Interactive Visualization of Shakespeare's Othello

Prifysgol Abertawe
Swansea University

Xiaoxiao Liu

Department of Computer Science
Swansea University

This dissertation is submitted for the degree of
Master

September 2015

Dedication

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 40,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 100 figures.

MoHo.Khaleqi
September 2015

Acknowledgements

And I would like to acknowledge ...

Acknowledgements

Contents

List of Figures

List of Tables

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:

- **1**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.
- **2** To design a software of textual data visualization to display more information by compare different versions of translations, such as time span, genre, interpretation.
- **3** 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 [**?**],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 paper 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 firstly introduced to present the most relevant works to this project. In the second part, previous systems in similar project is introduced. The third part provides an detailed analysis of the data characteristics.

2.1 Literature Review

This section introduced principles and techniques for data preprocessing and text data visualization.

Text Visualisation Browser [Kucher, 2014] is an online tool providing the most comprehensive summary of published text visualization [**?**]. According to Text Visualisation Browser, from 1976 to 2017, there are 400 published text visualisation papers in total, in which 396 publications are aim to analyse text alignment. By searching "Word", there shows 20 publications, and "Translation" gets 16 results. Whereas when typing "Frequency" and "Weighting", each key word get 1 results. Also, key words such as "Machine learning, "Data Mining", "Natural Language Processing" got no publication collected. The results indicate that in text visualization domain, most researches 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 with this project. However, applying more specific techniques such as "Natural Language Processing" haven't been applied in text visualisation widely.

2.2 Previous Systems

In this section, interactive visualization of previsous system are introduced. For each research, data set and visualization techniques will be presented. Also, closely related work will be discussed in this section.

Interactive Exploration of Versions across Multiple Documents

Work of [Jong et al., 2008] provide a interactive visualization tool, MultiVersioner, to address the issues of comparing several versions of texts.

2.3 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 correspond to Act1, Scene 3 of the English version of *Othello* (1604) play as 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 is an screen shot of the text data used in the project.

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

- The aim of this project focuses on word processing, which require 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 by Java, it is easier and safer to convert the .docx format into .txt format.
- There exists methods in Java.io, a Java API, used to read .txt data directly from files.
- Apart from the basic and simple information (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.

```
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 1: The text in the .txt file. All data has been segmented and cleaned

3 Project Specification

This part is the specification of the project which includes the features specification and technology choices to the software.

3.1 Feature Specification

3.2 Technology Choices

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 back to steps if they found some problems at last. The progress of the Waterfall Model is, according to [**?**], include 5 phases: Requirement analysis, design, implementation, testing, and operation and maintenance.(See Figure **??**)

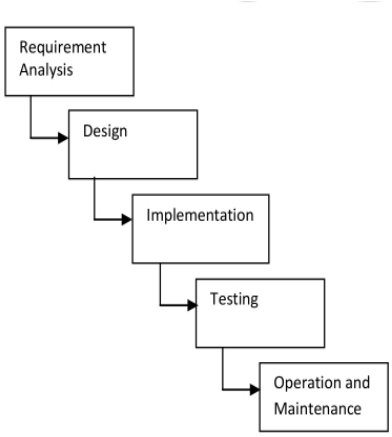

Figure 2: The 5 phases of Waterfall Model ([**?**])

However, when projects run out of time, testing phase will be cut, which may lead to poor quality of the outcomes. In addition, the operation is in the last step, developers may be unaware of where they've gone and what they've done, it is invisible for developers to know the progress. Last but not least, it is impossible for developers to change until the last phase.

Unlike with the Waterfall methodology which separates the whole project into several phases and implement 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. And hence the quality and visibility issues of Waterfall methodology are solved. Hence, we adopt Agile as our guiding methodology when implementing this project.

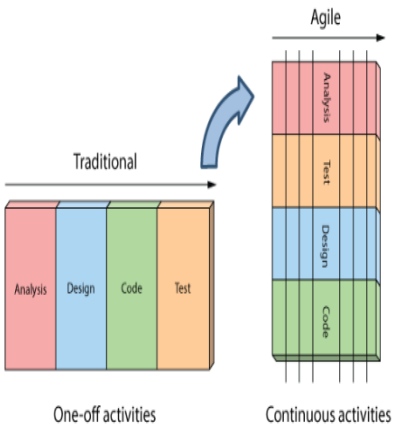

Figure 3: Comparison between Waterfall methodology and Agile methodology ([**?**])

4.2 Project Timetable

This section indicates time management for the project. These project separates into 5 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 indicates the Gantt of the project timeline. This project initiated from 17th February, and the final deadline is 30th September. In every phase, there are several tasks to be done. Most of the tasks in phase one has been done, except data preprocessing which needs more time to process more text data. The second phase is expected to finish before July. So more can be used in the implementation phase. Software implementation is supposed to spend the most time, which will be executed according to the designs done by previous work. After the implementation, simple GUI framework will be done. From the middle of August, the project is expected to start debugging and testing phase. Finally, a report and Doxygen will be done in September.

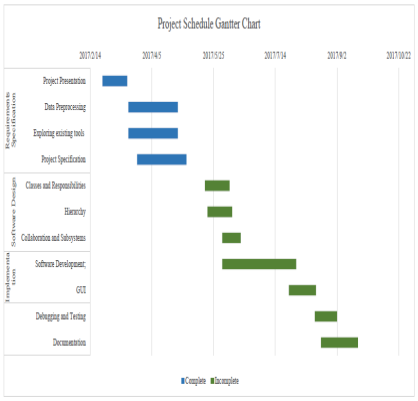

Figure 4: Gantt chart for project timeline

4.3 Risk Analysis

This part is about the potential risks which may happen when doing this project.

Figure mapping the analysis of these risks:The first risk is that the author may lack of the knowledge when carrying out the project. The probability is medium as the limited time author has been studying the computer science. The impact has been considered as high because the project will progress slowly and face obstacles without support of essential knowledge. So, the regular meeting with the supervisor is important to consult the difficulties encountered during the process. The second risk is that the whole project may be finished after the deadline. The possibility is considered medium due to the possibility of other risks. Lack of essential knowledge, problems in programming, or lack of project management skills may lead to the delay. The author should apply for delaying submission if this will happen in advance. The third risk identified is 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 fourth risk. It is classified as medium in terms of both possibility and impacts. Yet there are computer equipments on campus and the library open 24 hours so that the resources of university are available every day. Data loss is considered as the fifth risk. the possibility of this happening is low but will cause high impact to the project. To avoid this situation, using the application Github for regular backup is necessary. The last risk is lacking project management skills to implement the project. This is considered as medium possibility to happen with high impacts to the project. In this case, a strict plan following rules of Agile

software development methodology is important.

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 5: Risk Analysis Table

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 5 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;

The structure of the DataReader class is presented in figure .

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 .

Version Class

The Version class is another object class used to store information related to each version of 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 .

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 .

LemmaProcess Class

The LemmaProcess class is designed to generate lemma for each term. As shown in Figure , this class includes 3 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.

5.2 Visualisation generation

TranslationVisualisation Class

The translation generation stage comes after data reading and processing. The TranslationVisualisation 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;

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 library. 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 are 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 hierarchy of ConcordancePanel class is shown in Figure .

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. 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

VersionChoosenPanel Class

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

- Accept data from DataReader class;
- Initialize JPanel;
- Dispaly 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 .

6 Implementation

In this section we describe how the project is implemented in detail. The subsystems executed are data reading, visualization 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 [**?**]. 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 [**?**]. 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 6, the concordance visualisation involves several parts:

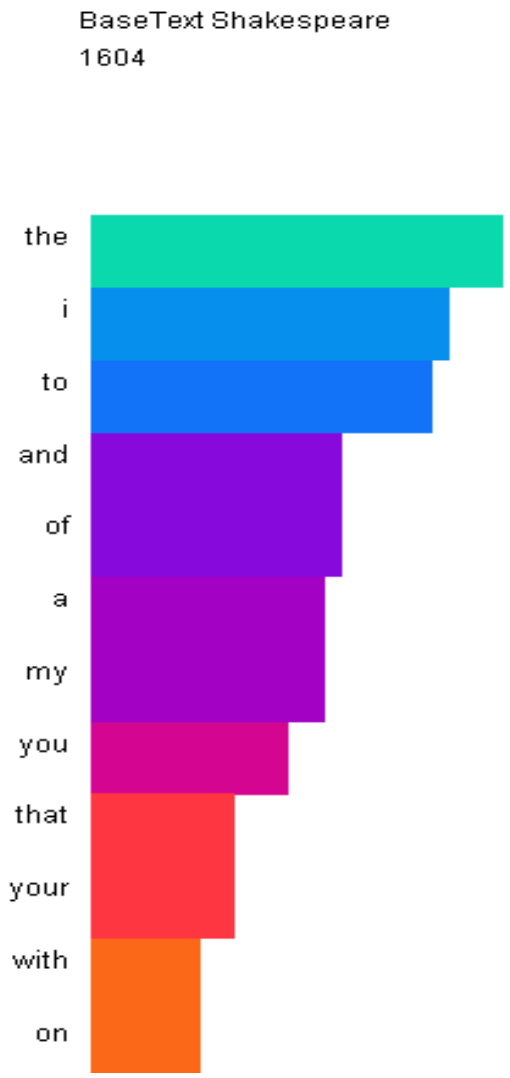

Figure 6: The Screen shot of one concordance in the visualisation

- **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 visualization 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):

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

  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 concordance visualization, a parallel view of all concordances is created. As shown in Figure 7, all versions of concordances are represented on the panel. During this stage, lines will be drawn to connect same terms. The comparison stage is done in concordancePanel class.

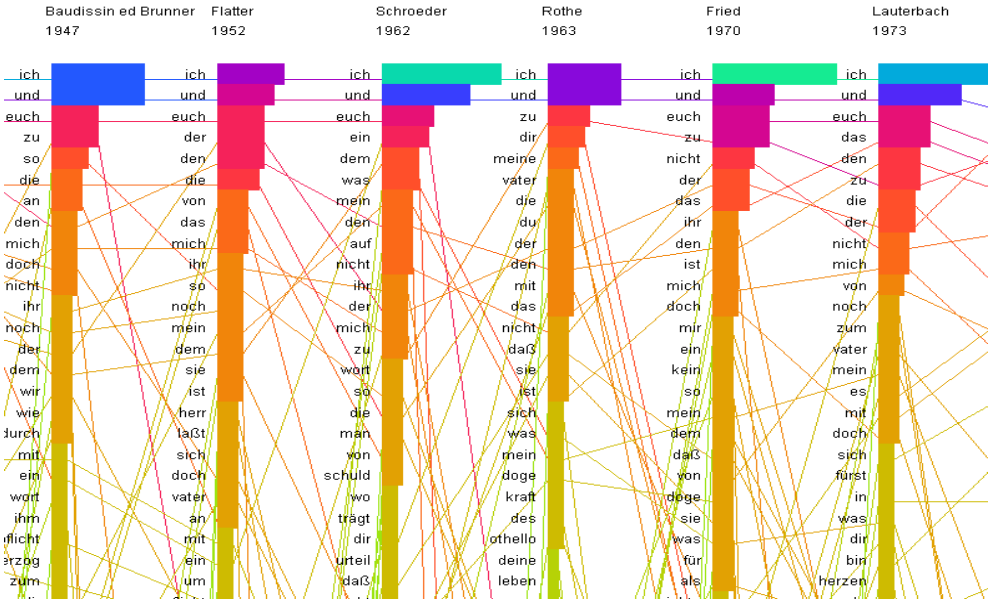

Figure 7: Parallel view of concordances

However, after this parallel visualization is being generated, an obvious problem appears: there is not enough space for all 16 concordances. So the solution is either to scale the panel,

or to select several versions showing one time. We have done both, which are introduced in the following section.

6.4 Zooming

Zooming in and out is a basic feature in the software which 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 8 displays the JSlider applied in the software.

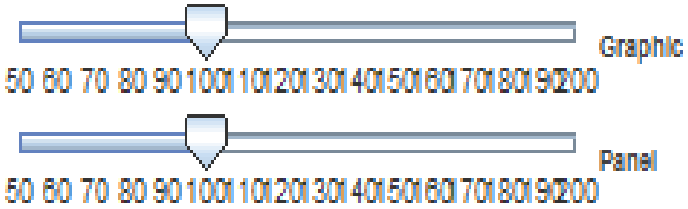

Figure 8: Sliders applied to zoom in and out the graphic and panel

- Obtain scale values from JSlider object and pass them to DataReader class.
- Recalculated 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 8, 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%. Figure 9 and Figure 10 show the zooming results for the visualisation.

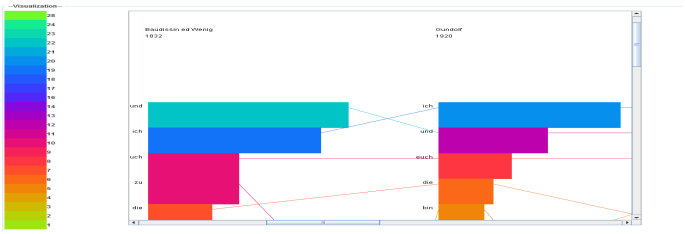

Figure 9:

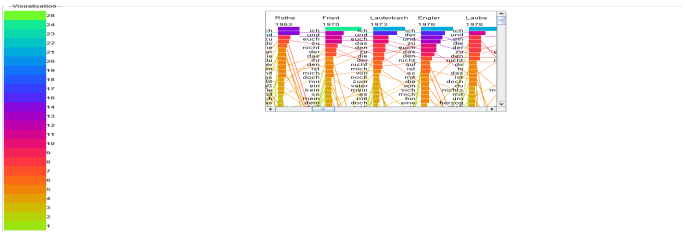

Figure 10:

6.5 Text Labels On and Off

When the scale values becoming smaller, the strings overlap. Hence a new desire appearing: hide strings on the visualisation. So that users can focus on the rectangles and colours only.

To implement this feature, a JButton is generated on the panel firstly. Also, "Text On" is set as default label displayed on the button. Secondly, event listener is added to the button. When button is clicked, the label "Text On" on the button will be switched to "Text Off" label. In the meantime, a boolean value which set "true" as default will change to "false", then being given to ConcordancePanel class. In the third step, a boolean value preset when drawing strings of terms will be switched equals to the boolean value passed in. if it is "true", then draw the strings, if it is "false" then not invoke the drawString() method. At last, repaint the graphic. Figure 11 is a screen shot when we turn off the text.

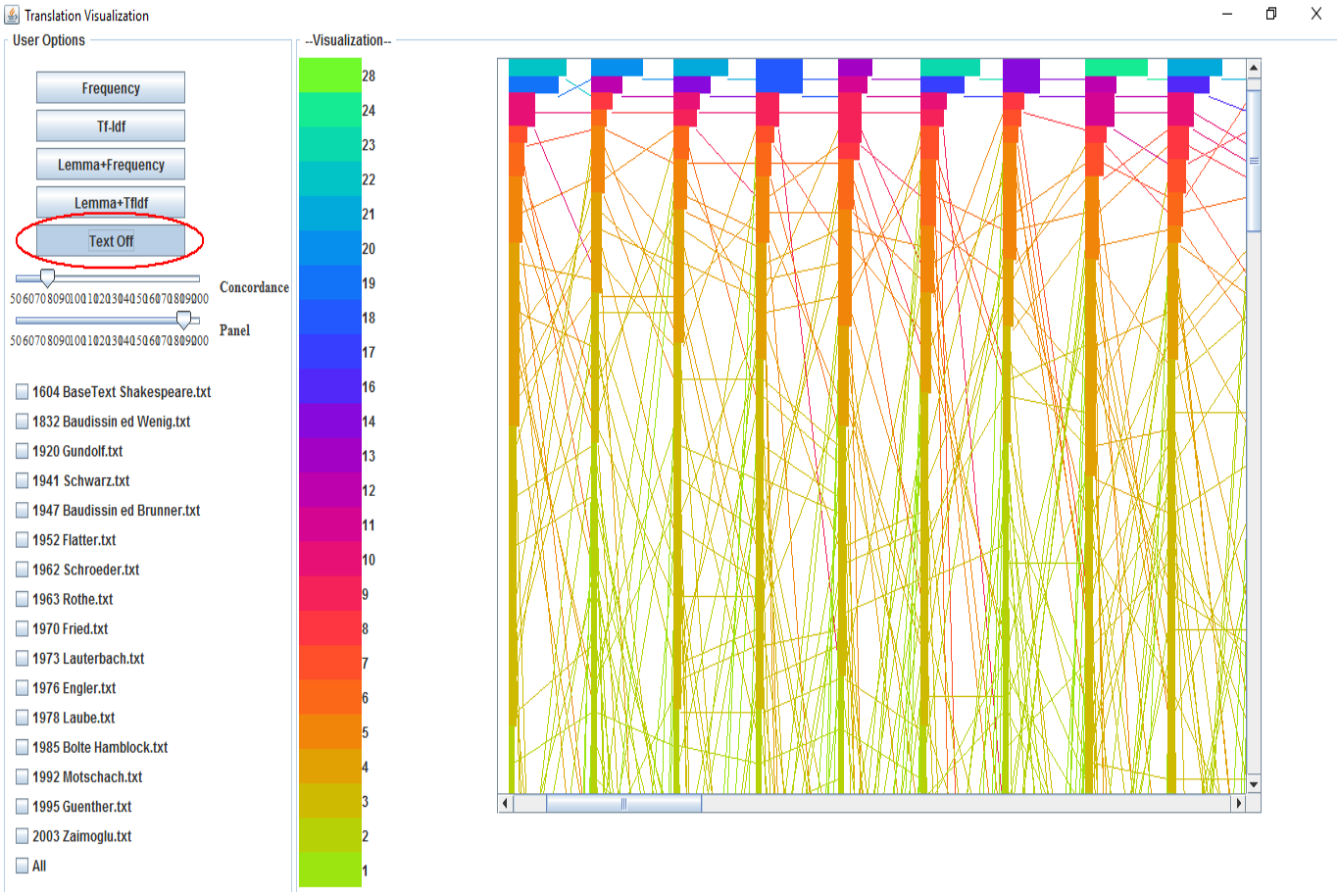

Figure 11:

6.6 Adding, Subtracting, Selecting Items

To render an user option feature for selecting several concordances displaying on the panel, a new class called VersionChoosenPanel is created. By interacting with this feature, not only can the user select which concordance to display in the visualization, but also the order of concordance displayed can be arranged. Figure 12 reveals the menu of version list can be selected.

The generation of version selection feature goes through the following steps:

- Generate a list of JCheckBox class to display the author name as the index.
- Add event listener for each JCheckBox object. So that the action of selection can be generated as an Object class.
- Change the selecting status of the index.

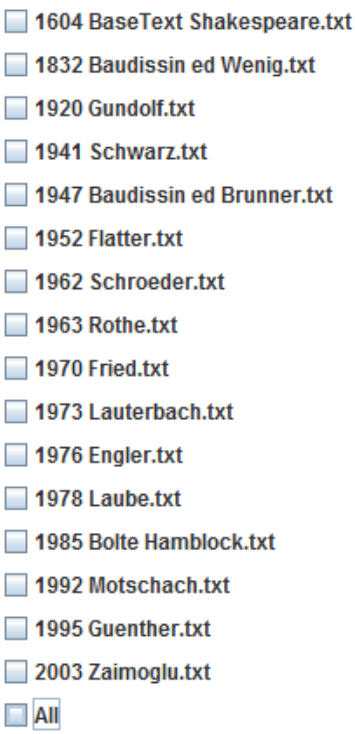

Figure 12: The index of version selection

- Generate new list of Version objects according to the events passed from JCheckBox ActionListener. Every time the user select a name in the index, a new list of Version objects will be generated and passed to ConcordancePanel class.
- Repaint the concordance visualisation. The ConcordancePanel will be repainted by invoking repaint() method.
- Add an option of "All" selection, which is responsible to display or hide all concordances as the original order.

Figure 13is a screen shot of selecting several versions of concordances to show on the visualisation. Further more, concordances are reordered on the visualisation, where the base text which supposed to be shown as the first version on the left, now being moved to the last one.

6.7 Interaction and Selection of Terms

On the ground that each concordance contains a large number of terms, highlight the term following users' options is desired. In order to provide an interactive features for terms, new

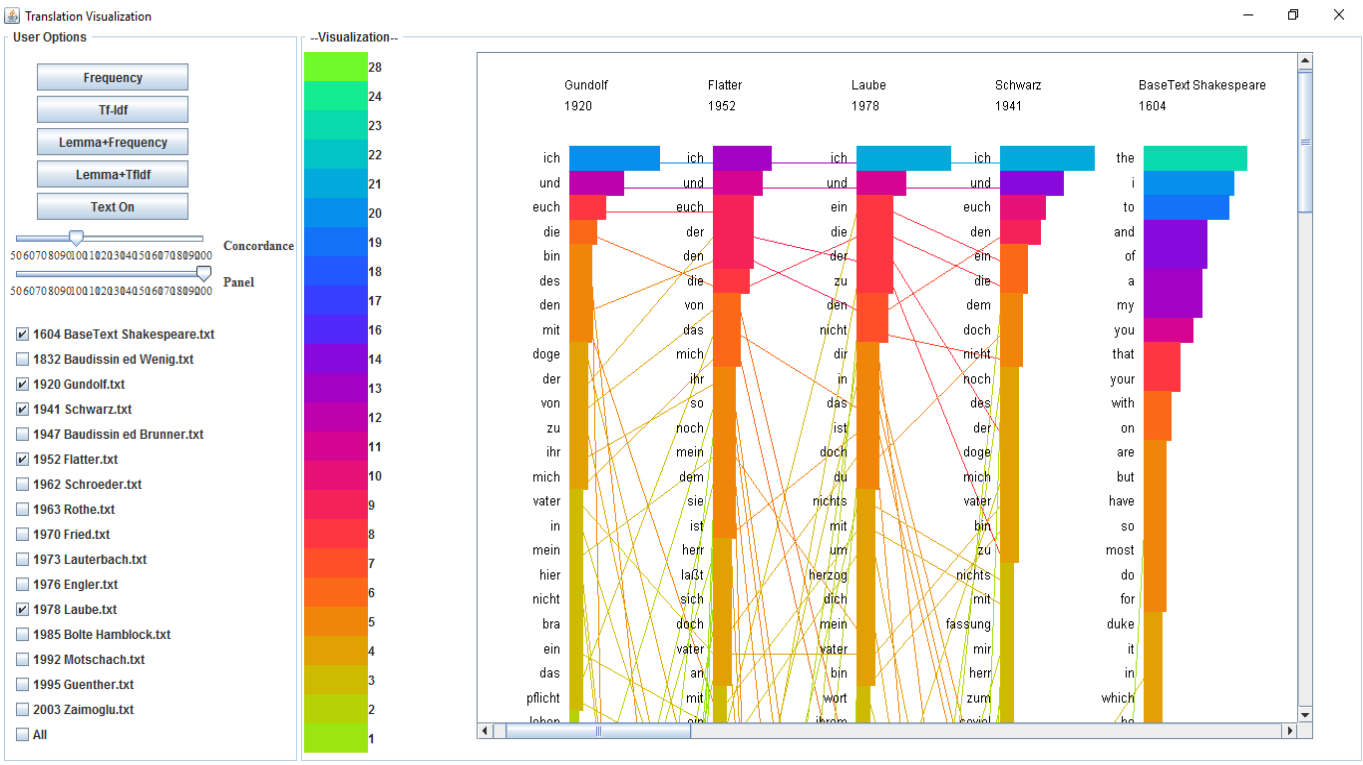

Figure 13: The screen shot of version selecting feature

feature are enable in the visualisation. Therefore a clear view of highlighting terms comes out.

This features are achieved by put into effect following phases:

- Obtain clicking point through getPoint() method in MouseEvent class.
- Calculate which item region the point belongs to. In this process, the regions of concordance and item are divided as illustrated in Figure **??**. As the value of each region can be achieved during data reading phase, the point passed from mouseClikcked() method can be used to identify which region the point belongs. Further more, the Item object of this block is singled out and returned.
- Identify the Item objects in other concordances sharing the same term. So that we get Item objects to be highlighted.
- Highlight the blocks of all singled out Item objects. In this step, lines are drawn to round the rectangles.

- Make other blocks transparent. The transparency values of other rectangles are set as semitransparent values by overwriting colour values of blocks.
- Overwritten the colour values of all lines to make sure lines connecting two highlighting items are highlighted as well, while other lines becoming semitransparent.
- Find out the translations of this term and highlight the blocks of them.

As a result, by clicking one single rectangle in the panel, the rectangle is highlighted and rounded by line while the colour of other rectangles become transparent. In the mean time, lines connecting same terms become highlighted by setting other lines transparent. See Figure 14

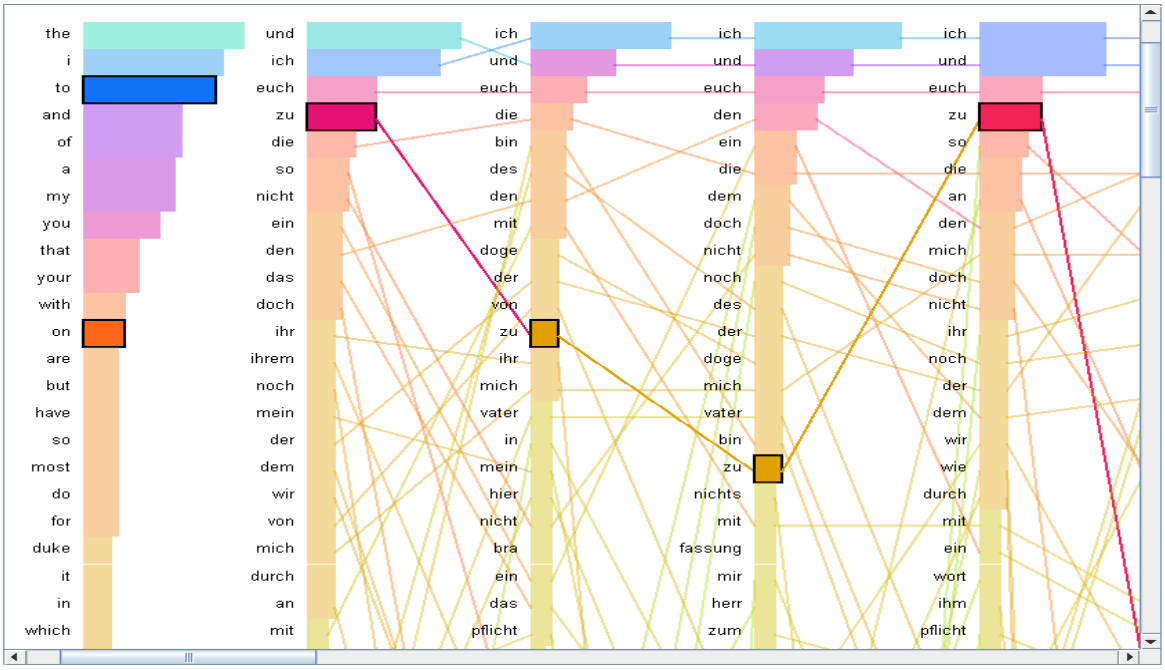

Figure 14:

6.8 Colour Mapping

Colour represents the occurrences of terms in each concordance. To demonstrate all colours in the visualisation, colour mapping is essential. In this project, we instantiate a ColorLegendPanel class to fulfill this feature. In addition, values of the frequency is displayed in the Colour Mapping view. Following is the process to implement this function:

- Achieve the colour value of each item from DataReader object. The detailed illustration of colour values can be retrieved in Data Reading chapter.
- Instantiate JLabel objects as the components representing colours.
- Add the JLabel objects to the panel.
- Display frequency values beside colour blocks.

The demo of Colour Mapping is shown in Figure **??**.

6.9 Interactive Color Legend

Apart from displaying data, colour mapping can be served as interactive visualisation. In this project, we add an feature so that user can interact with the colour legend. If the user click a label of colour 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 same frequency value. As illustrated in the Design chapter, an index of frequency values is generated in DataReader class. After data reading phase, we can access this list of frequency values through accessor method. By iterating all values in the list, the items are identified and returned to ConcordancePanel class. Then the list of Version objects are overwritten and the panel is repainted.

As a result, by clicking one colour block in colour legend, all items sharing same frequency, or colour, are highlighted using same methods of highlighting described in Interaction and Selection of Terms chapter. Figure **??** serves as a demo to illustrate this feature.

6.10 Lemmatisation

The lemma visualisation a significant feature in this project. Lemma is a linguistic term which described as the dictionary form of a word. Take the English word 'decide' for example: 'decide' is the lemma for 'decided', 'decides', 'deciding'. Accordingly, lemmatisation is the process to obtain the lemma for each word. To achieve the effect of lemma view, several steps is 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 being found and generate a new lemma index.
- Apply the lemmas into visualisation.

For this project, an inevitable dilemma is the limited resources of German lemma corpus. As illustrated in Data Characteristics chapter, there is no relevant German lemma corpus in this project when we start this project. Also, German, as an affected language, is difficult to lemmatise. It is more challenging to acquire this kind of corpus from other sources. During this process, we attempted two solutions: Treetagger and DeReWo, which will be explained in following sections.

6.10.1 TreeTagger

TreeTagger, developed by Helmut Schmid at the Institute for Computational Linguistics of the University of Stuttgart, is a tool for annotating text data and lemma information. It has been used to tag many languages including German. Figure **??** shows the Interface of TreeTagger. During applying this tool in the lemmatizing task of the project, we found this tool has following advantages:

- 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 concept in using software is to upload a .txt file, create a .txt file to store results, and lemmatise the data.

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 group of files at one time, which is not appropriate for project which needs to process large data sets.

We connected an domain expert, Dr. Tom Cheesman form the Mordern Language Center of Swansea University, to evaluate the results of the lemmatisation for TreeTagger. Appendix ? is the file of sample lemma we sent to Dr. Cheesman with the comments he sent back. Due to the low accuracy of the results for the data in this project, this solution is given up in the end.

6.10.2 DeReWo

DeReWo is a project done by Institut Fur Deutsche Sprache. This project aims at developing methods to create frequncy-based ranking lists of lemma 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 edit in Visual Studio Code. It also can be read from Java directly. Using this corpus, we successrully obtain 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 translation. Meanwhile, order of the lemma is kept as the same with their original text. 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 according lemmas and visualise the new results.

Figure **??** shows the outcome of the lemma visualisation.

6.11 Tf-Idf

Tf-Idf visualisation is an important feature in this project. As explained in Design chapter, the Tf-Idf value represents 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 apply the formula of the Tf-Idf into the codes. Equation **??** is used in this project to calculate the Tf-Idf value for each term [**?**]. 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. As shown in Equation **??**,
- Replace the frequency with Tf-Idf value.
- Visulise according to new results.

There are two visualisation generated using Tf-Idf value: one is to visualise data using Tf-Idf value and original words, as shown in Figure **??**; the other is to visualise using Tf-Idf value together with lemma data which is displayed in Figure **??**.

7 Evaluation

In this section, we provide the performance and feedback from a domain expert as evaluation.

7.1 Results

7.2 Domain Expert Feedback

7.2.1 Session 1

7.2.2 Session 2

8 Conclusion

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

9 Future Work

References

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

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

[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.

[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. Flow Visualization Lecture Part 4 Overview : Previous Lecture. pages 1–41.

[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.

[Tom et al., 2012] Tom, C., Kevin, F., and Stephan, T. (2012). VVV - Project.

[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

B JavaDoc of Project

pdf or latex