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 hava 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, a]. 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 in 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, exacting information from text data is becoming highly needed. In certain study area, studying the relationship between words, sentences and texts' structure may help researchers to understand important information hiding in the text. For example, in archaeologists' lab, analysing the text they found from historic site may help them understand the dates of files, events happened, or the host of the grave, even without knowing the meaning of the ancient language. Similarly in the archaeological industry, techniques in text data analysing 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 life status people in particular period of time.

The ways to analyse and present text data has 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. Same contents may appear in different style of expressions according to the purpose of texts. Also, to deal with these problems, text data can be analysed and represented from lexical, syntactic and semantic perspective [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 visualization 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 single document and a collection of documents. One such collection of documents is *Othello*.

Othello, as one of the greatest tragedies of Shakespeare's play, are translated more than 60 times in German by now [Geng et al., 2011].The College of Arts and Humanities at Swansea University has a collection of 57 different German translations of *Othello*. The time span of these translation is from 1766 to 2010. And there are also different genres such as poems, prose, as well as plays. Applying data visualization techniques to help to represent these text data will contribute to new research in Shakespeare's work studying, and visualization exploring. 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 assistance 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, styles [Alrehiely, 2014]. Based on the work of [Geng et al., 2015], [Alrehiely, 2014], and [**?**], we attempt to develop an interactive visualization system aims to allow our users to view, compare, and analyse tokens in each version. The visualization tool will be designed to assist in viewing 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, data of publication, there are three unique information of data are provided: the frequency of tokens, weight of tokens, and results from lemmatization for tokens.

The outcomes of the visualization system should be helpful to understand of the variation of word morphologies, varieties of text styles, as well as the complex features of German language. It also contributes to comprehend the dynamics of literature, the differences between language, 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.

However, there exists some challenges in this project. As a highly inflected language, German is featured as complex grammar structure and numerous compound words. With

the German Shakespeare text data in this project, several special problems are caused by antiquated language, and poetic orthography. The former means 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 normally written as "verloren", also can 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's no effective algorithms to recognise these forms. To find a solution for for these problems, some methods from Natural Language Processing may be applied, such as lemmatization.

Choosing this project for my dissertation was account of my interest in the field of data visualization and language analysis. The background of programming and language studying further my comprehension in 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, including literature review, introduction to existing systems, and data characteristics. Section 3 is the specification of the project which includes the features specification and technology choices to the software. 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, including the basic features of the software, and the enhancement features.In section 7, we provide the performance and feedback from a domain expert as evaluation. And section 8 draws a conclusion of this project and section 9 discusses the 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

The data is major part of visualization. Along with the user experience, data plays an important role as "driving factor" with respect to the choice and attributes of the visualization

method [Laramee, b]. 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 Digital Humanities researchers working on a project in [Tom et al., 2012]. To develop analytic tools and probe the translations in this corpus, the team has digitalized 32 translation versions, with the formats being normalized, texts being segmented, speech by speech and line by line. Also, the content of these 32 texts is a specific scene: the Act1, Scene 3 according to the English version of *Othello* play as base text . Based on this corpus, we select 16 text files of German translation versions to study and focus on the words analysis.

In this project, 17 text files are generated as corpus, among which 16 are German translation versions and 1 is the base text in English. All files are encoded as UTF-8 when converting from .docx file to .txt. file The number of words in each file are different according to the genres of text data (327 words at maximum, and 214 words at minimum). The data in each file has been preprocessed and organized line by line, and segmented speech by speech.

Since the project is programmed in Java and focus on the words, the .txt is an appropriate format to store the data. Choosing .txt file as the data set is owing to following reasons:

- The aim of this project focus on word processing, which require computer to read text literally, without applying complicated data processing techniques.
- Since the text data sets in the corpus are stored in .docx file which is difficult to read by Java directly, it is easy and safe to convert the texts into .txt file.
- There exits methods in Java API 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 exact more information from the text. And because the data set in each version is not large, the computer can calculate the essential features of 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 .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 **??**)

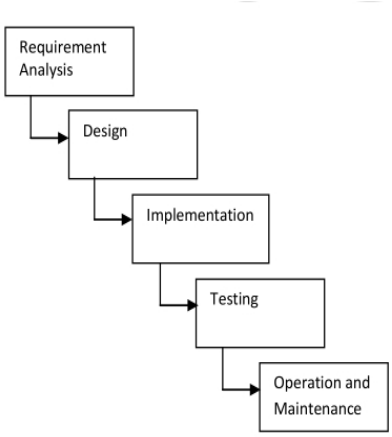

Figure 2: The 5 phases of Waterfall Model ([**?**])

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.

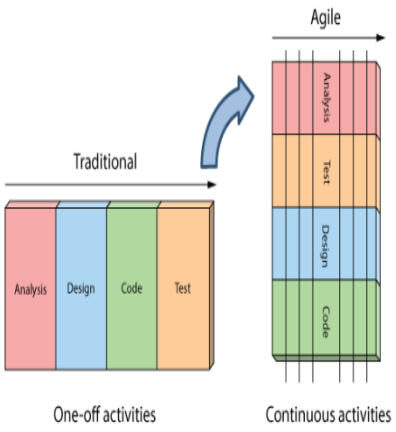

Figure 3: Comparison between Waterfall methodology and Agile methodology ([**?**])

4.2 Project Timetable

This section indicates time management for the project. These project separates into 5 phases [Laramee, b] 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.

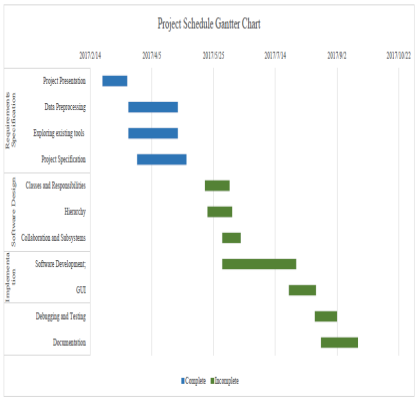

Figure 4: Gantt chart for project timeline

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

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,

Datareader Class

The DataReader class is one of the base class that is designed for reading and processing data from all 16 files. Then it parse to other objects to be stored and used. There are 5 mainly functions as followed:

- Read data from .txt files;
- Calculate term frequencies, point locations, color values;
- Parse 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 parse that to visualization parts;

The structure of the DataReader class is presented in figure .

Concordance Class

The Concordance 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 DataReader class. Then these values are stored into lists of concordance as a column. When the visualization being generated, these values will be used directly. The data can also be modified from accessors 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 concordance to store the concordance of this translation version. After all 16 texts being read and processed, there will be a list of Version objects generated. and These 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 frequecy 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 parse them back to DataReader class;

The algorithm of Tf-Idf value will be introduced in Implementation session. 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 parsed 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 Translation generation

TranslationVisualisation Class

The translation generation stage comes after data reading and processing. The TranslationVisualisation class is designed for accepting all data processed from data reading part and generating the visualization for software. This includes following stages:

- Accept data from DataReader class;
- Initialize all GUI components;
- Parse 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 inherit from Java AWT libraries.

ConcordancePanel Class

The ConcordancePanel is the main visualization panel in the software. It inherits the JPanel class which belongs to Java Swing library. It is designed to render a canvas drawing all parallel visualization of concordances. Data are parsed from visualization part and used to display visualization here. There are also a Mouse Click Listener in this class used to listen 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;
- Parse events data from event listeners;
- Recalculate data;
- Repaint the graphic;

The class hierarchy of ConcordancePanel class is shown in Figure .

ColorLegendPanel Class

The ColorLegendPanel class inherits from JPanel. This class renders a color map, where each color owns an event listener. The data parsed in this class are term frequencies, or Tf-Idf value, depending on user preference. Event listener is added to each color block to listen which block is clicked. Then the clicking data will be parsed 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;
- Parse events data to ConcordancePanel class;
- Display which version is selected;

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

6 Implementation

This section describes how the project is implemented in detail. it involves the implementation of the data reading, visualization rendering and the user options generating. Screen captures of the software interface 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 we needed, and store them into Java Arraylists. In this stage, the greatest challenge is to keep the data being flexible, as these data may be reprocessed and stored due to the change of events in other class. Hence, after the original data being read and stored, they can be set and got through accessor methods [**?**] for many times. In addtion to the flexibility, another difficulty in this stage is to generate various data values for each token data and store them appropriately. As discussed in Project Features section, the aim of software we designed is to present information about terms and provide an concordance view for each version. In this project, for each token, we calculate and sort frequencies, weights; compute color values; computed locations; generate rectangle information; create a group of translation and lemma values.

Java.io, which provides for system input and output through data streams, serialization and the file system [**?**], is the libraray used in this project. It serves as a data buffer and reader in this project. FileReader class, which extends the abstract calss "InputStreamReader", can be used to read character files which defaultly assuemed are in appropriate size. Since the volume of data in each document is not large, we generate FileReader class to access each text file. The other data reading class adopted is the BufferedReader class. It is available to read text data from character-input stream, buffer the data so as to provide for the efficient reading of strings, arrays and lines [**?**]. Java.util is another package imported in DataReader class. To store and access data, ArrayList, Hashtable, and Map of this library are applied. In addition, data structures such as JsonObject, JsonReader, and JsonArray in Javax.json library are applied to read data from 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 represents a version of German translations and contains essential data of the version. The Version object also has an accessor method to access the list of Concordance objects which contains all information of terms in the

concordance. For more detailed description of the Version class and Concordance class, please see the Design section.

6.2 Generating Concordances

6.3 Parallel View of Concordances

6.4 Adding, Subtracting, Selecting Concordances

6.5 Zooming

6.6 Text Labels On and Off

6.7 Interaction Selection of Terms

6.8 Color Mapping

6.9 Interactive Color Legend

6.10 Lemmatization

6.11 Tf-Idf

6.12

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

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

[Laramee, b] Laramee, R. S. Visualization , Introduction Lecture Part II About data.

[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