information. It has been used to tag many languages including German.

Github

For source code backup requirement, the Github is adopted. The main software we used most is the GitKraken (See Figure 9), which provides an interactive user interface to commit project. As a version control tool, the Github helps in organizing the development process of our software and in keeping an updated version of the software.

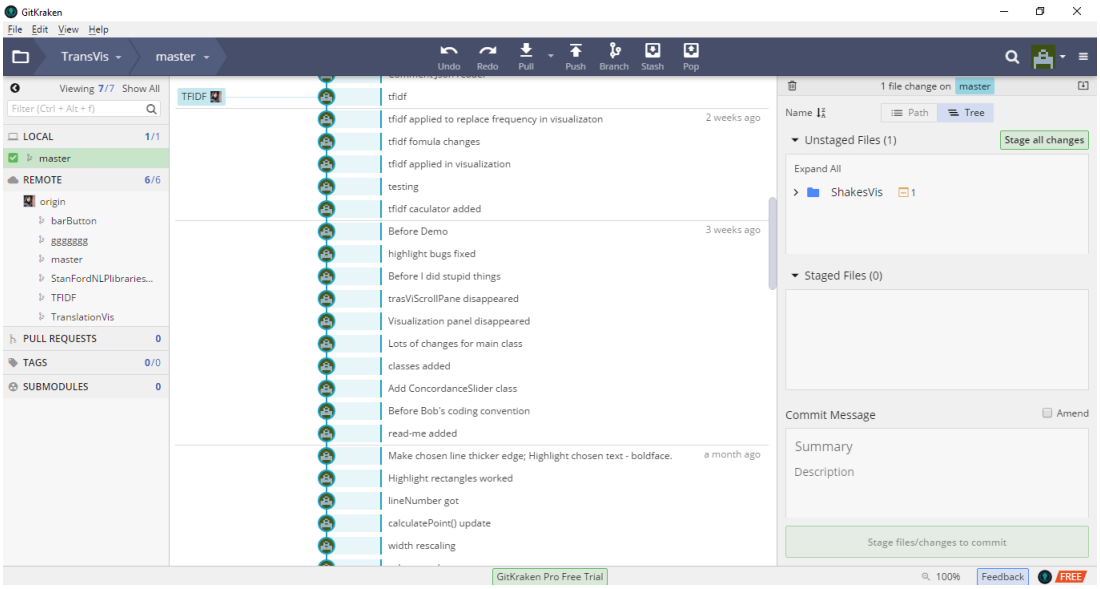


Figure 9: A screenshot for the user interface of GitKraken.

Dropbox

The Dropbox is another backup software which can be used to store data. We adopt this software to store our source data in the case that equipment is broken, or the website of Github is collapsed.

Eclipse and Visual Studio Code

Eclipse and Visual Studio Code are tools we used in this project for Java programming. The Eclipse is the main tool to program in Java, while the Visual Studio Code serves as a backup software in the case that Eclipse is collapsed.

Notepad++

The Notepad++ is a free and useful tool for source code editing. It also supports editing files in many kinds of format. In this project, we adopt Notepad++ to encode data during Data Processing phase.

4 Project Plan and Time Management

4.1 Development Approach

Traditionally, Waterfall Model is used as the guiding methodology for many projects. It uses linear flow to show the progress of the project and allow people to understand easily the further steps after completing the previous step. It is suitable for sequential design, which means it may be impossible for developers to go back to steps if they found some problems at the end. The progress of the Waterfall Model is, according to [Adenowo and Adenowo, 2013], include five phases: Requirement analysis, design, implementation, testing, and operation and maintenance.(See Figure 10)

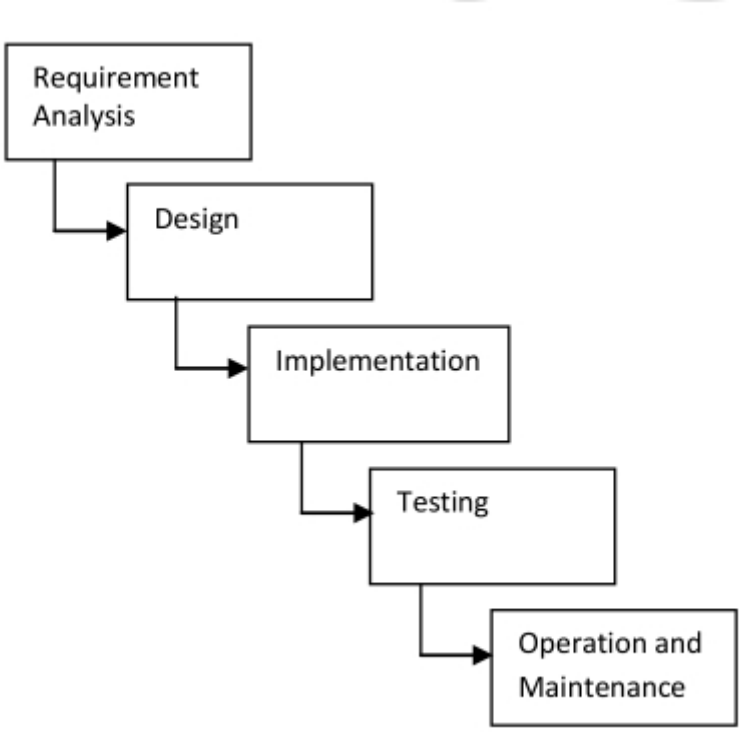


Figure 10: Five phases of the waterfall Module ([Adenowo and Adenowo, 2013]).

However, when projects run out of time, the testing phase will be cut, which may lead to poor quality outcomes. In addition, in the last step of implementation, developers may be unaware of steps they have taken;

hence, it is impossible for developers to change the code until the last phase.

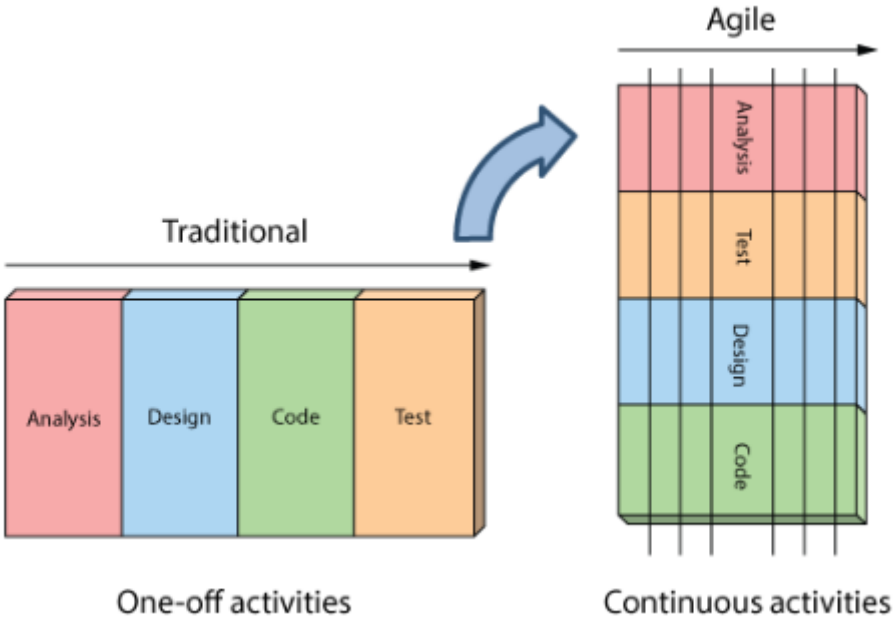


Figure 11: A comparison between Waterfall methodology and Agile methodology ([Agi,]).

Unlike with the Waterfall methodology which separates the whole project into several phases and implements it step by step, the Agile methodology separates the project into several tasks and every task is implemented in several phases. By doing this, it is changeable for developers when they find mistakes; therefore the quality and visibility issues of the Waterfall methodology are solved. For this reason, we adopt Agile as our guiding methodology when implementing this project.

4.2 Project Timetable

This section indicates time management for the project. This project can be separated into five phases [Laramee, a] as follow:

- 1. Requirements Specification; Data Preprocessing; Project Presentation; Exploring existing tools; Project Specification; complicated data processing techniques.

- 2. Software Design; Candidate Classes and Responsibilities; Candidate Hierarchy; Collaboration and Subsystems;
- 3. Implementation; Software Development; GUI;
- 4. Debugging and Testing;
- 5. Documentation;

Figure 12 indicates the Gantt chart of the project timeline. This project is initiated on the 17th February 2016, and the final deadline is the 15th December 2017. In every phase, there are several tasks to be done. Most of the tasks in phase one were completed by Dr. Tom Cheeseman before the data preprocessing took place; the second phase finished before July 2017, which allowed for more time to be used in the implementation phase. Priority needs to be given to software implementation, which will be executed according to the designs done by previous work. After the implementation, simple Graphic User Interface (GUI) framework will be put into effect. From the middle of November 2017, the project commenced its debugging and testing phase. Finally, a report and Doxygen was done in December 2017.

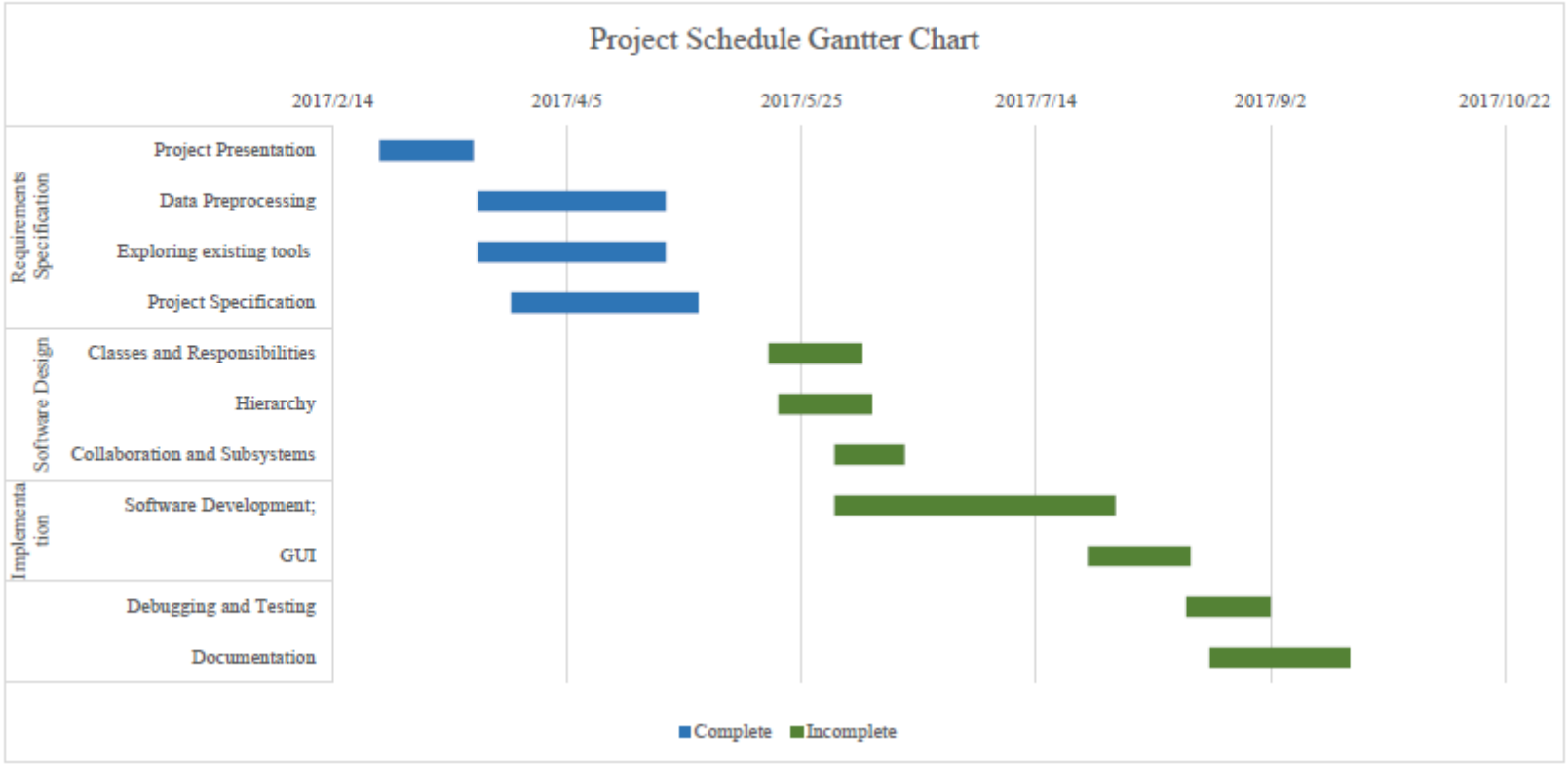


Figure 12: Gantt chart for project timeline.

4.3 Risk Analysis

This section explains the potential risks this project. Figure 13 illustrates the analysis of these risks: The first risk is the limited time. The impact of limited time has been considered as high because the project requires a high level of attention to detail which is time-consuming. Regular meetings with the supervisor were important to consult on the difficulties encountered during the process. The second risk identified is the personal illness of the author. It is a low possibility risk with medium impact for the project. To deal with this case, keeping a good healthy is important to author herself. In addition, there is welfare service department on campus and the author has the international student's insurance. Equipment failure is identified as the third risk. It is classified as medium in terms of both possibility and impacts. Yet the resources of Swansea University are available 24 hours per day, which will reduce the impact of this risk. In addition, using the application Github for regular backup is essential. Data loss is considered as the fourth risk. The possibility of this happening is low but will cause high impact on the project. To avoid this situation, Dropbox is essential for backup. The last risk the necessities having a new supervisor, due to unforeseen circumstances. Swansea University procedure exists to support students by providing a second supervisor.

Risk	Probability	Impact	Precautions
Lack of knowledge	Medium	High	Regular meetings with supervisor for consultation
Delay of completion of the project	Medium	High	Application of postponing submission of the project
Personal illness	Low	Medium	Doctors available on university campus
Equipment failure	Medium	Medium	Computers available on campus
Data loss	Low	High	Using Dropbox for regular backups
Lack of project management skills to implement the project	Medium	High	Adopting Agile software development approach

Figure 13: A screen shot of Risk Analysis Table ([Liu,]).

5 Project Design

Project design is the first step in software development. Due to the programming language used to implement the project being Java, the design will follow object-oriented principles. Classes and their responsibilities will be provided in following sections.

5.1 Data Reading

Data preprocessing is the first stage in doing this project. As discussed in Chapter 2, the data source is a collection of 16 .txt files. In the following part, all classes in Data preprocessing phase are introduced, with diagrams to illustrate the concept of the design.

DataReader Class

The DataReader class is one of the base classes that is designed for reading and processing data from all 16 files. The data is then passed to other objects to be stored and used. There are five main functions as follows:

- Read data from .txt files;
- Calculate term frequencies, point locations, colour values;
- Pass the calculated values to other classes and store them;
- Accept Tf-Idf values from other classes;
- Generate a List of Lists to store all information needed and pass that to visualisation parts;

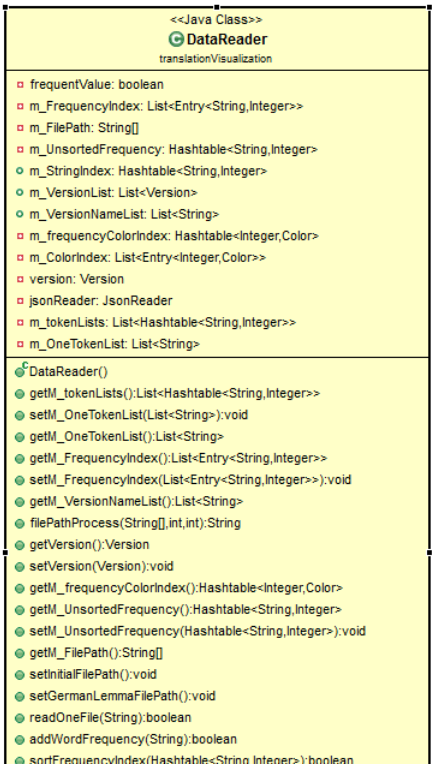


Figure 14: DataReader Class Diagram.

The structure of the DataReader class is presented in Figure 14.

Item Class

The Item class is an object class used to store the basic information of terms: the string of word, frequency, rectangle, Tf-Idf value, location, font, translation sets, and lemma. All these values are generated from the DataReader class. Then these values are stored into lists of Item objects as a column. When the visualization is being generated, these values will be used directly. The data can also be modified from accessor methods when interacting with software. The class diagram is shown in Figure 15.

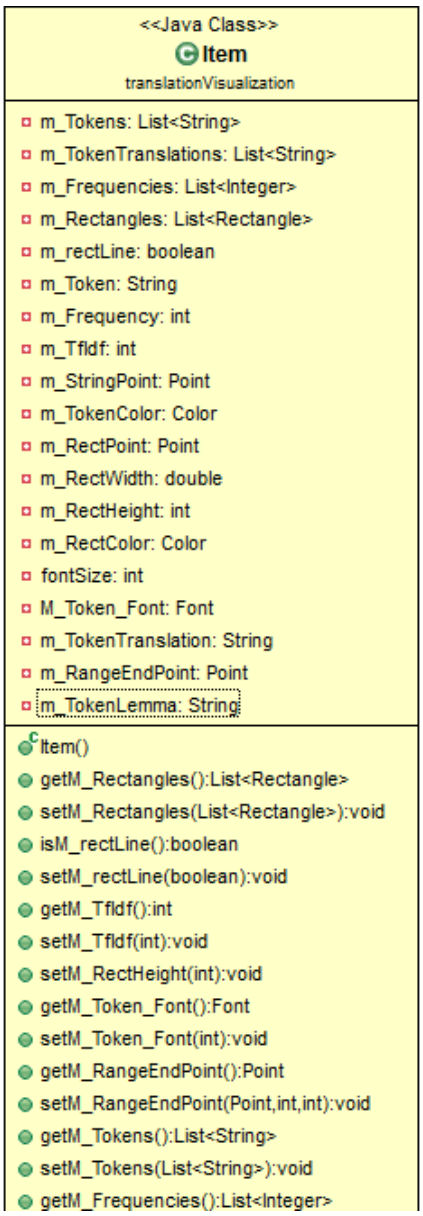


Figure 15: Item Class Diagram.

Version Class

The Version class is another object class used to store information related to each version of the text, such as author, publication year, title location. It also includes a list of Item objects for the concordance of this translation version. After all 16 texts have been read and processed, there will be a list of Version objects generated and the data will be displayed and modified on the visualization panels. The class diagram of Version class is shown in Figure 16.

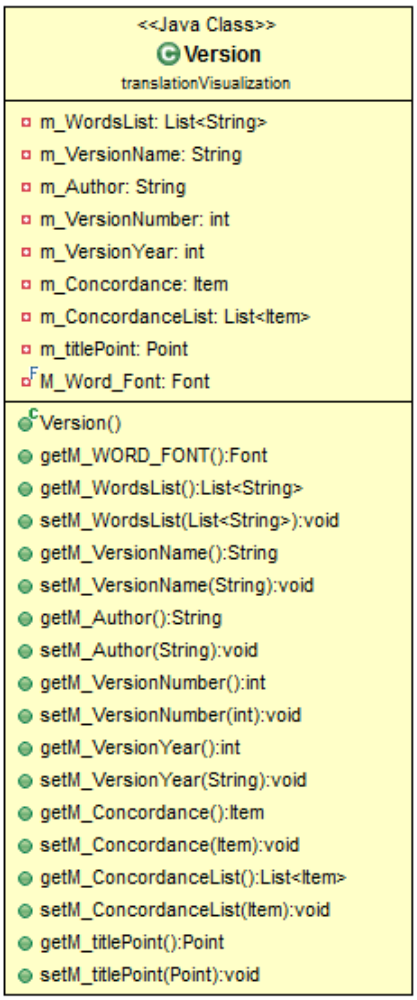


Figure 16: Version Class Diagram.

TFIDFCalculator Class

The TFIDFCalculator class is designed to process the Tf-Idf value for each term. The Tf- Idf calculation comes after the original data is read and processed so that the term frequency can be used directly in this class. This class includes 3 stages:

- Accept frequency data and word sets from DataReader class;
- Calculate Idf value using word sets;
- Calculate Tf-Idf value and pass them back to DataReader class;

The algorithm of Tf-Idf value will be introduced in the Implementation

chapter. The class diagram of TFIDFCalculator is presented in Figure 17.

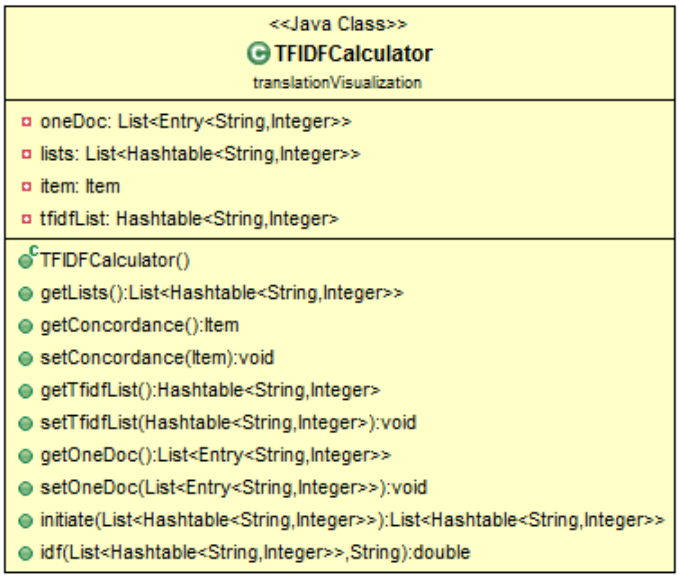


Figure 17: Tf-IdfCalculator Class Diagram.

LemmaProcess Class

The LemmaProcess class is designed to generate lemma for each term. As shown in Figure 18, this class includes three main steps:

- Read data from German lemma corpus;
- Search lemma for each term which is passed from DataReader class;
- Store the lemma for each term into a new .txt file;

Detailed information of the lemma processing part will be stated in Implementation part.

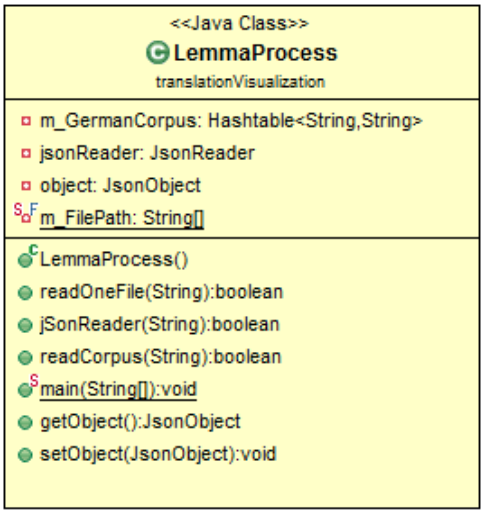


Figure 18: LemmaProcess Class Diagram.

5.2 Tasks to Support Visualisation Mantra

TranslationVisualisation Class

The translation generation stage comes after data reading and processing (See Figure 19). The Transla- tionVisualisation class is designed for accepting all data processed from the data reading phase and generating the visualization using the software. This includes following stages:

- Accept data from DataReader class;
- Initialize all GUI components;
- Pass the data to GUI components;
- Set GUI components and add them to accordingly visualisation panels and frames;

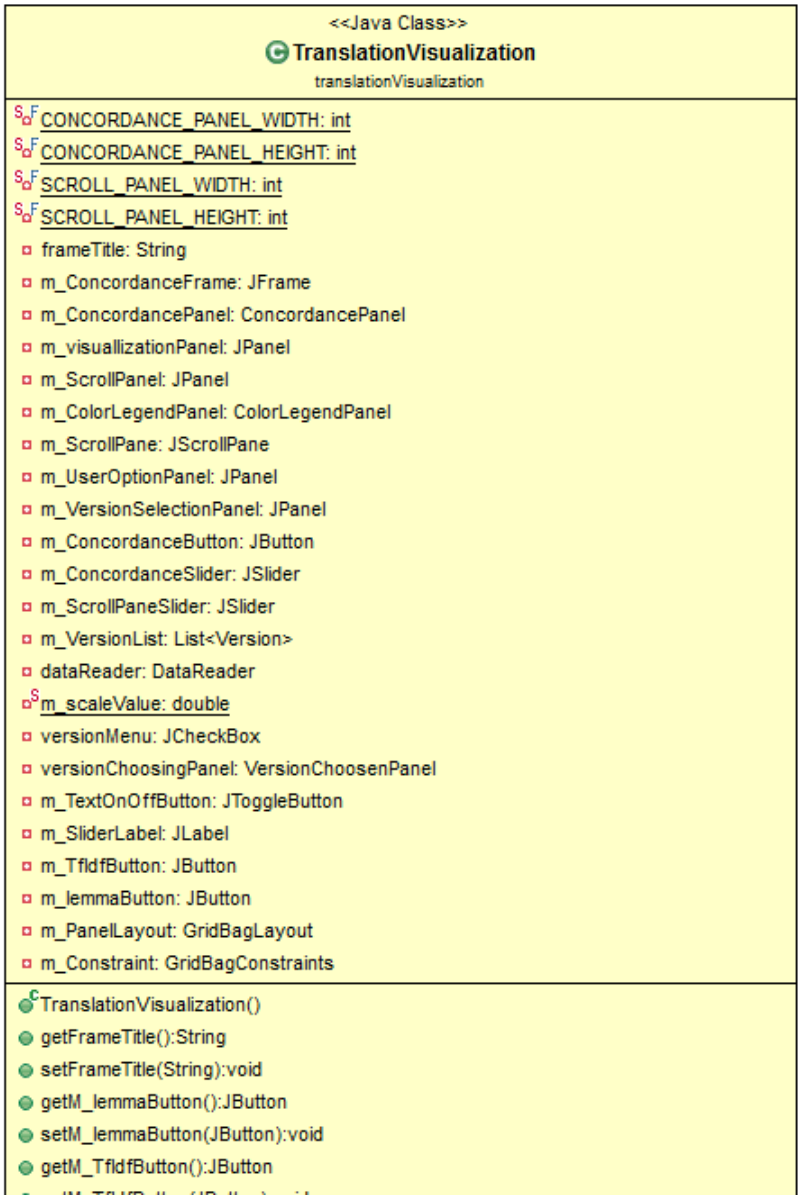


Figure 19: TranslationVisualisation Class Diagram.

5.3 GUI

Most of the GUI classes in the software are inherited from Java AWT libraries.

ConcordancePanel Class

The ConcordancePanel is the main visualization panel in the software. It is inherited the JPanel class which belongs to Java Swing li-

brary. It is designed to render a canvas drawing of all parallel visualization of concordances. Data is passed from the visualization part and used to display visualization within the ConcordancePanel. There is also a Mouse Click Listener in this class used to listen to the rectangle area clicking event. Several functions of this class are as follow:

- Accept data from DataReader class;
- Initialize JPanel;
- Draw strings, rectangles, lines on the canvass;
- Pass events data from event listeners;
- Recalculate data;
- Repaint the graphic;

The class card of ConcordancePanel class is shown in Figure 20.

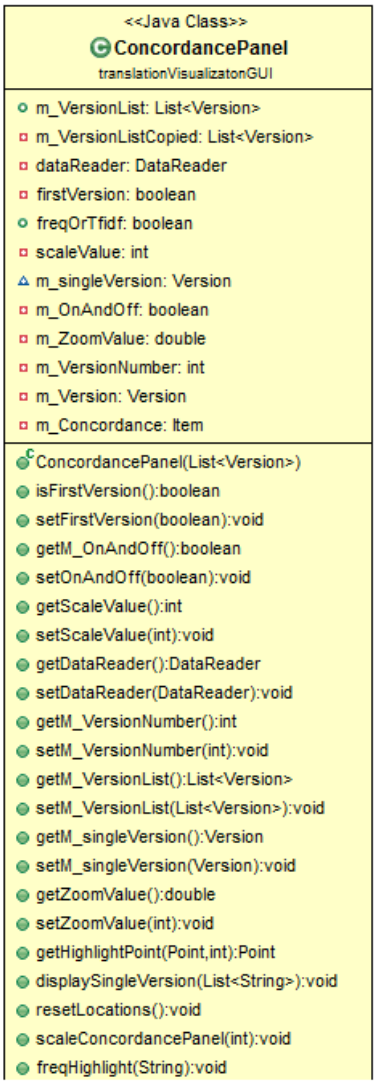


Figure 20: ConcordancePanel Class Diagram.

ColorLegendPanel Class

The ColourLegendPanel class inherits data from JPanel. This class renders a colour map, where each colour owns an event listener. The data passed in this class is term frequencies, or Tf- Idf values, depending on user preferences. An event listener is added to each colour block to listen which block is selected. Then the selected data will be passed to TranslationVisualisation class. The class diagram is displayed in Figure 21.

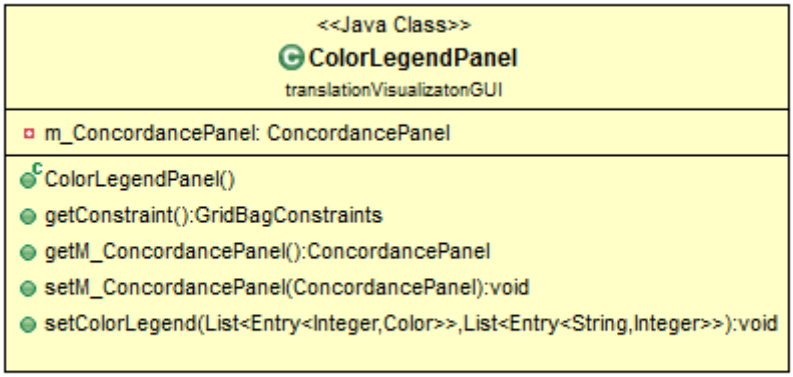


Figure 21: ColorLegendPanel Class Diagram.

VersionChoosenPanel Class

The VersionChoosenPanel class inherits from JPanel. There are several steps in this class:

- Accept data from DataReader class;
- Initialize JPanel;
- Display version titles in JCheckBox as a version selector;
- Pass events data to ConcordancePanel class;
- Display which version is selected;

The structure of this class is shown in the diagram of Figure 22.

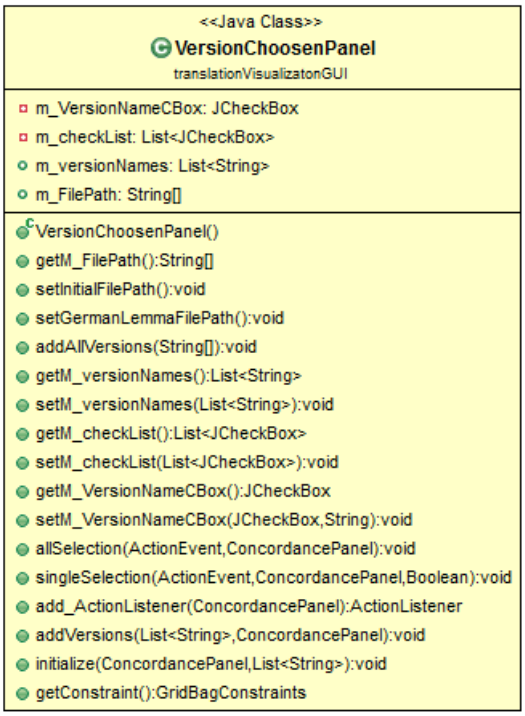


Figure 22: The VersionChoosePanel Class Diagram.

6 Implementation

In this section we describe how the project is implemented in detail. The subsystems executed are data reading, visualisation rendering and the user options. Screen captures of the GUI are added to illustrate further information.

6.1 Data Processing

The main concept of data processing is to read text data from .txt files, calculate values need, and store them into Java Arraylist. At this stage, the greatest challenge is to keep the data being accessible and can be changed, as the new data may be written over the old due to events in other class. Hence, after the original data is read and stored, it is retrieved and modified through mutator methods [Laramee, 2009]. In addtion to the flexibility, another difficulty at this stage is generating values for each term and store them appropriately. As discussed in the Project Features section, the aim of the software we designed is to present information about terms and provide an concordance view for each version. In this project, we calculate and sort frequencies of terms; compute colour values; computed locations of strings; instantiate Rectangle objects to represent data; create arrays to store translations.
Java.io, which enables for system input and output through data streams, is used in this project. It serves as a data buffer and reader in this project. The FileReader class, which extends the InputStreamReader class, can be used to read character files which by default are assumed to be an appropriate size. Since the volume of data in each document is not large, we instantiate a FileReader (object) to access each text file. The other data reading class adopted is the BufferedReader class. It is used to read text data from a character-input stream, and buffer the data to provide efficient reading of strings, arrays and lines [Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad; Buckley, 2014]. Java.util is another package imported in the DataReader class. To store and

access data, the ArrayList, Hashtable, and Map classes from this package are used. In addition, classes such as JsonObject, JsonReader, and JsonArray in Javax.json package are used to read data from a Json file.
The main class that is responsible for reading the original data file is the DataReader class. This class analyses .txt files and generates a list of Version objects to parse all information needed in the software. Each Version object stores information of the concordance. For a more detailed description of the Version class and Item class, please see the Design section.

6.2 Generating Concordances

Concordances are the most basic visualisation in this project. They are designed to display the information of terms, and to help in comparing the terms between different translation versions.As shown in the Figure 23, the concordance visualisation involves several parts:

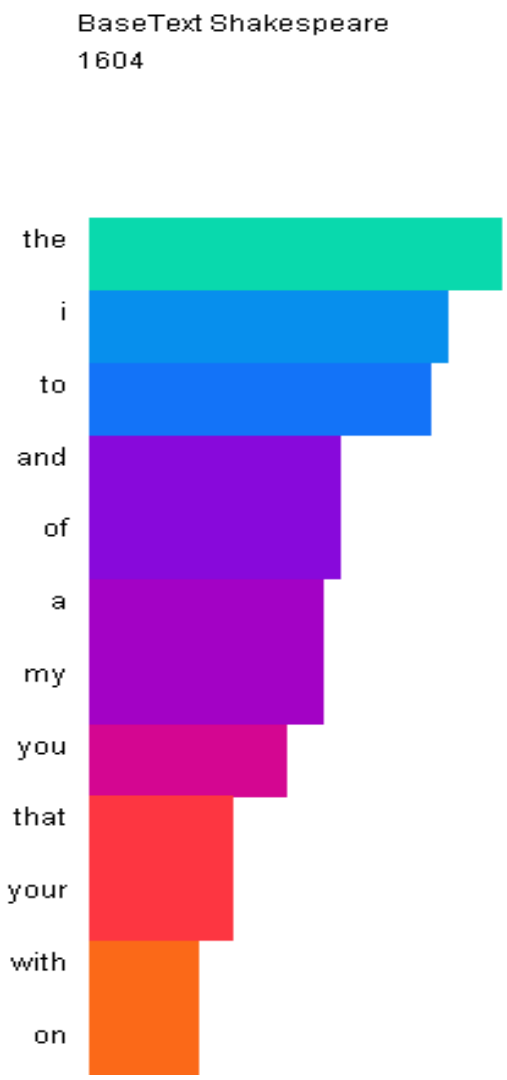


Figure 23: The Screen shot of one concordance in the visualisation

- **String** is drawn to display the term, frequency, version author, publication year;
- **Rectangle** is used to present the frequency. As the values are sorted in data processing phase, the width of rectangles are set according to these sorted values.
- **Colour** is used to present differences on frequency. Each colour represents a number of frequency, so there will be same colours in different terms.

The process in generating the concordance visualisation goes through the following steps:

- Obtain the string of each term from text source. This step is done in the DataReader class. Detailed illustration seeing Data Reading implementation section.
- Calculate the number of times, namely term frequency, of each term occurred in the text (See Data Reading implementation section).
- Calculate the rectangle width for each term using the frequency of term. The equation of the rectangle width calculating is show in Equation (1):

$$rectWidth = wordFrequency * unit * scaleValue \quad (1)$$

Where unit is the width of each segment since the rectangle is composed of a number of segments. WordFrequency is the value deciding how many segments compose the rectangle, while scaleValue is the percentage value used to scale the rectangle, range from 10% to 200%.

- Calculate the location of the string and rectangle.The location, or point, is the start drawing point for the string and rectangle. It combined with two point value: point.X, and point.Y. The Equation (2), (3) illustrate how we calculate these points in the software:

$$point.x = versionNumber * versionDistance * scaleValue \quad (2)$$

$$point.y = lineNumber * lineDistance * scaleValue \quad (3)$$

Where versionNumber represents order number of the version. versionDistance performs the distance between two neighbour

versions. In addition, a scale value need to be multiplied so that the location of string and rectangle changes according to user preference. Similarly, the lineNumber is order number of the term while lineDistance represents the distance between two terms.

- Calculate the value of colour. According to [Jbu,], we use the equation as shown in Equation (4):

$$color = Math.sin(colorFrequency * wordFrequency + phase) * amplitude + center \quad (4)$$

  Where colorFrequency is a constant that controls how fast the wave oscillates. The wordFrequency is variable used to display different colour according to word frequency. The phase is applied to change the alignment of the green or blue sine waves. The amplitude controls how high (or low) the wave goes. The center controls the center position of the wave.

- Paint the strings, blocks, and colours by invoking drawing methods in Graphic class.

6.3 Parallel View of Concordances

Following the generation of the Concordance visualisation, a parallel view of all concordances is created, as shown in Figure 24. During this stage, lines are drawn to highlight identical terms. The comparison stage is done in the ConcordancePanel class.

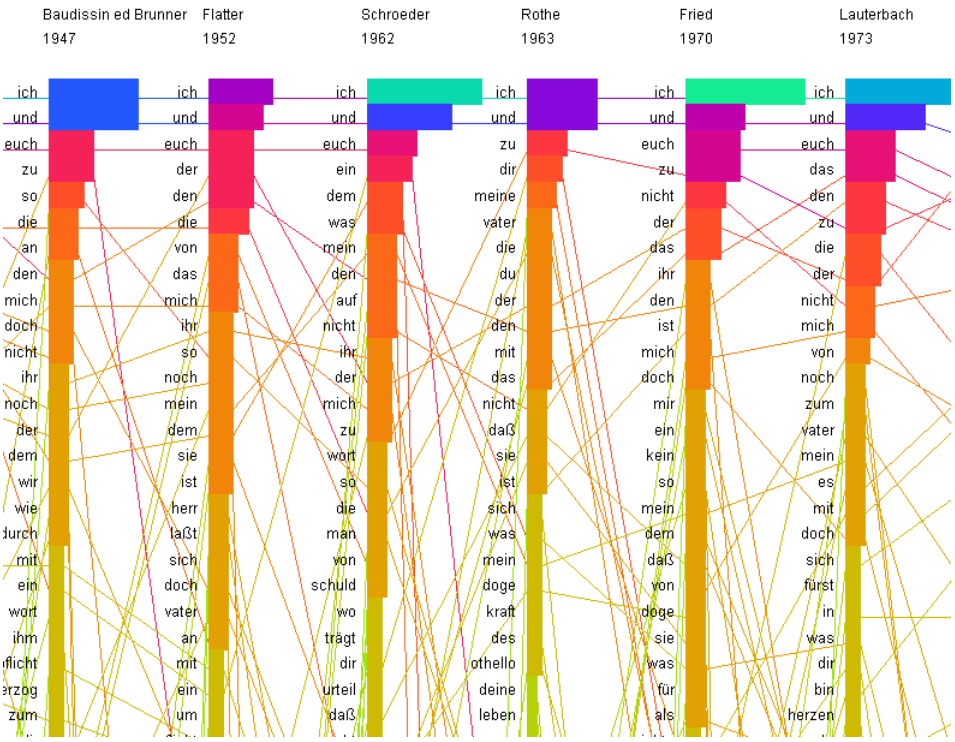


Figure 24: Parallel view of concordances.

However, after this parallel visualisation is generated, an obvious problem appears, there is not enough space for all 16 concordances to be shown simultaneously. Therefore the solution is either to scale the panel, or to select several versions for showing at one time. We have included both solutions, which are detailed in the following section.

6.4 Zooming

Zooming in and out is a basic feature in the software which is designed to provide two zooming options: one is for scaling the content of the visualisation, the other is for scaling the frame. In addition to these two scaling options, there are also scroll bars used to scroll the visualisation panel.
To implement these features, several steps as followed are gone through:

- Generate the JSlider objects. This is carried out in the TranslationVisualization class. Figure 25 displays the JSlider applied in the software.

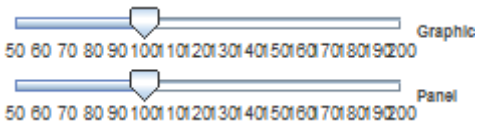


Figure 25: The screen shot of the JSliders.

- Obtain scale values from JSlider object and pass them to DataReader class.
- Recalculate the data by invoking the calculating methods such as calculatePoint() and setRectWidth().
- Update the List¡Version¿ object.
- Repaint graphics.

During this process, the most difficult part is to recalculate all values of graphics: points, widths and heights for rectangles, and the distances between versions. To overcome this dilemma, two solutions are attempted: At the first phase, scale() method in Graphics2D class is invoked. By applying this method, computer will calculate and repaint all the graphics using scale parameters passed in. However, when the project prompting to the Term Selecting phase (See Interactive Selection of Terms section below), a problem of obtaining mouse clicking location appears. Hence, the second phase of scaling visualisation comes out.
At the second phase, scale values attained from JSlider objects are passed to DataReader class and applied in relevant formulas to calculate variables such as points, widths and heights of rectangles. See Equation (1), (2), and (3). As shown in 25, 100 is set as the initial value for the slider, so that the visualisation shown when the visualisation generated at the first time is scaled as 100%.

6.5 Text Label Toggling

As the scale values become smaller, the strings begin overlap. This inspires a new feature; toggling labels in the visualisation, enabling users to focus only on the rectangles and colours. Hence a new desire appearing: hide strings on the visualisation. So that users can focus on the rectangles and colours only. Next, an event listener is registered to the JButton. When the button is clicked, its label value is replaced by “Text Off”. Consequently, with the default state ‘true’ is toggled ‘false’, then passed to the ConcordancePanel class. In the third step, a boolean value preset when drawing strings of terms will be assigned the same truth value as the boolean passed in. If it is ”true”, then draw the strings, else if it is ”false” then do not the drawString() method. Finally, we repaint the graphic. Figure 26 is a screen shot of concordance view with labels toggled off.

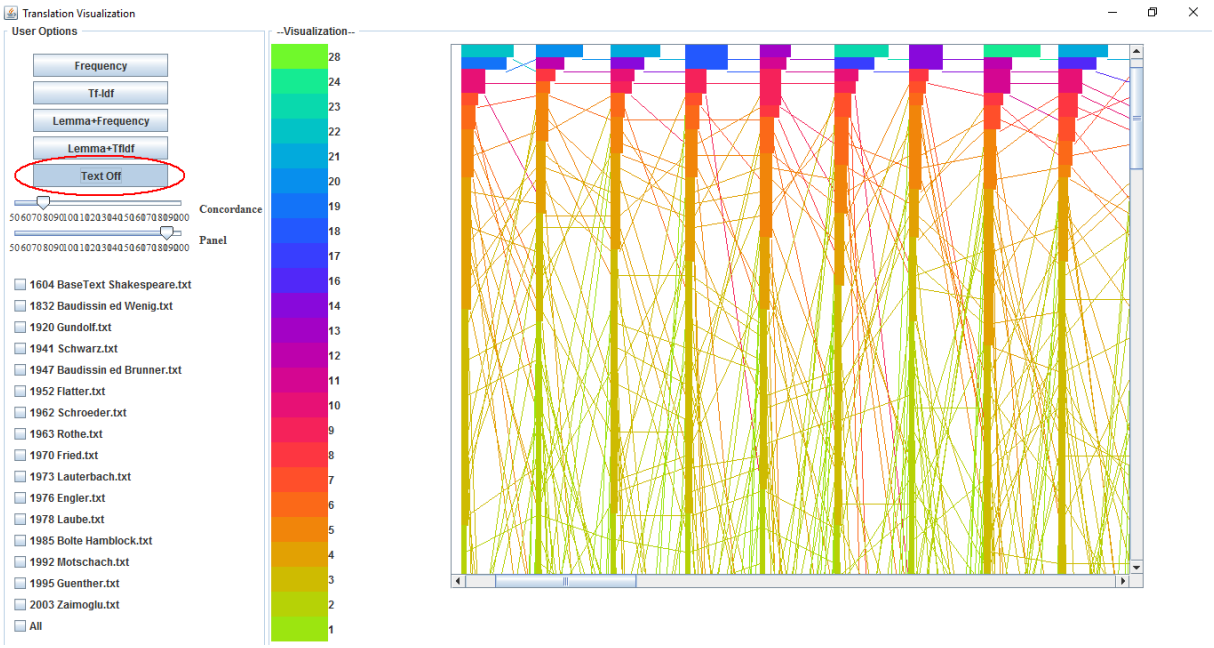


Figure 26: Results of toggling off text.

6.6 Adding, Subtracting, Selecting Items

To render the user option feature for selecting several concordances displaying on the panel, a new class called VersionChosenPanel is cre-

ated. By interacting with this feature, not only can the user select which concordance to display in the visualisation, but also the order in which they are displayed. Figure 27 shows the menu where users can select which versions to display.

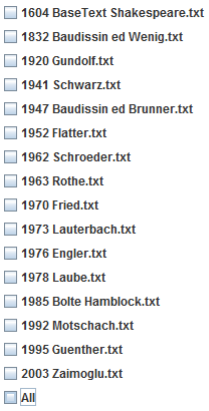


Figure 27: Version selection menu.

To implement the version selection feature we did the following steps:

- Rendering a list of JCheckBox objects to display the author name as the index.
- An event listener for each JCheckBox object, so that the action of selection can be generated as an object of class Object.
- Toggle the selecting status of the index.
- Generate a new list of Version objects according to the events passed from JCheckBox's ActionListener. Every time the user selects a name in the index, a new list of Version objects will be generated and passed to the ConcordancePanel.
- Redraw the concordance visualization by invoking ConcordancePanel's repain() method.
- Add an option to toggle the display of all concordances, which is responsible to display or hide all concordances as their original order.

Figure 28 shows the base text, which originally appears first in the display is now last.

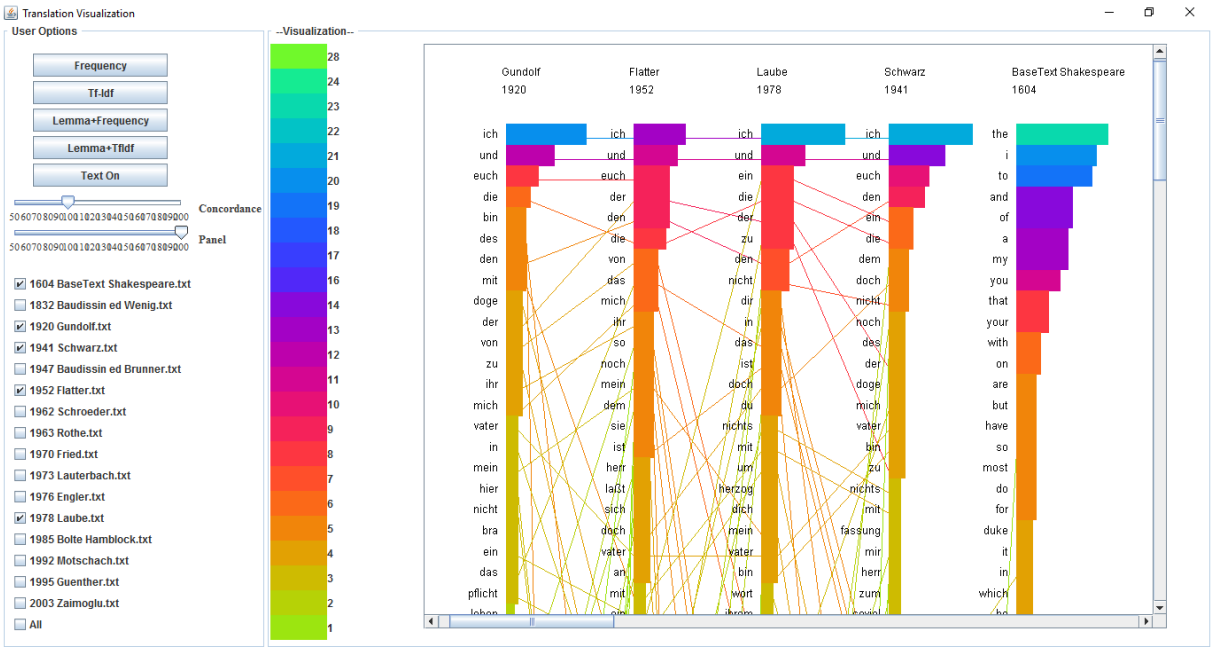


Figure 28: The screen shot of version selection feature.

6.7 Interaction and Selection of Terms

On the grounds that each concordance contains a large number of terms, the ability to highlight terms defined by the user is desired. In order to provide an features for terms, new features are putting in the visualisation. Therefore a clear view of highlighting terms comes out. This features are achieved by put into effect the following phases:

- Obtain the mouse pointer's co-ordinates via the getPoint() in MouseEvent class.

- Calculate which item region the point lies in. In this process, the regions of the display are divided as term regions. As the value of each region is calculated during data reading phase, the point passed from mouseClicked() method can be used to identify which region the point belongs. Further more, the Item object of this block is singled out and rendered.

- Identify the Item objects in other concordances sharing the same term, and pass them to [the highlighting method].
- Highlight the blocks of all selected Item objects. In this step, lines are drawn around the rectangles.
- Make other blocks transparent by setting the transparency values.
- Overwrite the colour values of all lines to make the lines connecting two highlighted items are also highlighted, while other lines becoming semitransparent.
- Retrieve the translation of this term and highlight the blocks in the original English version.

As a result, by clicking one single rectangle in the panel, the rectangle is highlighted and given an border while the colour of the other rectangles become transparent.additionally, the lines connecting identical terms are likewise highlighted by setting other lines transparent. An illustration of this feature can be seen in Figure 29

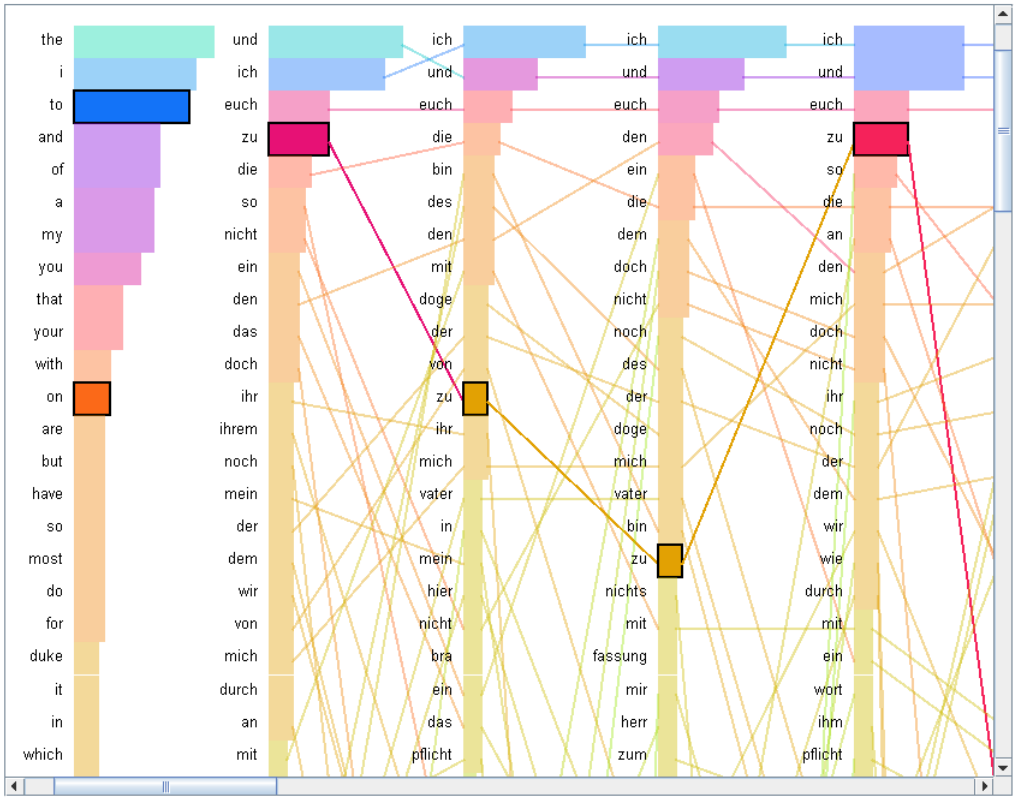


Figure 29: A visualisation of highlighting certain word in the concordance.

6.8 Colour Mapping

The colours selected are related by the number of occurrences of terms in each concordance. To increase visibility, the visualization should use as large a range of colours possible., which can be achieved through colour mapping. In this project, we instantiate the ColorLegendPanel class to fulfill this role. In addition, values of the frequency are displayed in the Colour Mapping view. The following is the process took to implement this function:

- Retrieve the colour value of each item from the DataReader class. An explanation of colour values can be found in Data Reading chapter.
- Instantiate JLabel objects as representation for colours.
- Add the JLabel objects to the panel.

- Display frequency values beside colour blocks.

The demo of Colour Mapping is shown in Figure 30.

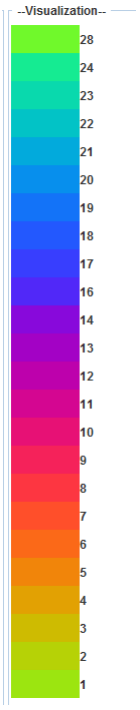


Figure 30: A screen shot of colour mapping.

6.9 Interactive Colour Legend

Colour mapping is not reserved for just static visualizations, we can dynamically change the map to create an interactive visualization. In this project, we add a feature enabling users to interact with the colour legend. If the user clicks a label in the colour legend, all blocks of that colour in the concordance view will be highlighted. This function is performed by identifying all items possessing the same frequency value. As illustrated in the Design chapter, an index of frequency values is generated in the DataReader class. After data reading phase, we can access this list of frequency values through an accessor method. By iterating through all values in the list, the ConcordancePanel class, where the list of Version objects is overwritten and the panel is repainted. As a result, by clicking one colour block in colour legend, all items sharing the same frequency, or colour, are highlighted using the same meth-

ods of highlighting described in Interaction and Selection of Terms chapter. Figure 31 serves as a demo to illustrate this feature.

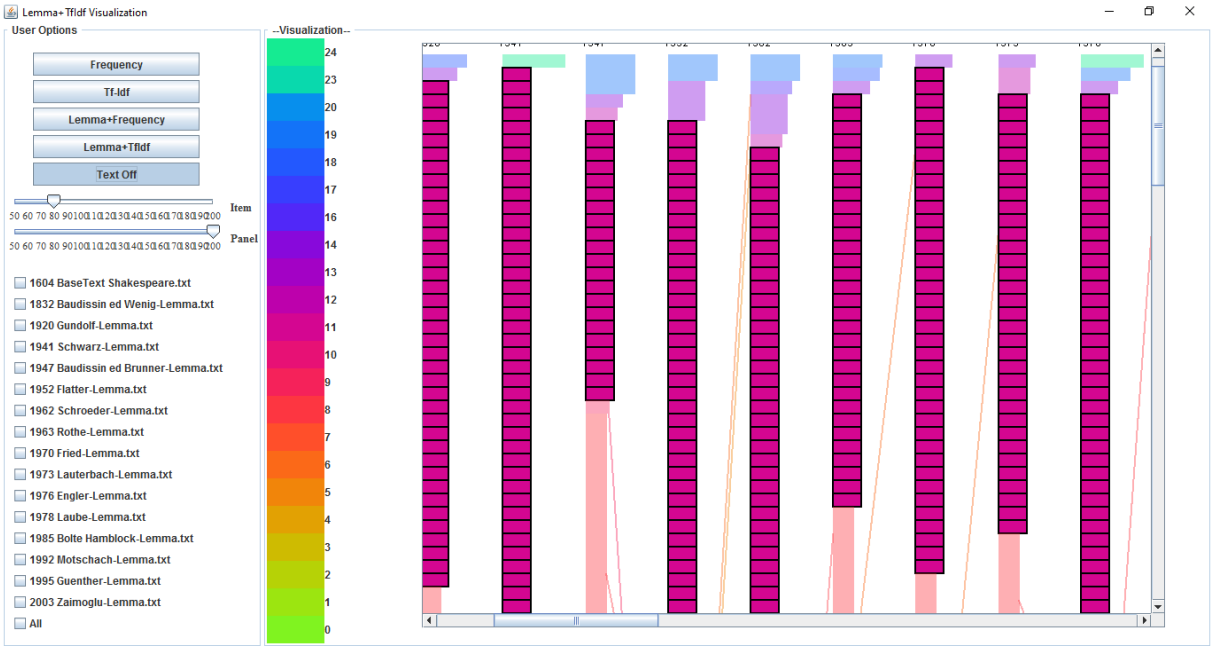


Figure 31: A screen shot of highlighting items sharing same number of frequency.

6.10 Lemmatisation

The lemma visualisation is a significant feature in this project. Lemma is a linguistic term which is described as the dictionary form of a word. Word 'decide' is the lemma for 'decided', 'decides'and 'deciding'. Accordingly, lemmatisation is the process to obtain the lemma for each word. To achieve the effect of lemma view, several steps are necessary:

- Obtain the German lemma corpus which contains an index of lemmas and words.
- Compare the words in our German translation corpus with the words in the German lemma corpus, and find the lemma for each term in our corpus.
- Store all the lemmas found and generate a new lemma index.

- Apply the lemmatisation results into visualisation.

For this project, an inevitable dilemma is the limited resources of German lemma corpus. As illustrated in Data Characteristics chapter, no relevant German lemma corpus was provided when we start this project. Also, German, as an affected language, is difficult to lemmatise. During this process, we attempted two solutions: Treetagger and DeReWo, which we explained in following sections.

6.10.1 TreeTagger

As introduced in the Technology Choices chapter, TreeTagger is a tool for annotating text data and lemma information. Figure 32 shows the User Interface of the TreeTagger. While applying this tool in the lemmatising task of the project, we found this tool has the following advantages:

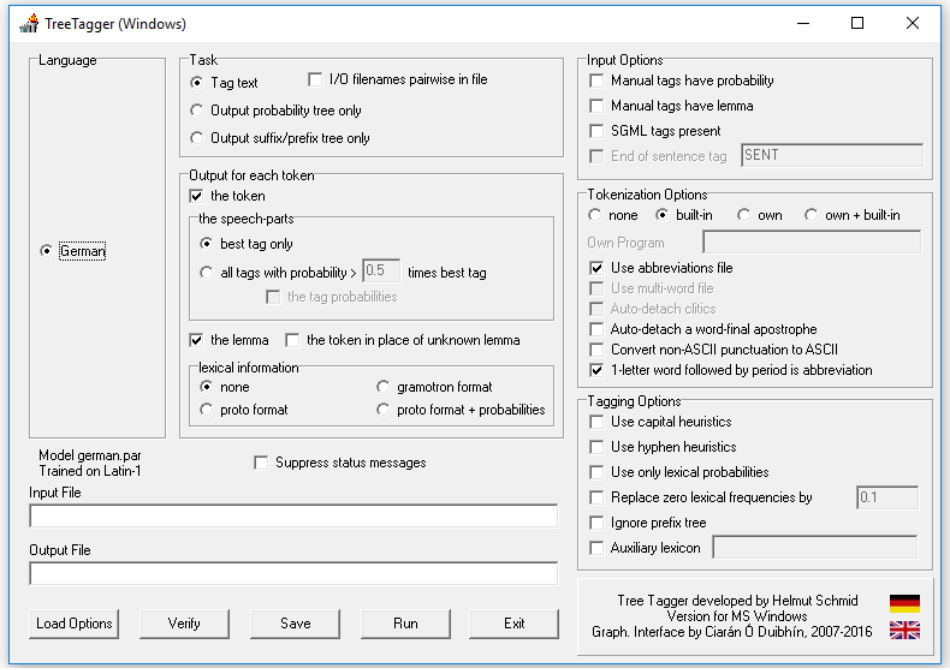


Figure 32: The User Interface of the TreeTagger.

- The software is easy to obtain, without any redundant procedures such as registration. The tool can be downloaded directly from the website `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`.

- The user interface of the tool is clear and easy to use.
- The software takes as parameters a .txt file and returns a .txt file of results.

However, there are also some problems we encountered:

- The format of the text file must be encoded in Latin-1, while the text file we have is encoded in UTF-8. Therefore, some German terms with special characters cannot be recognised by the tool.
- From the results we achieved, the tool cannot recognise words with capital letters.
- The tool cannot be used to lemmatise a batch of files at one time, which is not appropriate for a project which needs to process large datasets.

We connected a domain expert, Dr. Tom Cheesman from the College of Arts and Humanities at Swansea University, to evaluate the results of the lemmatisation produced by TreeTagger. Due to the low accuracy of the results for the data in this project, this solution was discarded in the end.

6.10.2 DeReWo

DeReWo is a project done by 'The Institute for the German Language'. This project aims at developing methods to create frequncy-based ranking list of lemmas based on random virtual corpora. In the DeReWo website `http://www1.ids-mannheim.de/direktion/kl/projekte/methoden/derewo.html?L=1`, there are some downloadable resources of German lemma, including 'DeReKo-2014-II-MainArchive-STT.100000.freq', which is a file storing top 100,000 German words, lemmas and POS. The format of this document is .freq, which can be edited in Visual Studio Code. It also can be read by Java directly. Using this corpus, we successfully obtained all the lemmas for each term in the *Othello* corpus for this project.

There are several phases of using DeReWo to generate lemma visualisation:

- Read text file from *Othello* corpus.
- Read German lemma corpus file.
- Search for the lemma of each term.
- Store the lemma in a new text file. In this step, we create 15 .txt files for all German translations. For each version of *Othello* translation, the two files (*Othello* source file and lemma file) are served as an index for words and lemmas.
- Replace all terms with their lemmas and visualise the new results.

Figure 33 shows the outcome of the lemma visualisation.

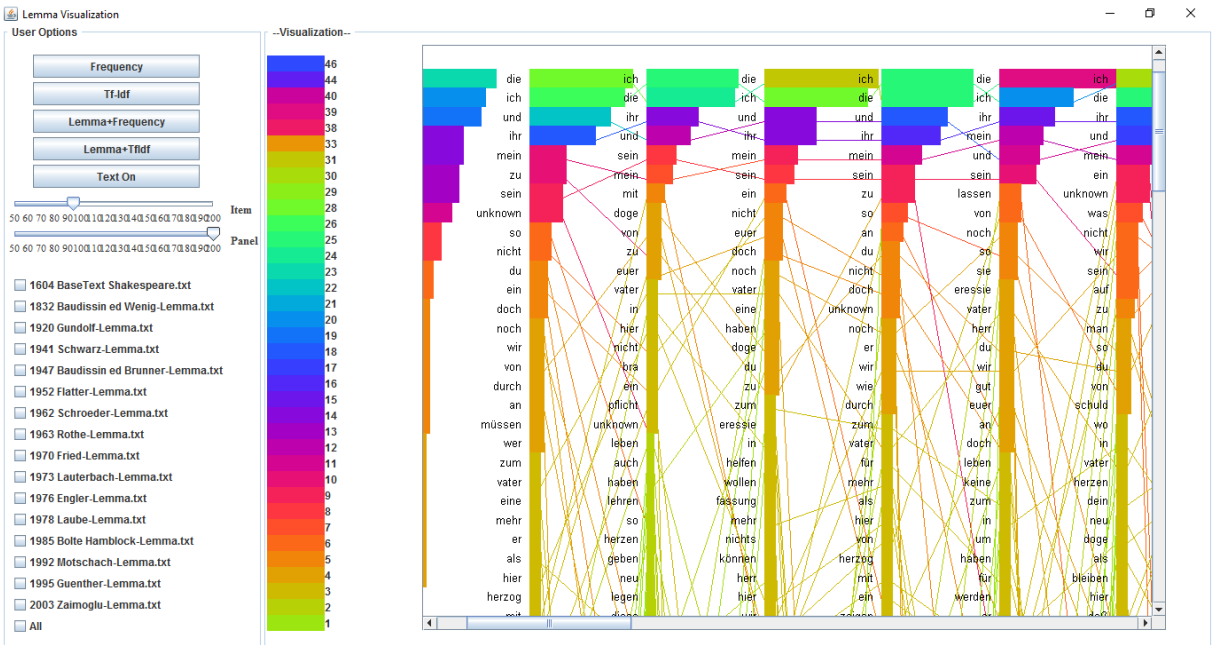


Figure 33: A screen shot of Lemma view.

6.11 Tf-Idf

Tf-Idf visualisation is an important feature in this project. As explained in the Design chapter, the Tf-Idf value represents the weightings of

words, which also means we can get rid of unimportant words, namely stopping words. Therefore, the visualisation provided will be more helpful for researchers to study the varieties of translation. To fulfill this function, the most challenging step is to implement the formula of the Tf-Idf into code [Manning et al., 2009]. The TfIdfCalculator class is created to process the Tf-Idf values.
The Tf-Idf visualisation generation process goes through the following steps:

- Calculate term frequency value. This step is done in the data reading stage. The Tf value hence can be achieved from DataReader object.
- Calculate Idf value.
- Replace the frequency with Tf-Idf value.
- Visualise according to the new results.

There are two visualisations generated using Tf-Idf value: one is to visualise the data using its Tf-Idf value and the original unchanged terms, as shown in Figure 34; the other form is visualized using Tf-Idf is to visualise using Tf-Idf value together with lemma data which is displayed in Figure 35.

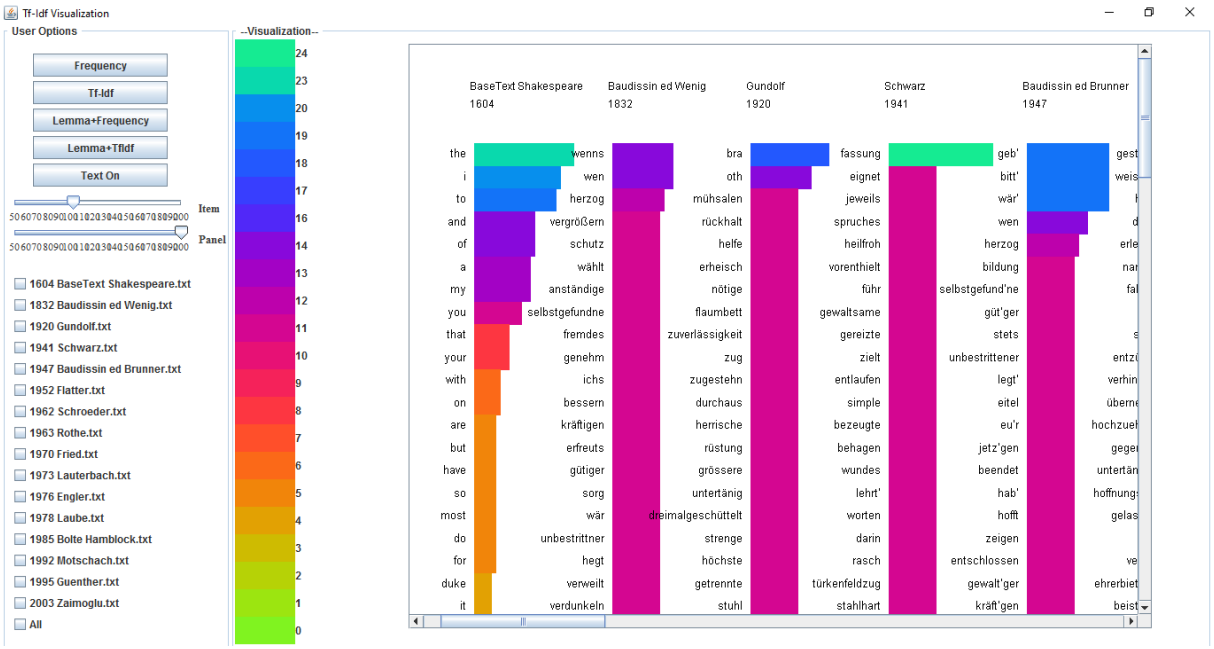


Figure 34: A result of Tf-Idf visualisation.

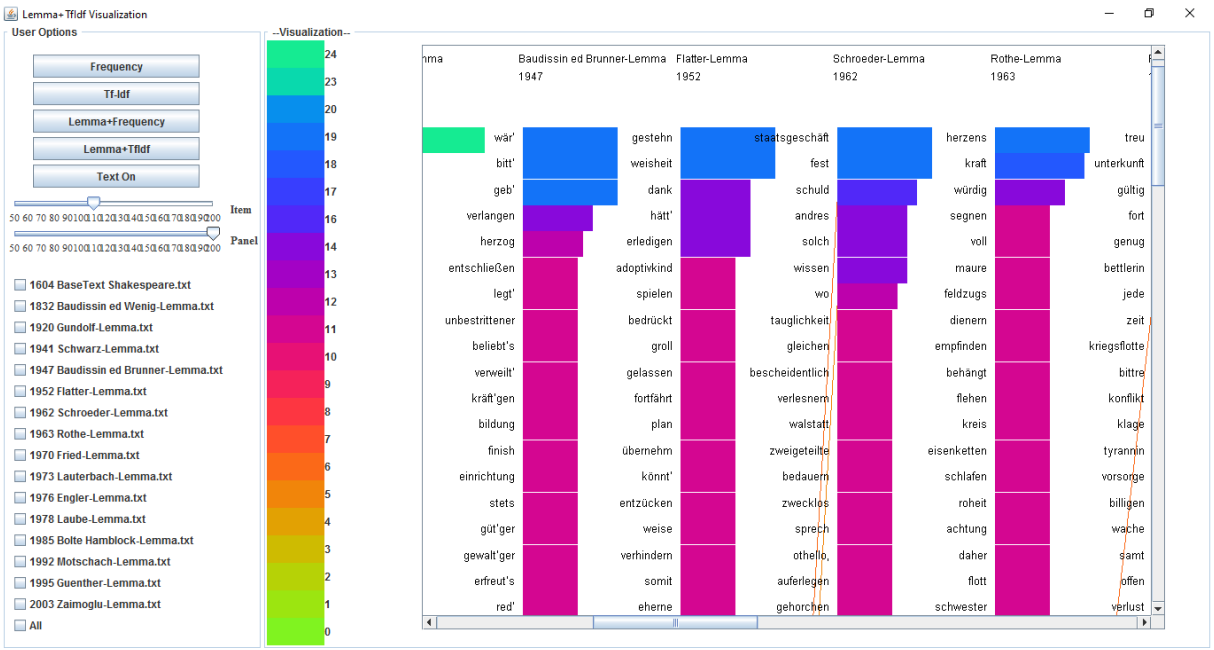


Figure 35: Lemma and Tf-Idf Visualisation.

7 Evaluation

In this section, we provide the description of the performance of the project, along with the illustration of our observations from the outcome of the visualizations. Feedback from a domain expert was also sought and forms part of this evaluation. evaluation.

7.1 Results

High Frequency View

The High Frequency View is the basic feature and the primary task in this project. In this visualisation, we provided a parallel view of concordances to display the most frequent words in each version of the translation. As shown in 29, the colour of blocks differs from each other and the width of blocks represents ranges from the longest to the shortest. A highlight feature allows user to see same terms in each concordance. However, from the results in the frequency visualisation, we can tell that the most frequent words in each version are the noise, and stop words such as 'ich' which means 'I' in English, or 'und' which means 'and' in English. These kinds of words are little help in translation comparison.

Tf-Idf View

The Tf-Idf View is another feature provided in this project. Based on features in High Freuency View, the Tf-Idf View displays the most important words in the concordance. Therefore, the results of this visualisation are quite different compared to the High Freuency View. In Figure 34, 27, terms in each concordance changed significantly. As illustrated in the Tf-Idf visualisation implementation section, if a word is listed on the top of a concordance, it means this word may appear many times in this version of the translation, while appearing not so frequently in the other concordance. For example, in Figure 36, we see

the word ’fassung’ meaning ’composure’ (in the 1941 version of the translation written by Schwarz) appears to have a high Tf-Idf value. When selecting ’fassung’, it appears that no other blocks are highlighted, which means this word is not used by other authors. There are several guesses for this result:

- The word ’fassung’ is used multiple times in this translation version. So this is used to translate certain word or express specific meaning. (In this text, it is used to translate ’patience’ or express ’the state of being calm and in control of oneself’)
- This word does not appear in other versions, and we can assume that special trans- lation techniques were adopted, such as reformulation, adaptation, or compensation.
- This unique outcome is likely because of culture influences. Different cultural influences may lead to nuances in linguistic expression. In other words, the author may come from a distinct region when compared with other places and people can use different words to express the same meaning.

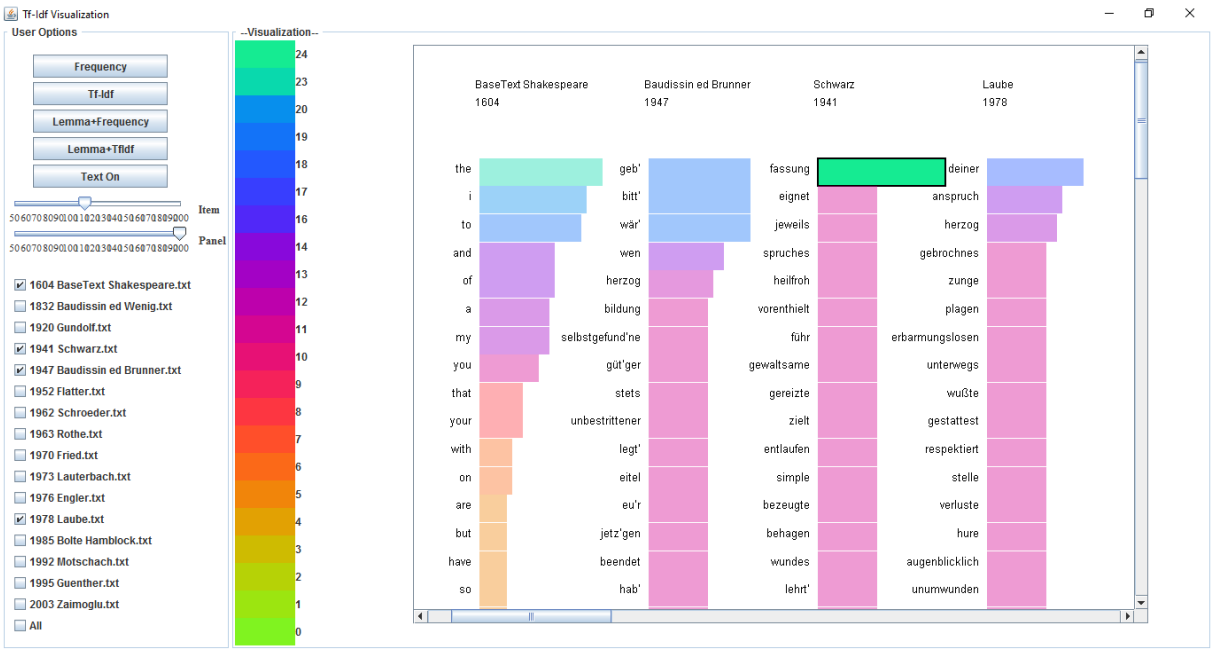


Figure 36: The word ’fassung’ is ranked high Tf-Idf value in one version

Lemmatised View

Lemmatised View is generated after applying a lemma corpus to process our data. After lemmatisation, words are supposed to change into the original form, namely, the dictionary form (See Implementation chapter for relevant explanation). This feature is designed to combine the same words with different inflected forms. For example, the English word 'you' can be translated into 'dir', 'du', 'sie' and 'euch' in German. Figure 37 is the result after we select 'you' in the base text concordance. However, in Lemmatised View, only 'ihr' is highlighted.

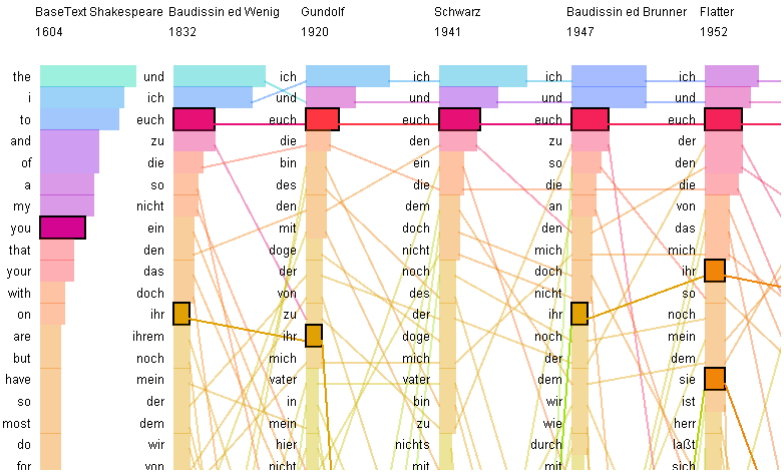


Figure 37: After clicking the block of English 'you' in base text, all German translations in other concordances are highlighted, such as 'dir', 'du', 'sie' , 'euch', and 'euch'.

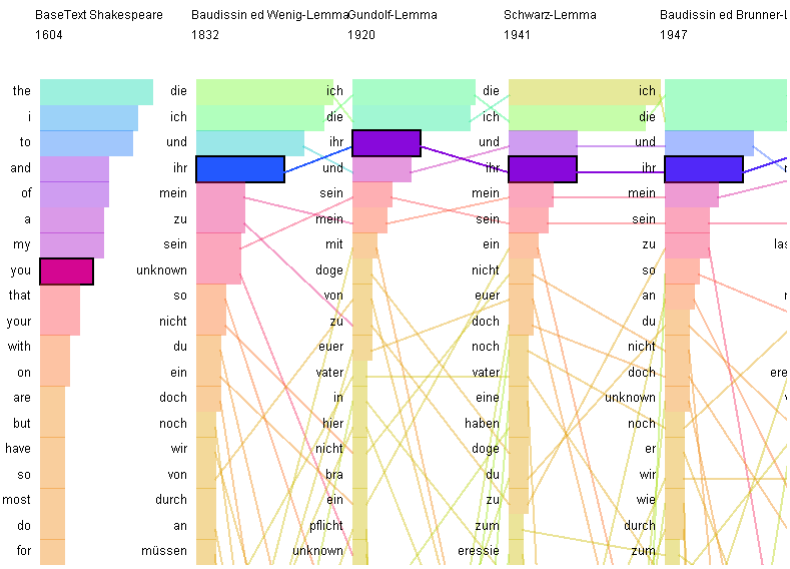


Figure 38: After clicking the block of English ’you’ in base text, only one German word is highlighted.

As a result of lemmatisation, the length of column is shorter in the lemma view than in the frequency view. Also the frequency of words are changed. This can be seen from the colour legend in 39. Similarly it can be assumed that the variety of words are more obvious. However, more proofs and interpretations need to be explored in the future.

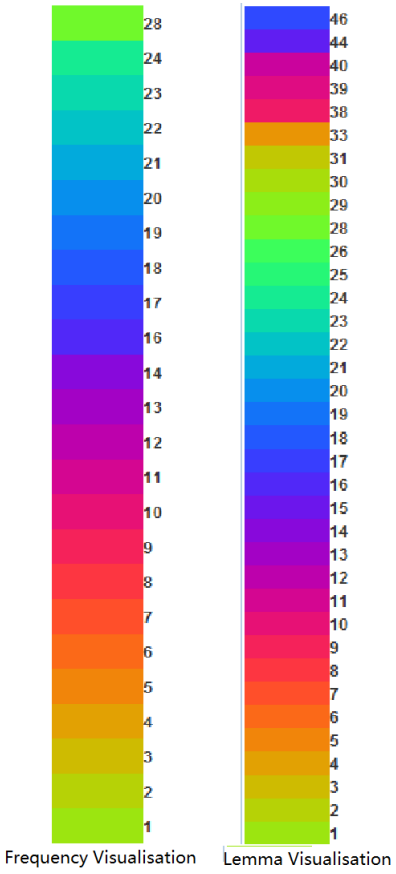


Figure 39: After combining the inflected form of words, the frequencies are incremented.

Lemmatised Tf-Idf View

The last view we rendered is the Lemmatised Tf-Idf View. In this view, both the techniques of lemmatisation and Tf-Idf are utilized to display a parallel translation comparison in the software. From this visualisation, not only can we see the dictionary words, but the stop words are filtered. To prove the advantages of these results, two comparisons is necessary:

- **Tf-Idf View vs Lemmatised Tf-Idf View**
  The German word 'wen', meaning 'whom' in English, which bears little content information but merely performs grammar function. In Tf-Idf visualisation of Figure 34, the word 'wen' ranked on the top of second version from the left which is written by Baudissin ed Wenig in 1832 [Hotho et al., 2005]. However, in Figure28,

which applied both lemmatisation techiniques and Tf-Idf algorithm, this word disappears. This can be caused by the virtue that the word is combined with the dictionary word of 'wen', which has low Tf-Idf value.

- **Lemmatised View vs Lemmatised Tf-Idf View**
  If we only apply lemmatisation in the visualisation, stop words are still present. In Figure 33, words such as 'die'('the' in English), 'ich'('I' in English), and 'und'('and' in English) are all considered as stop words [Hotho et al., 2005], which contribute little in translation studies. In the Lemma and Tf-Idf view, these words have disappeared because they are ranked to the bottom.

7.2 Domain Expert Feedback

A domain expert, Dr. Tom Cheesman from the College of Art and Humanities at Swansea University, was invited as the user to give feedback on this project. During the meetings, we demonstrated the visualisations and features to him. The first one was organised on 13th November, 2017. Following the feedback, some features were overwritten and several new features were developed. The second meeting was on 4th December, 2017, in which he approved the new features.

7.2.1 Session 1

At this stage, High Freuency View was finished, along with features such as turning the visualisation on and off, scaling the frame, version selection, and colour legend were implemented. Dr. Tom Cheesman expressed his interest by leaning his body to watch closer to the laptop. He asked some questions such as "What do the numbers beside colour legend represent?", "What are the connections?". He also requested that only the base text and three other German translations were shown so that he could understand the alignments. In the feed back, words like 'interesting', 'useful' and 'good' were used a lot. Mean-

while, other suggestions were mentioned and discussed, such as filtering stop words and seeing the most important words, together with showing the lemma of the words to decrease inflected words.

7.2.2 Session 2

In this stage, according to the feedback Dr. Cheesman gave at the first meeting, we did some more changes for the project: utilizing Tf-Idf algorithm in data processing, and using the German lemma corpus to lemmatize the terms.
In the meeting, Dr. Cheesman expressed his interest and excitement by saying ”This is good, this is pulling up some interesting stuff”, ”This is great...I can play with this”. He showed great interest in the Lemma and Tf-Idf view, and pointed out some remarkable translations such as abbreviation words in the version written by Baudissin ed Brunner in 1947, (See Figure 40). From the view, we can tell that more abbreviation words are used in this version which implies there is a unique translation strategy which the author uses. At the end of this meeting, Dr. Cheesman asked for a copy of the tools for assisting his translation studies.

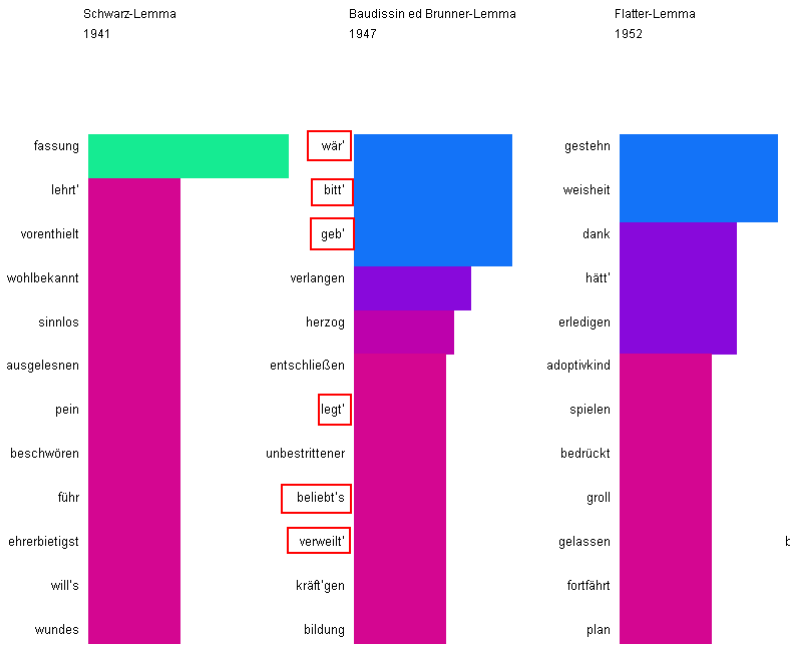


Figure 40: After combining the inflected form of words, the frequencies are incremented.

8 Conclusion

In conclusion, data visualisation is becoming a fundamental necessity in various domains due to the need for understanding increasingly large volumes of information. This project, which aims at developing an interactive visualisation system, reveals that there is great potential in the ways that data visualisation can be exploited in information analysis.
The first aim for this project was to create a parallel text visualisation to assist users comparing words in different translation versions. This aim is met with the four visualisations:

- Visualise the most frequent words in each version. This has been achieved in the Frequency Visualisatoin.
- Visualise the most important words in each verson. This feature can be seen from the Tf-Idf Visualisation.
- Visualise the most frequent dictionary words in each version, which is provided in Lemma and Frequency Visualisation.
- Visualise the most important dictionary words in each version. This visualisation is displayed in the Lemma and Tf-Idf Visualisation.

All these four visualisations have been accomplished in this project. However, the performance of the visualisations remain to be improved in the future. For example, all the four visualisations are parallel texts, so the Frequency Visualisation can be combined into one of other three visualisations.
The second aim is to provide a software application which enables interactive user options. This is also met by rendering following features:

- **Button** used to toggle the visualisations.
- **Slider** used to scale the visualisation.

- **Slider** used to scale the frame.
- **Button** used to turn on and off the texts in the visualisation.
- **Menu** used to select certain concordances. It also can be used to arrange the order of concordances.
- **Interactive Colour Legend** used to visualise frequency or Tf-Idf values of the words. It also can be used to view words with certain values.
- **Interactive blocks** used to view specific words and the same words in other versions by highlighting these words.

Overall, the project can be deemed to be a success. However, there are some issues with the performance of the algorithm when version selection is attempted in Lemma visualisation. A limitation with highlighting items has also been identified if the slider bars are notmoved. A video demonstration of the software is available along with other resources produced for the project at the following web address: `http://cs.swansea.ac.uk/~cswang/Xiaoxiao_Othello/Doxygen/`

9 Future Work

The first priority to continue this project would be to lemmatise the English version text. The project for now is only supported lemma views for German lemmatisation, with English version still keep many inflected words such as 'I', 'my', me. Further more, if weightings for these English words calculated in specific contexts is rendered, this project may display new results. For the English lemmatisation, this can be accomplished by introduced extra English lemmatisation libraries, such as Stanford NLP Java library. For the English Tf-Idf, this can be attempted by introduce the results from VVV project, which has a corpus stored all the Tf-Idf values computed in the corpus of all Shakespeare works in English.

The software could also be extended to combine the views into one or two visualisation. For example, to provide two views such as Tf-Idf Visualisation and Lemma and Tf-Idf Visualisation in one visualisation, which would allow users to compare different results. In order to do this, new frame can be added to enable the comparison view.

Further improvements could be made to the software such as enabling users to search words by adding a 'search box'. This can be worked out by applying 'Text Field' in Java.

References

[Agi,] Agile vs Waterfall.

[Sta,] Software - The Stanford Natural Language Processing Group.

[Tag,] TagCrowd – create a word cloud from any text.

[Tre,] TreeTagger.

[Jbu,] Tutorial: Making annoying rainbows and other color cycles in Javascript.

[Adenowo and Adenowo, 2013] Adenowo, A. A. A. and Adenowo, B. A. (2013). Software Engineering Methodologies: A Review of the Waterfall Model and Object-Oriented Approach. *International Journal of Scientific & Engineering Research*, 4(7):427–434.

[Alrehiely, 2014] Alrehiely, M. M. (2014). Visualization of version variation. (October).

[Cao and Cui, 2016] Cao, N. and Cui, W. (2016). Overview of Text Visualization Techniques. In *Introduction to Text Visualization*, pages 11–40.

[Cheesman et al., 2012] Cheesman, T., Laramee, R. S., Hope, J., Flanagan, K., and Thiel, S. (2012). Version Variation Visualization.

[Geng et al., 2015] Geng, Z., Cheesman, T., Laramee, R. S., Flanagan, K., and Thiel, S. (2015). ShakerVis: Visual analysis of segment variation of German translations of Shakespeare's Othello. *Information Visualization*, 14(4):273–288.

[Geng et al., 2011] Geng, Z., Laramee, R. S., Cheesman, T., Ehrmann, A., and Berry, D. M. (2011). Visualizing Translation Variation : Shakespeare ' s Othello. pages 653–663.

[Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad; Buckley, 2014] Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad; Buckley, A.

(2014). The Java® Language Specification - jls8.pdf. *Addison-Wesley*, page 688.

[Hotho et al., 2005] Hotho, A., Nürnberger, A., and Paaß, G. (2005). A Brief Survey of Text Mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 20:19–62.

[Jong et al., 2008] Jong, C.-h., Rajkumar, P., and Siddiquie, B. (2008). Documents. pages 1–8.

[Kucher, 2014] Kucher, K. (2014). Text Visualization Browser : A Visual Survey of Text Visualization Techniques. *InfoVis*, (November 2014).

[Laramee, a] Laramee, R. CSCM94 Software Engineering Principles Introduction and the Software Crisis.

[Laramee, b] Laramee, R. S. Data Visualization : An Introduction.

[Laramee, 2009] Laramee, R. S. (2009). Bob ' s Concise Coding Conventions (C3). pages 1–11.

[Laramee, 2010] Laramee, R. S. (2010). Bob ' s Minutes of Meeting Protocol : Incentive and a Description Sample Minutes. pages 1–2.

[Laramee, 2011] Laramee, R. S. (2011). Bob's project guidelines: Writing a dissertation for a BSc. in computer science. *ITALICS Innovations in Teaching and Learning in Information and Computer Sciences*, 10(1):43–54.

[Liu,] Liu, X. Interactive Visualization of Shakespeare ' s Othello.

[Manning et al., 2009] Manning, C. D., Ragahvan, P., and Schutze, H. (2009). An Introduction to Information Retrieval. *Information Retrieval*, (c):1–18.

[Stefan and Wrisley, 2017] Stefan, J. and Wrisley, D. J. (2017). Interactive Visual Alignment of Medieval Text Versions. *IEEE Visual Analytics Science and Technology (VAST)*.

[Tom et al., 2012] Tom, C., Kevin, F., and Stephan, T. (2012). delighted-beauty.org.

[Ward et al., 2015] Ward, M., Grinstein, G., and Keim, D. (2015). *Interactive Data Visualization*.

[Williams et al., 1995] Williams, J. G., Sochats, K. M., and Morse, E. (1995). Visualization. *Annual Review of Information Science and Technology (ARIST)*, 30.

Appendices

A Minutes of Meeting

Minutes of Meeting: Bob, Rhodri, Xiaoxiao, Carlo, Mohammed

Date: 1 December 2017
Start time: 15:00
End time: 16:00
Date and time of next meeting: Friday, 8 December 2017, 15:00 (Last ”Official” meeting)
Topics discussed:
– Das Institut für Deutsche Sprache–Corpus-Based Lemma and Word Form Lists(http://www1.ids-mannheim.de/direktion/kl/projekte/methoden/derewo.htr
– Gregynog Presentation
– Meng initial document
Progress:
– 3rd year project initial document feedback
– Carlo: Implements recursive web crawler in python
– Carlo: Started sentiment analysis in NLTK in python
– Xiaoxiao: Working on swithing between TF-IDF and high frequency coloumns
– Rhodri: Plot of 1 Atom over time
– Rhodri: Draft of initial document
TODO:
– Mohammed: Ask Ben Mora for initial document specification and deadline
– Mohammed: Demonstrate working example of chess game
– Mohammed: Investigagte existing chess engines in java
– Xiaoxiao: Send link of German lemmatization ot Bob, Mohammed, Tom and Angelika
– Xiaoxiao: Prepare for demonstration on Monday
– Xiaoxiao: Try to get TF-IDF verses frequency terms user option working

– Xiaoxiao: Investigate writing help services- visit English language training services- 3rd floor Margam Building- ask if they can help
– Xiaoxiao: Send email to 2nd supervision asking for MSc viva on afternoon of 14, 15, Dec. (CC: Bob)
– Rhodri: Try to verify correctness of Atom pth
– Rhodri: Show starting positions of Atoms and compass with VMD (lipids) lipids only
The above is only one meeting. 'Bob's Minutes of Meeting Protocol' [Laramee, 2010] is followed to document the meetings. The URL below contains all the other minutes of meetings. `http://cs.swansea.ac.uk/~cswang/minutes/` `http://cos-ugrad.swansea.ac.uk/838940/minutes/`

B Documentation

The file of Doxygen documentation is presented in the following pages. The URL below contains the full Doxygen docuementation:
`http://cs.swansea.ac.uk/~cswang/Xiaoxiao_Othello/Doxygen/`

My Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

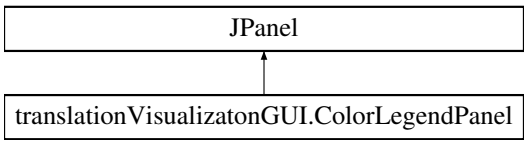
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 translationVisualizatonGUI.ColorLegendPanel Class Reference

Inheritance diagram for translationVisualizatonGUI.ColorLegendPanel:



Public Member Functions

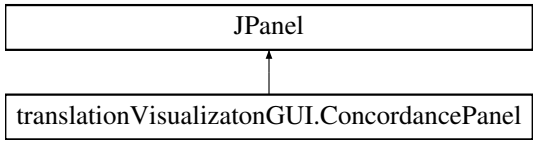
- GridBagConstraints **getConstraint** ()
- ConcordancePanel **getM_ConcordancePanel** ()
- void **setM_ConcordancePanel** (ConcordancePanel m_ConcordancePanel)
- void **setColorLegend** (List< Map.Entry< Integer, Color >> m_ColorIndex, List< Map.Entry< String, Integer >> m_FrequencyIndex)

The documentation for this class was generated from the following file:

- ColorLegendPanel.java

3.2 translationVisualizatonGUI.ConcordancePanel Class Reference

Inheritance diagram for translationVisualizatonGUI.ConcordancePanel:



Public Member Functions

- ConcordancePanel (List< Version > versionList)
- boolean **isFirstVersion** ()
- void **setFirstVersion** (boolean firstVersion)
- boolean **getM_OnAndOff** ()
- void **setOnAndOff** (boolean onAndOff)
- int **getScaleValue** ()
- void **setScaleValue** (int scaleValue)
- DataReader **getDataReader** ()
- void **setDataReader** (DataReader dataReader)
- int **getM_VersionNumber** ()
- void **setM_VersionNumber** (int m_VersionNumber)
- List< Version > **getM_VersionList** ()
- void **setM_VersionList** (List< Version > m_VersionList)
- Version **getM_singleVersion** ()
- void **setM_singleVersion** (Version m_singleVersion)
- double getZoomValue ()
- void setZoomValue (int zoomValue)
- Point **getHighlightPoint** (Point eventPoint, int scaleValue)
- void displaySingleVersion (List< String > VersionSelected)
- void **resetLocations** ()
- void scaleConcordancePanel (int scaleValue)
- void freqHighlight (String token)
- void tokenHighLight (Point point)
- void **clickTransVersion** (int versionNumber, int lineNumber, Version choosenVersion, Item choosen↩ Concordance)
- void **clickBaseText** (int versionNumber, int lineNumber, Version choosenVersion, Item choosen↩ Concordance)
- void drawTitleString (Version version, Graphics g)
- void drawTokenStrings (Graphics g)
- void drawRectangles (Graphics g)
- void paintComponent (Graphics g)
- void **drawLines** (Item concordanceCompare, Graphics g)
- boolean **isFreqOrTfidf** ()
- void **setFreqOrTfidf** (boolean freqOrTfidf)
- Version **getM_Version** ()
- void **setM_Version** (Version m_Version)
- Item **getM_Concordance** ()
- void **setM_Concordance** (Item m_Concordance)

Public Attributes

- List< Version > m_VersionList =new ArrayList<Version>()
- boolean **freqOrTfidf** =false

3.2.1 Constructor & Destructor Documentation

3.2.1.1 ConcordancePanel()

```
translationVisualizatonGUI.ConcordancePanel.ConcordancePanel (
            List< Version > versionList )  [inline]
```

Constructor

Parameters

versionList	

3.2.2 Member Function Documentation

3.2.2.1 displaySingleVersion()

```
void translationVisualizatonGUI.ConcordancePanel.displaySingleVersion (
            List< String > VersionSelected )  [inline]
```

Parameters

List<String>	

3.2.2.2 drawRectangles()

```
void translationVisualizatonGUI.ConcordancePanel.drawRectangles (
            Graphics g )  [inline]
```

Draw all the rectangles for tokens

Parameters

g	- the Graphic to be drawed

3.2.2.3 drawTitleString()

```
void translationVisualizatonGUI.ConcordancePanel.drawTitleString (
            Version version,
            Graphics g )  [inline]
```

Draw the stirng of titles, including author names and publish years

Parameters

version	- current Version object
g	- the Graphic to be drawed

3.2.2.4 drawTokenStrings()

```
void translationVisualizatonGUI.ConcordancePanel.drawTokenStrings (
            Graphics g )  [inline]
```

Draw token strings

Parameters

g	- the Graphic to be drawed

3.2.2.5 freqHighlight()

```
void translationVisualizatonGUI.ConcordancePanel.freqHighlight (
            String token )  [inline]
```

if one color on color legend panel is clicked, all tokens with the ame frequency will be highlighted

Parameters

token	

3.2.2.6 getZoomValue()

```
double translationVisualizatonGUI.ConcordancePanel.getZoomValue ( )  [inline]
```

a formula is applied here to make sure we return a float value to fit the .scale() method.

Returns

zoomValue

3.2.2.7 paintComponent()

```
void translationVisualizatonGUI.ConcordancePanel.paintComponent (
            Graphics g )  [inline]
```

Draw the version visualization on ConcordancePanel. This method is called from ConcordancePanel.

3.2.2.8 scaleConcordancePanel()

```
void translationVisualizatonGUI.ConcordancePanel.scaleConcordancePanel (
            int scaleValue )  [inline]
```

rescale concordance panel, recalculate the locations and widths

Parameters

scaleValue	

3.2.2.9 setZoomValue()

```
void translationVisualizatonGUI.ConcordancePanel.setZoomValue (
            int zoomValue )  [inline]
```

this method pass zoomValue from translationVisualization.getM_Slider().addChangeListener()

Parameters

scaleValue	

3.2.2.10 tokenHighLight()

```
void translationVisualizatonGUI.ConcordancePanel.tokenHighLight (
            Point point )  [inline]
```

set the transparency of the rectangles' when one token is clicked

Parameters

point	

3.2.3 Member Data Documentation

3.2.3.1 m_VersionList

```
List<Version> translationVisualizatonGUI.ConcordancePanel.m_VersionList =new ArrayList<Version>()
```

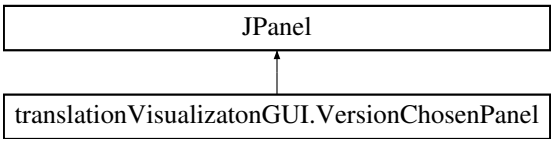
the list of versions passed from translation visualization

The documentation for this class was generated from the following file:

- ConcordancePanel.java

3.3 translationVisualizatonGUI.VersionChosenPanel Class Reference

Inheritance diagram for translationVisualizatonGUI.VersionChosenPanel:



Public Member Functions

- String [ ] getM_FilePath ()
- void **setInitialFilePath** ()
- void **setGermanLemmaFilePath** ()
- void **addAllVersions** (String[ ] string)
- List< String > **getM_versionNames** ()
- void **setM_versionNames** (List< String > m_versionNames)
- List< JCheckBox > **getM_checkList** ()
- void **setM_checkList** (List< JCheckBox > m_checkList)
- JCheckBox **getM_VersionNameCBox** ()
- void **setM_VersionNameCBox** (JCheckBox m_VersionNameCBox, String str)
- void **allSelection** (ActionEvent actionEvent, ConcordancePanel concordancePanel)
- void **singleSelection** (ActionEvent actionEvent, ConcordancePanel concordancePanel, Boolean one↩ Selected)
- ActionListener **add_ActionListener** (ConcordancePanel concordancePanel)
- void **addVersions** (List< String > versionNameList, ConcordancePanel concordancePanel)
- void **initialize** (ConcordancePanel concordancePanel, List< String > versionNames)
- GridBagConstraints **getConstraint** ()

Public Attributes

- List< String > m_versionNames
- String [ ] **m_FilePath**

3.3.1 Member Function Documentation

3.3.1.1 getM_FilePath()

```
String [ ] translationVisualizatonGUI.VersionChosenPanel.getM_FilePath ( )  [inline]
```

Returns

file path string array

3.3.2 Member Data Documentation

3.3.2.1 m_FilePath

```
String [] translationVisualizatonGUI.VersionChosenPanel.m_FilePath
```

Initial value:

```
={ "1604 BaseText Shakespeare.txt", "1832 Baudissin ed Wenig.txt", "1920 Gundolf.txt", "1941 Schwarz.txt",
            "1947 Baudissin ed Brunner.txt",    "1952 Flatter.txt", "1962 Schroeder.txt",
            "1963 Rothe.txt", "1970 Fried.txt", "1973 Lauterbach.txt",
            "1976 Engler.txt", "1978 Laube.txt", "1985 Bolte Hamblock.txt",
            "1992 Motschach.txt", "1995 Guenther.txt", "2003 Zaimoglu.txt" }
```

3.3.2.2 m_versionNames

```
List<String> translationVisualizatonGUI.VersionChosenPanel.m_versionNames
```

the list of String to store version names and pass this list to concordance panel and repaint new panel with versions only selected

The documentation for this class was generated from the following file:

- VersionChosenPanel.java

Appendix

RECORD OF SUPERVISION

NB: This sheet must be brought to each supervision and submitted with the completed Dissertation

(to be completed as appropriate by student and supervisor at the end of each supervision session, and initialed by both as being an accurate record. NB it is the student's responsibility to arrange supervision sessions and he/she should bear in mind that staff will not be available at certain times in the summer)

Student Name:
Student Number:
Dissertation Title:
Supervisor:

Supervision	***Date, duration***	***Notes***	***Initials Supervisor***	***Initials student***
1: Brief outline of research question and preliminary title (**by pre June**)				
2: Discussion of detailed plan and bibliography (**by June**)				
3: Progress report, discussion of draft chapter (**by August**)				
4: (optional) progress report (**by September)**				
5: Submission (**by 30 September**)				

Statement of originality

I certify that this dissertation is my own work and that where the work of others has been used in support of arguments or discussion, full and appropriate acknowledgement has been made. I am aware of and understand the University's regulations on plagiarism and unfair practice.