VISUALIZATION OF VERSION VARIATION

Majedah Mohammad Alrehiely

715803@swansea.ac.uk

October 2014

Project Dissertation document submitted to the University of Wales, Swansea, in fulfillment of the requirements for the Degree of Master of Science Department of Computer Science;

Swansea University
Prifysgol Abertawe

Department of Computer Science

Swansea University

Declaration

This work has not previously been accepted in substance for any degree and is not being currently submitted for any degree.

Date:

Signed:

Statement 1

This dissertation is being submitted in partial fulfillment of the requirements for the degree of a MSc in Advanced Computer Science.

Date:

Signed:

Statement 2

This dissertation is the result of my own independent work/investigation, except where otherwise stated. Other sources are specifically acknowledged by clear cross referencing to author, work, and pages using the bibliography/references. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure of this dissertation and the degree examination as a whole.

Date:

Signed:

Statement 3

I hereby give consent for my dissertation to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Date:

Signed:

Abstract

Over the past few years, the size of data has increased dramatically. Big data sets are an issue in certain domains. There are a huge number of studies that have taken place to address this issue. Visualizing the data is an effective way for data representation due to the fact that we use our sight as one of the main senses to understand information. In this project, we concentrate on the visualization of textual data, which is in demand these days. A group of researchers in the College of Art and Humanities at Swansea University has collected different German translations of Shakespeare's play, Othello, which have evolved over a long period of time. The aim of their research is to study the variations between these different translations, and understand how the translations have evolved throughout this period of time. Furthermore, how they vary from one author translator to another. Our goal in the visualization of version variation project is to develop an interactive visualization system that applies effective visualization techniques in such a way that helps the researchers in the College of Art and Humanities to analyze, explore and identify the variations of theses large textual data.

Acknowledgements

Mostly, I wish to express my gratitude and appreciation to my supervisor Dr. Robert Laramee for his guidance, support and valuable feedback. I have really appreciated his patience and his positive criticism during this project. Furthermore, I would like to thank him for his effort and motivation that encouraged me to improve my research and programming skills. I would like to express my deepest gratitude to my family for their endless encouragement and support throughout my life. In addition to these people, I would like to express my appreciation to my sponsor, the Ministry of Higher Education of Saudi Arabia, for its financial support. I thank James Walker, a PhD candidate, for his helpful information about the visualization of the project and the Qt GUI library. I thank Kevin Flanagan for his valuable discussion and explanation of the project's application domain. My sincere thanks to Susan James who assisted me in proofreading my dissertation. Finally, special thanks and appreciation go to my extended family and friends for their continued encouragement and support.

Contents

List of Figures

List of Tables

1 Introduction

During the past few years, there has been a dramatic increase in the size of data. Dealing with theses huge data sets in their various formats has created obstacles in certain domains. The need for presenting the data in a graphical representational format has arisen to deal with big data sets. Transforming data into a graphical format provides a better representation due to the fact that we use our sight as one of the main senses to understand information.

Data visualization is a modern field which is becoming one of the fundamental requirements to presenting, exploring and analyzing big data sets of various formats in different fields. Data visualization is defined as "the communication of information using graphical representations". It is important for several studies such as decision making assistance and human interpretation [Ward et al., 2010]. It is also used for many different purposes and could be applied to different types of data.

In diverse domains such as academic, business, medical, political and commercial, the number of documents and information are getting larger over time. This has made it necessary for a variety of studies to take place to find a suitable solution to deal with this problem. According to [Ward et al., 2010], data visualization provides massive and helpful assistance in analyzing and exploring textual data. Developing an interactive visualization system to help in the process of data exploration, analyzing and retrieving is a necessity these days. In our project we are going to focus mainly on the field of textual data visualization.

1.1 Project Scope and Objective

William Shakespeare is one of the most famous poets and an English playwright. His works and plays have been translated and retranslated many times into a variety of foreign

languages, and in many languages, there are several different versions of translations of each work. These versions of the translated plays have evolved over many decades [Geng et al., 2013a]. Studying the changes and the way these translations differ is a recent important topic needed for studying the cultural background in that period. Therefore, the studying of version variation of German translations of Shakespeare's Othello play was a new and interesting research topic for the researchers in the College of Arts and Humanities at Swansea University. The reason why this topic is important for them is because of their aim to understand how the cultural changes have evolved over space and time, and studying the variations of the translations will provide assistance in this study. In addition to this reason, this type of study will contribute and enable new research, new teaching and new learning [Cheesman et al., 2012].

However, in our project we are working on the data provided by the College of Arts and Humanities at Swansea University. A team of researchers in this College has worked on collecting a group of about fifty-five different translated versions of Shakespeare's play, Othello. However, only a sub-collection of these translated versions has been converted into a digital form as a parallel corpus [Geng et al., 2013a].

The main objective of this project is to design and develop an interactive visualization system for text type data. The user of this system is Dr. Tom Cheesman, a reader in German in the College of Arts and Humanities, and the principal investigator on the "Version Variation Visualisation" project [Cheesman et al., 2012]. The purpose of our visualization system is to make the user able to view, compare and analyze the translations corpus by providing an effective type of visualization techniques to represent the data. The visualization will be designed in such a way as to assist in identifying the variations and help them to extract the required information from the visualized data. The developed interactive visualization system will involve multiple document visualization that applies parallel comparison between different documents.

Choosing this project for my dissertation was on account of my interest in the field of data visualization in general and of my interest in developing an interactive system that can be used to effectively deal with and represent large textual data in an expressive format. Developing a project for this purpose is becoming extremely important and has a huge significance due to the rapid growth of the volume of information and the huge increase of the size of data sets.

The project that we aim to develop is significant and valuable for the researchers in the fields of linguistics and translation, and for those in field of Art and Humanities. We aim to design and build a visual interface system that will enable the users to find out the essential information about each version of translations, such as the author, date, genre, description and other related information, and explore the content of that version of translation, compare it with other versions and identify how it differs from the others by looking at the visualization made.

The information extracted from the graphical representation made by visualizing the data should be helpful in understanding the role of world culture, cross cultural dynamics and the perception of the history of translating cultures. It will also assist in finding out how a piece of work would be delivered into other foreign languages and how the same piece of work will vary when retranslated into the same language over space and time [Cheesman et al., 2012]. Moreover, developing a system to visualize the translations of Shakespeare's Othello will lead to developing more effective visualization tools to represent other large data of textual form including books, newspapers, journal articles, scripts and many others to extract and get the required information.

The remainder of this document is organised as follows: The second section presents the background, which discusses the work related to my project and the previous system.

The third section discusses the application domain and data characteristics in detail. The fourth section presents the project's feature specification and requirements and the choices of technology. The fifth section examines the use of the existing visualization tools and their limitation. Section six discusses the project design. The seventh section reviews the plan and timeline. Section eight demonstrates the project implementation and the result and finally the conclusion.

2 Background

The background section includes two main sub-sections. The first one presents the literature work that was most related to our project. It includes the sources and research papers that the project relies on. The second sub-section contains a detailed description of the previous system. The background section has been updated from the final report of my project.

2.1 Related Work

In this section, we will discuss the main literature that was most related to our project, and on which it basically relies. We will present the main concept of each of these researches, application domain and dataset, and the applied visualization techniques. We also mention other, closely related work.

Visualizing Email Content: Portraying Relationships from Conversational Histories

The objective of Themail [Viegas et al., 2006] is to design a tool for visualizing the content of the archives of individual emails, throughout a period of time. It produces a visualization of textual data gained from processed email files, starting with email archives, and shows the relationships that will be produced according to the analysis of email archive

content.

The Themail system provides various visualization techniques. It includes an email contacts list direct visualization [1], and a Visualization panel, which displays the history of conversations between the owner of the input email and user's selected contact in parallel tags, which provides direct visualization, in addition to the use of coloured circles that provides indirect visualization to show the sent and received email messages during every month [Viegas et al., 2006].

In Themail Visualization System [Viegas et al., 2006], the application domain is an individual email messages. Conversation Map [Sack, 2001] is the most related literature to Themail [Viegas et al., 2006].

Figure 1: Visualizing Email Content: Portraying Relationships from Conversational Histories. This figure views a screenshot of Themail Visualization System [Viegas et al., 2006]. It shows a user"s email conversations with a friend over 18 months. Image credit: [Viegas et al., 2006].

Mapping Text with Phrase Nets

Mapping Text with Phrase Nets [van Ham et al., 2009] paper presents a new visualization technique for unstructured text mapping, which is named Phrase Nets. The aim of the

[1] Direct visualization means that the actual text appears in the visualization where it does not appear in the indirect visualization.

Phrase Nets visualization is to look for a balance in analysis and display of the targeted text. The analysis process is based on "phrase", which is used as analysis unit. The 'phrase' is the relationship between words within the text and it could be defined by pattern matching or by syntactic analysis.

The application domain for Phrase Nets [van Ham et al., 2009] is unstructured large textual data. It has been applied to books, poems and novels such as "Pride and Prejudice". The data goes through analysis and pattern matching processes before the visualizing process. However, the visualization techniques applied in the Phrase Nets [van Ham et al., 2009] is phrase nets direct visualization. A simplified version of Phrase Nets has been deployed to the Many Eyes web site [IBM, nd]. Phrase Nets [van Ham et al., 2009] was based on the idea of the "Semantic Net" [Shapiro, 1971] of artificial intelligence and DocuBurst visualization [Collins et al., 2009a].

DocuBurst: Visualizing Document Content using Language Structure

The main concept of DocuBurst [Collins et al., 2009a] is to develop a visualization tool that creates text summaries for a large textual data set. It provides useful interactive options, which give the user the ability to decide which text of interest to visualize. This visualization system visualizes document content according to the annotations of "Is-A" noun and verb hierarchies of Wordnet [Fellbaum, 1998], which can assist in the comparisons between documents [Collins et al., 2009a].

The main visualization techniques developed by DocuBurst [Collins et al., 2009a] are the visualization of the Generalized Fisheye View. The Fisheye View involves a cumulative view and single-node view and it provides direct and indirect visualization. Another technique applied is Dynamic Legend indirect visualization. In addition, the Detailed View of the input text, which is direct visualization [Collins et al., 2009a].

DocuBurst system [Collins et al., 2009a] is designed to produce informative visual-

Figure 2: DocuBurst: Visualizing Document Content using Language Structure.DocuBurst visualization [Collins et al., 2009a] using science textbook data. The root is the word “”idea””. Image credit: [Collins et al., 2009a].

ization for data with large textual content for example, books and electronic resources and any other large textual data, and is for multiple documents visualization. The most relative work to the DocuBurst system [Collins et al., 2009a] are Generalized Fisheye Views [Furnas, 1986]; Treejuxtaposer, Scalable tree comparison using focus+context with guaranteed visibility [Munzner et al., 2003] and WordNet, an electronic lexical database [Fellbaum, 1998].

Parallel Tag Clouds to Explore and Analyze Faceted Text Corpora

Parallel Tag Clouds visualization (PTC) [Collins et al., 2009b] is a novel approach for visualizing the variations between different large text corpora. The visualization methods applied in [Collins et al., 2009b] system are the Parallel tag clouds direct visualization, Tooltip (direct), Document Browser indirect visualization and Data-Rich Tooltip, which provides both direct and indirect visualization [Collins et al., 2009b]. PTC [Collins et al., 2009b] is applied to a domain that consists of a collection of more than six hundred thousand documents of the United States Circuit Court decisions. The dates of these documents cover a period of fifty years. However, only 13 documents from that collection are visualized by the PTC system.

Parallel tag cloud system [Collins et al., 2009b] is based on Themail Visualization System [Viegas et al., 2006], which basically aims to discover significant words from email messages and then visualizes them as tags in parallel columns along a time axis. The similarity between the two systems is in the concept. They are both looking for significant words within a corpus in order to create a graphical representation of these words in parallel columns. However, PTC provides an additional important feature that identifies the presence or absence of a selected word. It shows precisely how frequently a specific word is used within the visualized text [Collins et al., 2009b].

Figure 3: Parallel Tag Clouds to Explore and Analyze Faceted Text Corpora [Collins et al., 2009b]. This shows the significant words in each document in parallel. Colours are used to distinguish between high and low significant words. Image credit: [Collins et al., 2009b].

SparkClouds: Visualizing Trends in Tag Clouds

The SparkClouds paper [Lee et al., 2010] introduces SparkClouds visualization, which is a new class of the typical tag cloud visualization. SparkClouds visualization is a combination of Sparklines and the basic features of typical tag cloud. This is to transfer the changes across several tag clouds. However, Sparklines are line graphs where the axes are implicit, rather than being drawn explicitly.

It also introduces a guided study, which aims to perform a comparison between the proposed SparkClouds visualization, Parallel Tag Clouds (PTCs) visualization [Collins et al., 2009b], the multiple line graphs visualization and the visualization of stacked bar

Figure 4: SparkClouds: Visualizing Trends in Tag Clouds. SparkClouds Visualization System [Lee et al., 2010]: shows the top (25) words in a series. Image credit: [Lee et al., 2010].

charts. This comparison of these four types of visualization is performed in term of accuracy and speed in supporting specific tasks. The tasks are topic trends, specific data and providing an overview.

The visualization techniques performed in [Lee et al., 2010] are SparkCloud and Parallel Tag Cloud direct visualization and Multiple Line Graph and Stacked Bar Chart, which both provide direct and indirect visualization [Lee et al., 2010]. The dataset used in this study consists of five sets of typical tag clouds. The data of these tag clouds are the most frequent seventy-five words selected from the first chapter of five different books. The most related previous literature are Parallel Tag Clouds (PTCs) [Collins et al., 2009b] and the Sparklines [Tufte, 2006] since it incorporates theses two visualizations.

ParallelTopics: A Probabilistic Approach to Exploring Document Collections

The concept of the ParallelTopics [Dou et al., 2011] research paper is to develop a technique for analyzing and summarizing large text corpora. It presents the ParallelTopics Visualization System, which is an interactive analytical visual system. The proposed system is an integration of an interactive visualization and a probabilistic topic model. ParallelTopics uses the topic model to provide summaries of text corpora and to intro-

duce the possible distribution of each individual document across the corpora. The aim of ParallelTopics [Dou et al., 2011] is to detect the topics that are related to a document of interest and how important each of these topics is to that document [Dou et al., 2011].

ParallelTopics involves text processing before the visualization and it requires user interaction to answer questions for topic extraction and other tasks. The visualization techniques applied in ParallelTopics [Dou et al., 2011] are Topic Cloud which presents the extracted topics as direct tag clouds visualization, Document Distribution, Document Scatterplot and Temporal View indirect visualization, and it includes actual text and Detailed Text View direct visualization in addition to View Coordination and Interactions, which provides direct and indirect visualization [Dou et al., 2011].

The application domain for ParallelTopics consists of two text corpora. The first is composed of a collection of proposals that were favoured by the National Science Foundation, and the second is a collection of the publications of the period between 2006 and 2010 in the IEEE VAST proceedings [Dou et al., 2011]. The most related previous work to ParallelTopics [Dou et al., 2011] is TIARA, the visual text analysis system [Wei et al., 2010]. ParallelTopics [Dou et al., 2011] adds a description of topic evolution over a period of time and it provides documents characteristics according to their topical distribution, while TIARA [Wei et al., 2010] provides a less clear relationship between documents and extracted topics [Dou et al., 2011].

Hierarchical Topics: Visually Exploring Large Text Collections Using Topic Hierarchies

The Hierarchical Topics Visualization System [Dou et al., 2013] introduces a visualization of a big number of topics either titles or subtitles, which are contained in huge textual collections. The visualization techniques used by this visualization system are tree indirect visualization, ThemeRivers indirect visualization and the actual text direct visualization.

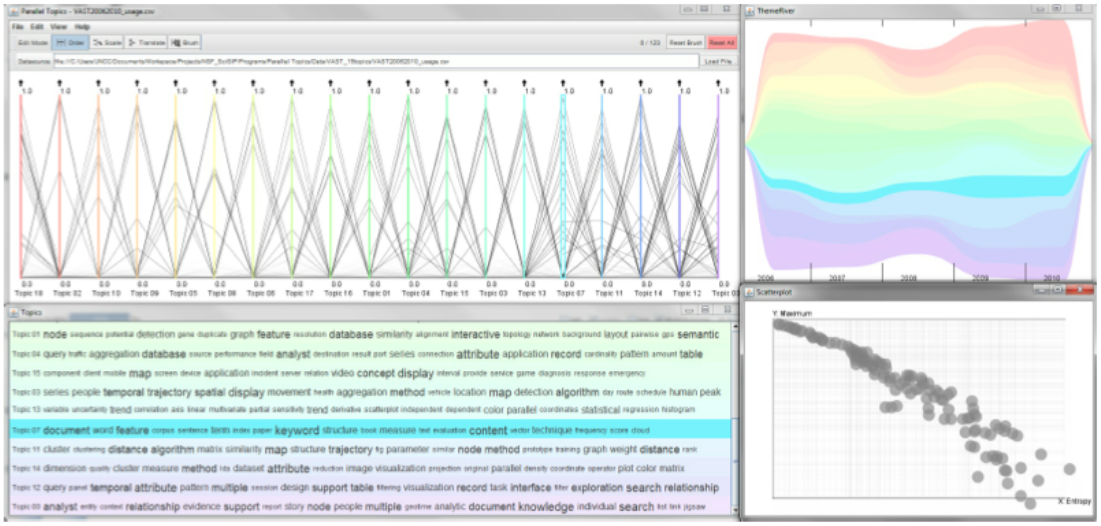


Figure 5: ParallelTopics: A Probabilistic Approach to Exploring Document Collections. Overview of ParallelTopics Visualization System [Dou et al., 2011]: Document Distribution view top left, Temporal view top right, Topic Cloud bottom left and Document Scatterplot bottom right [Dou et al., 2011]. Image credit: [Dou et al., 2011].

The application domain is a big textual data set of CNN news articles. The visualization system proposed by [Dou et al., 2013] can visualize a group of 2453 different news articles. The system is based on many previous systems including TIARA, the visual text analysis system [Wei et al., 2010]; (hLDA) hierarchical topic model [Griffiths et al., 2004] and LeadLine, a method to detect "Bursts" from topic streams [Wenwen Dou and Xiaoyu Wang and Drew Skau and William Ribarsky and Michelle X. Zhou, 2012].

The main feature in the hierarchical topics visualization tool [Dou et al., 2013] is that it has the ability to visualize large numbers of documents where the previous systems (hLDA) [Griffiths et al., 2004]; TIARA [Wei et al., 2010] and LeadLine [Wenwen Dou and Xiaoyu Wang and Drew Skau and William Ribarsky and Michelle X. Zhou, 2012] are limited to a smaller number of documents. However, the (hLDA) topic hierarchies visualization [Griffiths et al., 2004] is a based fixed size structure were the number of topics is predefined, while the visualization in [Dou et al., 2013] system is built on a multi level hierarchical structure with the number of topics of user selection.

Visualizing Translation Variation: Shakespeare's Othello

The purpose of Visualizing Translation Variation [Geng et al., 2011] is to design and develop a visualization system that can be used to interactively view, explore and analyze the differences between the German translations of Shakespeare's play, Othello. The application domain of [Geng et al., 2011] is a collection of German translations of Othello. Different visualization techniques are applied to get an appropriate visualization in this system. These techniques involve Parallel coordinates, which produce both direct and indirect visualization in addition to Tree map and DOI-tree direct visualization and the view of Detailed text direct visualization [Geng et al., 2011].

The system [Geng et al., 2011] visualizes eight different documents of the play. The related work to [Geng et al., 2011] is the Parallel Tag Clouds PTC [Collins et al., 2009b]. The [Geng et al., 2011] visualization system adds a feature that enables the user to brush any number of words to present them in the parallel tag cloud instead of presenting only one single word, as in PTC [Collins et al., 2009b] system.

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

The main concept of [Geng et al., 2013a] is to design an interactive system for visualizing version variations. The system is used to view and compare different translations based on analyzing segments variations [Geng et al., 2013a]. The ShakerVis visualization tool [Geng et al., 2013a] applies several visualization techniques. Shakervis performs Parallel Coordinate visualization, Scatterplot View and Heat Map visualization, all provide indirect visualization. On the other hand, the system includes Document Control Panel and Actual Text visualization, which are direct visualization. In the ShakerVis visualization tool, the application domain consists of a group of different German versions of Othello. However, only small amount of the text of eight translations are visualized [Geng et al., 2013a].

The previous system on which ShakerVis is built is the "*version variation visualization*" web tool [Cheesman et al., 2012]. In addition to the VVV web tool, ShakerVis is based on visualizing translation variation: Shakespeare's Othello [Geng et al., 2011]. The system proposed in [Geng et al., 2011] provides a visualization that shows how each term would differ in each translated text, whereas in ShakerVis it shows a visualization of segments or speeches of the translations. Moreover, ShakerVis applies the Heat Map visualization and the Scatterplot view visualization.

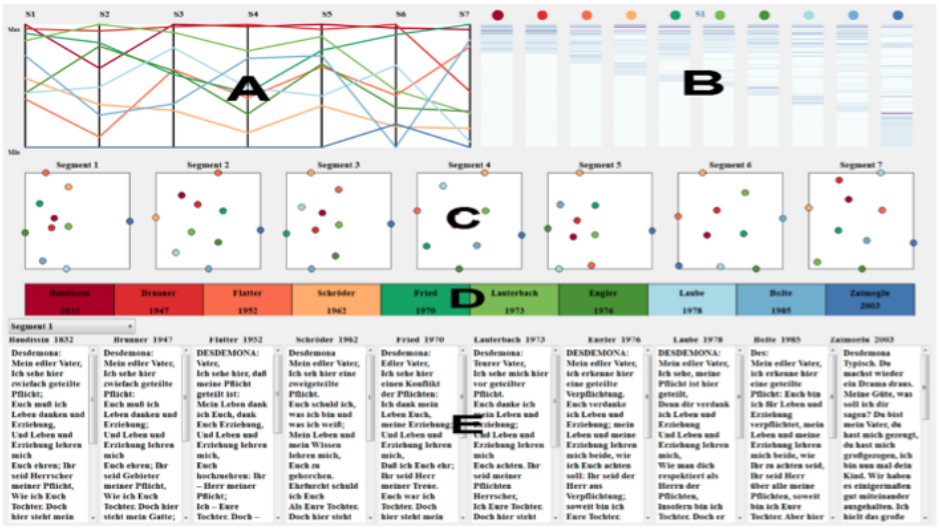


Figure 6: An overview of ShakerVis Visualization System [Geng et al., 2013a]. (A) Parallel Coordinates View (B) Heat Map visualization. (C) A Scatterplot View (D) The Document Control Panel (E) The Actual Text. Image credit: [Geng et al., 2013a].

Visualizing Translation Variation of Shakespeare's Othello: A Survey of Text Visualization and Analysis Tools

The objective of [Geng et al., 2013b] research paper is to study and survey the freely available text visualization tools that are suitable for visualizing the different translations of Shakespeare's Othello and compare these different tools. This research paper presents a specialized comparison and it performs two types of them. The first one is the comparison of research prototypes, which aims to compare the visualization techniques used. The second one is the comparison of the freely available software, which is based on comparing the user interaction options each of theses tools provides.

It proposes using a visualization method, which is designed for producing a special type of analysis. This way of analyzing the data makes the user able to zoom into the document and precisely to the segment selected by the user. The zoom-in option assists in providing in-depth reading.

The supplementary material related to the [Geng et al., 2013b] paper, presents the investigation applied to the state-of-the-art text visualizations from two different perpectives: the research prototypes for visualizing textual data and it involves the: Parallel Tag Clouds; ThemeRiver; TextArc; Document Contrast Diagram; SparkClouds and DocuBurst. The second is the free off-the-shelf text visualization tools, and involves the: ManyEyes visualization tool; Tagline Generator and TokenX. These visualization software apply various types of text visualization techniques [Geng et al., 2013b].

ConVis: A Visual Text Analytic System for Exploring Blog Conversations

The ConVis research paper [Hoque and Carenini, 2014] presents a novel visualization technique designed to explore and analyze blog conversations. The proposed approach integrates new data mining methods with interactive visualization to provide effective exploration for large textual data [Hoque and Carenini, 2014]. The main goal of the ConVis [Hoque and Carenini, 2014] is to assist the user in identifying and extracting the topics, views and opinions in the blog conversations.

It applies the following visualization techniques: The Thread Overview indirect visualization, The Facet Overview direct visualization and the Conversation View direct visualization that displays the text of the actual conversation. The application domain in ConVis [Hoque and Carenini, 2014] mainly consists of two sources, which are totally different blogs. These blogs are Slashdot [Slashdot, nd], a blog site for technology news and Daily Kos, a blog site for political analysis [Daily Kos, nd]. The previous work,

which is more related to ConVis [Hoque and Carenini, 2014], is Pivotpaths [Dörk et al., 2012] since they both provide Facet Overview visualization and allow the user to view and access the relationships between the facets.

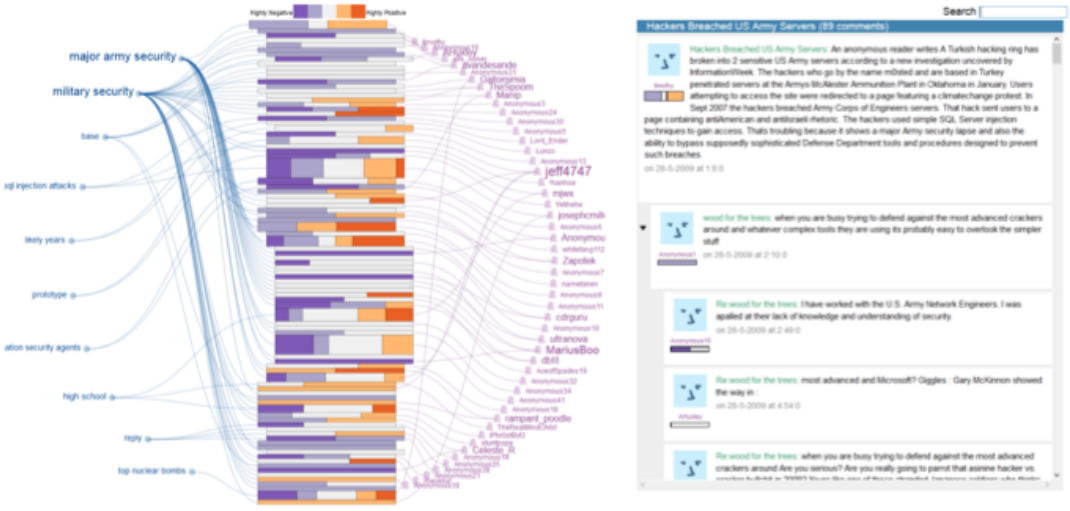


Figure 7: ConVis: A Visual Text Analytic System for Exploring Blog Conversations. A screenshot of ConVis visualization system [Hoque and Carenini, 2014]: This shows the visualization result of exploring blog conversation. The Thread Overview visualization is presented in the middle; Facet Overview visualization views the topic and the authors around the Thread Overview; and on the right the Conversation View that displays the actual conversation. Image credit: [Hoque and Carenini, 2014].

Table 1: Visualization Techniques Classification for the Literature Review: This table shows the classification of the literature review according to the type and the visualization techniques used.

	No of Documents			
Direct	1-9	10-19	20-29	30 and more
	[Geng et al., 2013a]: (DTV).	[Collins et al., 2009b]: (PTC) (TT) (DTT) .		[Dou et al., 2013]: (DTV).
	[Geng et al., 2011]: (PC) (DTV) (TM) (DOI-tree).			[Viegas et al., 2006] (EL)
	[Lee et al., 2010]: (PTC) (SC) (MLG) (SBC)			[Dou et al., 2011]: (TC) (DTV) (VCI)
	[van Ham et al., 2009]: (PN)			[Hoque and Carenini, 2014]: (FO) (CV)
Indirect	[Geng et al., 2013a]: (PC) (HM) (SP).	[Collins et al., 2009b]: (DB) (DTT).		[Dou et al., 2013]: (TM) (THT)
	[Geng et al., 2011]: (PC)			[Viegas et al., 2006]: (CC)
				[Collins et al., 2009a]: (FE)
	[Lee et al., 2010]: (MLG) (SBC)			[Dou et al., 2011]: (DD) (SP) (TMP) (VCI)
				[Hoque and Carenini, 2014]: (TO)

Table 2: The key entries for table 1 (The shortcut for the visualization techniques)

(CC)	Color Circles	(PC)	Parallel Coordinate
(DB)	Document Browser	(PTC)	Parallel Tag Clouds
(DTV)	Detailed Text View	(SP)	Scatterplot view
(DTT)	Data-Rich Tool Tip	(THR)	ThemeRiver
(EL)	Email List View	(TT)	Tool Tip
(FE)	Fisheye view	(TM)	Tree Map
(HM)	Heat Map	(SP)	Spark Clouds
(MLG)	Multi Line Graph	(SBC)	Stacked Bar Chart
(PN)	Phrase Nets	(TC)	Topic Clouds
(DD)	Document Distribution	(TMP)	Temporal View
(VCI)	View Coordination and Interaction	(TO)	Thread Overview
(FO)	Facets Overview	(CV)	Conversation View

2.2 Previous Systems

In the previous systems section we are going to discuss the VVV visualization tool, which stands for the Version Variation Visualization. The VVV tool is an interactive web visualization tool. The concept of the VVV project was produced by Dr. Tom Cheesman, a reader in German, Languages, Translation and Communication in the College of Arts and Humanities at Swansea University [Cheesman et al., 2012].

The VVV tool was developed to find out the variations between various translations of the same literary work. The translations are different versions of translations in one language or in different languages. It has an interactive visual interface, which is supported by an analysis tool. The application domain for this project is an experimental corpus of various translations of the Othello play . The system was launched in September 2012 [Cheesman et al., 2012].

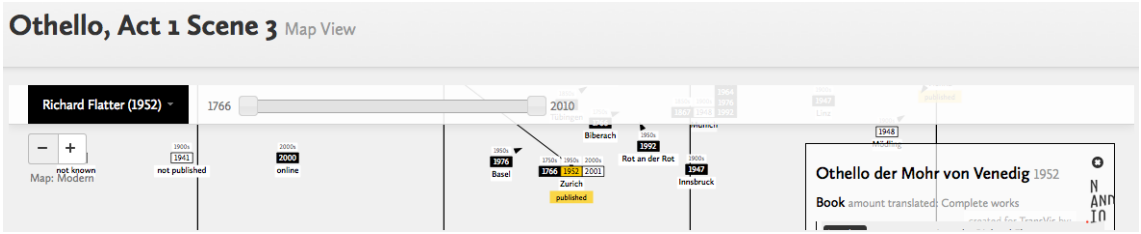


Figure 8: Othello Interactive Time-Map of version variation visualization VVV web tool [Cheesman et al., 2012]. It is an interactive map that allows the user to view the place, date and author translator of each of the German translations of the Othello play.

The tool has some interesting features and consists of four main visualizations. They are: the Othello Time-Map; the alignment maps; parallel views and Eddy and Viv view. These visualizations will be discussed in the following paragraphs.

Firstly, the Othello Time-Map, which is presented in Figure 8, is an interactive map that is used to visualize the information of all the versions of the Othello German translations. It shows the places where each of the translations was created and it displays the author's name. The user can explore all the versions on the map, click on each version and then view the date, the location and the author translator name of that translation.

Secondly, the Alignment map view, shown in Figure 9. The alignment map view enables the user to compare the original English text and the German Translated text. The system views the original English text of Act 1 scene 3 of the Othello play (on the left as shown in figure 9) and enables the user to select the targeted version of translations from a drop-down list to view it on the right as shown in figure 9. Then the visualization is created in the middle between both texts. The visualization consists of a list of bars, each represents a single segment. The height of each bar expresses the length of that segment. This shows how each author translator expands or shrinks the original text when translating. The comparison applied is segment by segment. The user can select a segment and compare the two documents based on the selected segments.

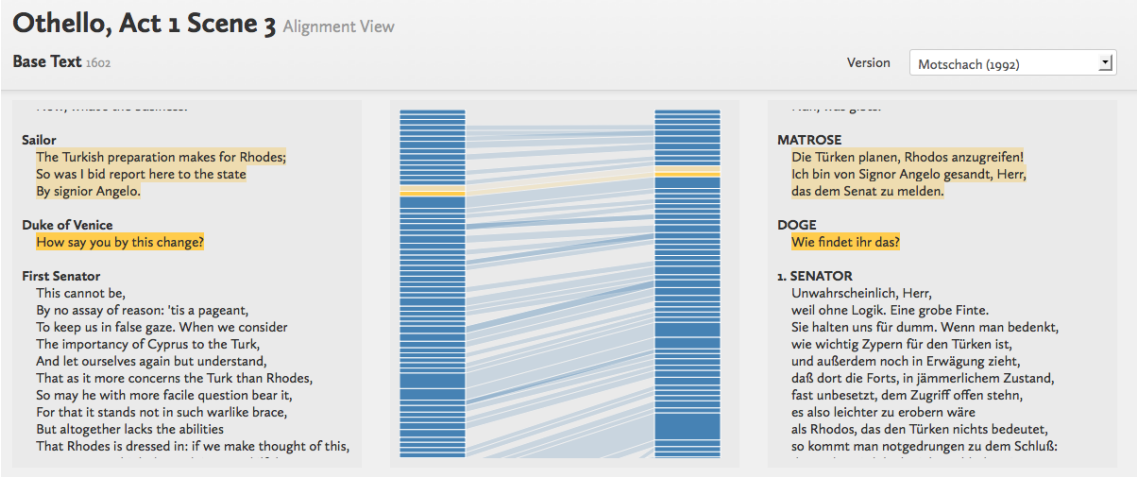


Figure 9: The Alignment map view of VVV web tool [Cheesman et al., 2012]. It enables the user to select a version of translations from the drop-down list to start the comparison with the original English text of the Othello play segment by segment.

The VVV web tool applies a visualization of a parallel view (shown in Figure 10) that assists in comparing the original English text of Othello with a single version of the German translated texts. The user can select one segment at a time for comparison. The text presented can be sorted based on the speaker, text length or text flows. The user is able to click on any segment of the original text to view the corresponding translated segment. The visualization shown in Figure 10 is a group of bars, each represents the length of each segment. However, the parallel view is a very effective visualization technique in the process of comparing and identifying variation, and it could be applied for multiple document comparisons. This visualization technique is suitable for my project and it will be implemented because of its efficiency.

Finally, the system implements the visualization of Eddy and Viv value, clarified in Figure 11. It allows the user to select the Viv type, Viv value and the sorting order and then apply them to the text.

Figure 10: The parallel view visualization of Version Variation Visualization VVV tool [Cheesman et al., 2012]. It gives the user the ability to compare the original text of Othello with a single version of the translated texts. The user can select one segment at a time for comparison.

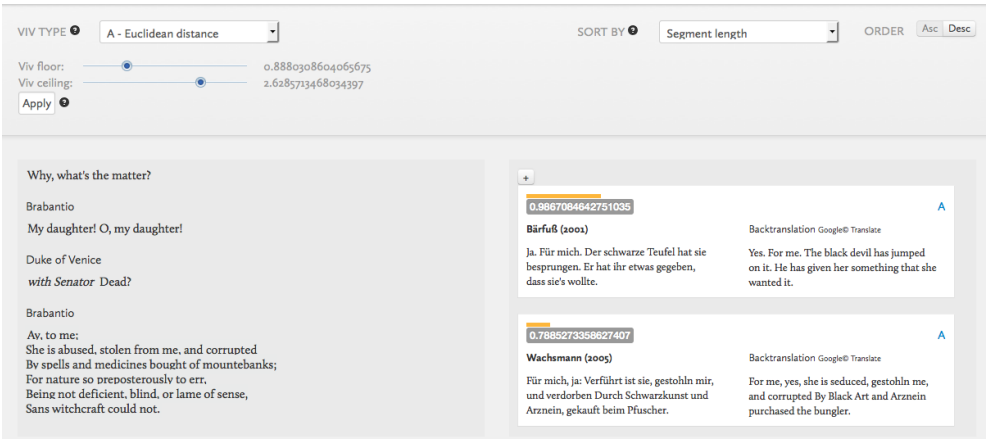


Figure 11: The Eddy and Viv view of VVV web tool [Cheesman et al., 2012]. It allows the user to select the Viv type, Viv value and the sorting order and then apply them to the text.

3 Data Characteristics

The data is the fundamental part in any visualization project. Since the aim of the visualization is to convert the data from its original form to a graphical form in order to provide a better understanding and representation of the data. Good understanding of data behaviour leads to more efficient and accurate results. In this chapter, the data relevant to

this project is discussed in detail, this includes the data source, data type and files and data description.

3.1 Data Source and type

The type of data this project visualizes is textual type data and, as declared in the introduction, it is a collection of different German translations of Shakespeare's play, Othello. The data was given to us by the Art and Humanities Department at Swansea University. The evolution of these translations has been done over a long period of time. They date from the period between 1760 and 2010. A team of researchers in the College of Art and humanities collected these versions of translations and worked on processing and converting the data into a digital text form.

However, only a sub-collection has been digitalized and preprocessed in order to be used with different software. This resulted in different file formats, so the data we have is in various types. There is a group of about thirty-four different translations of the same segment of the play in Microsoft word format, a collection of plain text files, each contains a single speech from different translations, and a group of pdf files and text files of a scene from the play.

In addition to those different files, there is an XML file that contains thirty-eight different German translations of a single scene of the Othello play. The data in this file is cleaned, preprocessed and organized in a positive way by providing all the meta-data that would be needed for further data processing.

Since our project is interested in identifying and representing the versions variations by comparing the translations using data visualization, the XML file is the most appropriate data source to use. It was decided to work with the provided XML file for many reasons. First of all, because of its well formed structure that we will discuss in the next

section. The second reason is because it includes the required information about each version and its content. Another reason is the use of segment identification in each document definition. In addition, the XML files can be read and processed efficiently by the computer.

A segment refers to a smaller portion of the document, which could be as a small as a line of text or a single word or could be a scene or paragraph of the play. Identifying each segment and defining its related attributes is very important when we are applying parallel comparisons between documents.

In the XML file there are thirty eight different translations plus the original English document. The translations are for Act 1 scene 3 of the play. Each document is composed of two main sections. The first one is the document content section, which is referred to by the tag `<docontent>`. The second is the segment definitions section, which is referred to by a `<segmentdefinitions>` tag and includes the meta-data related to each of the segments presented in the first section. In the following each of these tags content and attributes is presented.

3.2 Data Description

The XML file starts with `<eblacorpus>` start tag and ends with the matching end tag. This is the essential element that includes other tags inside it. According to T. Cheesman (personal communication, Jun 19, 2014) ebla is a "Historical name: ancient city with one of the world's first libraries", and the corpus is a collection of documents [Ward et al., 2010]. The tag has two attributes, name, which is "Othello, Act 1 Scene 3" and description. The other tags are dicussed in the following sections.

```
<?xml version="1.0" encoding="utf-8"?>
<eblacorpus name="Othello, Act 1 Scene 3" description="The Tragedy of Othello, the Moor of Venice">
    <documents>
        <document name="" authortranslator="" copyrightinfo="" description="" genre="" information="" langcode="" referencedate="">
            <doccontent>
                    <blockquote>
                        <i>
                            <q data-eblatype="startmarker" data-eblasegid="38073">[</q>
                                <q data-eblatype="startmarker" data-eblasegid="38292">[</q>
                                <q data-eblatype="endmarker" data-eblasegid="38292">]</q>
                             Text of the segment
                            <q data-eblatype="endmarker" data-eblasegid="38073">]</q>
                        </i>
                    </blockquote>
                    <blockquote> ... </blockquote>

            <segmentsefinitions>
                <segmentdefinition id="38073" startpos="0" length="14">
                    <segmentattributes>
                            <segmentattribute attribname="type" attribval="S.D." />
                    </segmentattributes>
                </segmentdefinition>

                <segmentdefinition id="38074" startpos="14" length="18">
                    <segmentattributes>
                        <segmentattribute attribname="type" attribval="S.D." />
                        <segmentattribute attribname="speaker" attribval="S.D." />
                    </segmentattributes>
                </segmentdefinition>

            </segmentdefinitions>
        </document>
        <document ....> </document>
        .
        .
  </documents>
</eblacorpus>
```

Figure 12: The Basic structure of the Othello Data XML File. This is an example of the main elements and attributes that are interested in this project.

3.2.1 All Documents Element

All the translations are enclosed within `<documents>` start and end tags, which includes a number of `<document>` tags. Each document element represents a single translation file and has a number of attributes declaring essential information about that version. These attributes are name, author translator, copyright information, genre (book, script. etc.), reference date, description, language code, “eng” for English and “deu” for German language, and version information.

3.2.2 Single Document Element

Each `<document>` element consists of two parts `<doccontent>` and `<segmentdefinitions>`, in addition to the `<alignments>` tag, which is included in all the versions of translations, but is not included in the original English text.

1. `<doccontent>` element:

    - It is composed of a number of `<blockquote>` tags.

- Each <blockquote> tag has a number of <q> tags, which refer to a “quote”. Each quote tag indicates the beginning and the end of each segment within a BlockQuote.
- Every <q> tag has two main attributes: data-eblatype, which could have one of two values, either “startmarker” or “endmarker” and “data-eblasegid” that stores the segment id. The text between the <q> start tag with the “start-marker” and the <q> start tag with “endmarker” attribute value with the same id is considered to be the text of that segment.

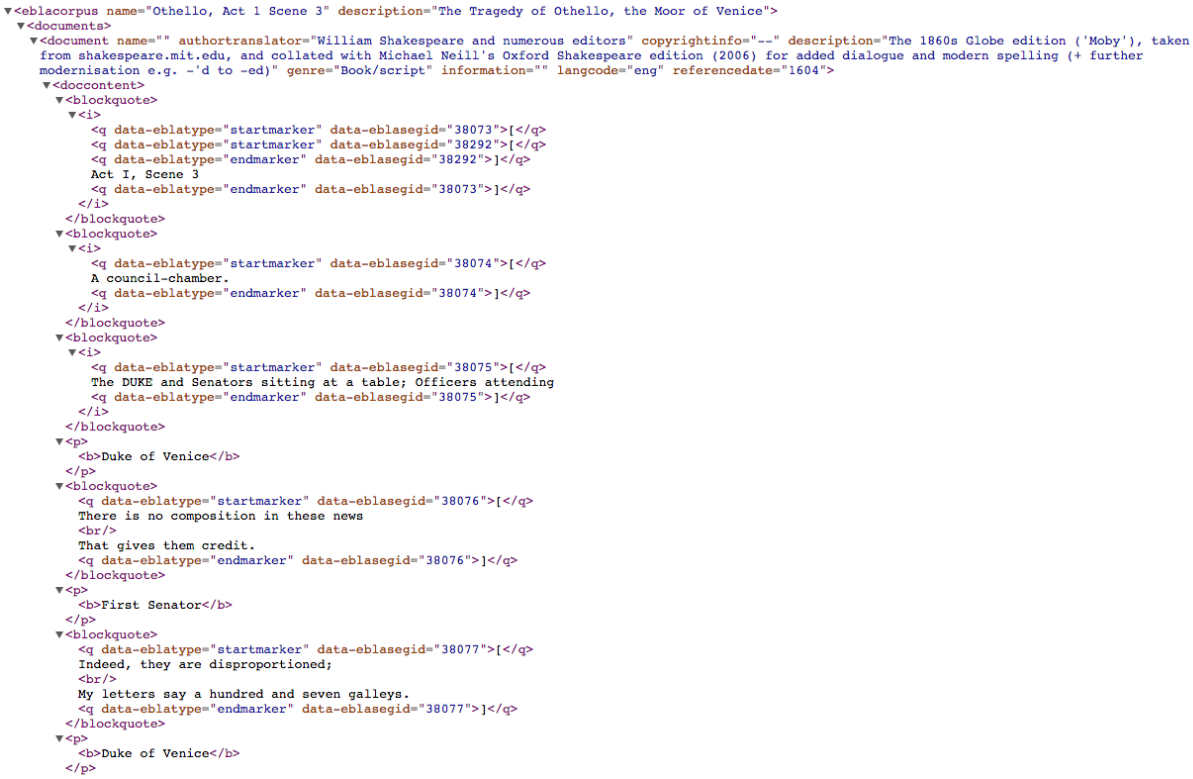


Figure 13: The structure of the <doccontent> element of the Othello Data XML File. This is a screenshot of the original XML file presenting the <doccontent> element and its nested elements and attributes.

2. <segmentdefinitions> element: this element is for the meta-data related to each of the segments in the <doccontent> section.
   - It is composed of a number of <segmentdefinition> elements. Each has three main attributes: the “id”, which is the id of the segment that the def-

inition is related to; the “length”, which is the number of characters of that segment; and finally the “startpos” which shows the start position, the number of characters before the segment, of that segment in the document.

- Each <segmentdefinition> element has a <segmentattributes> element that has one or two <segmnetattribute> elements.
- The <segmentattribute> element has two main attributes. The first is “attribname” which refers to the attribute name, and can have one of two values, type or speaker. The second attribute is “attribval” that identifies the value of the attribute name depending on its type. If “attribname” value is “speaker”, it identifies the name of the speaker of the related segment. By contrast, if “attribname” value is “type”, it identifies the type of that segment. For example, the type could be a speech or S.D. According to T. Cheesman (personal communication, Jun 19, 2014), S.D is a stage direction, which, in a play, could be ”Enter a Clown” or ”Sound of thunder”.

3. The <alignments> element: the alignment element is an element that is included only in the translated documents and not in the base text. It is for mapping each segment of the original English text with the corresponding segment in the translated text for comparison purposes.

   - The <alignments> tag consists of a number of <alignment> tags.
   - Each <alignment> has five attributes. These attributes are: id to give the alignment a unique number; notes; status; BaseTextSegIds that stores the ids of the segment in the base text and VersionSegIds that stores the ids of the same segment in that version of translation. This assists in mapping all the segments of the translated text with the corresponding segments in the original text.

Figure 14: The structure of `<segmentdefinitions>` element of the Othello Data XML File. This is a screenshot of the original XML file presenting the `<segmentdefinitions>` element and its nested elements and attributes.

There are a number of HTML elements such as `<p>`, `<br>`, `<i>` and `<b>`, which are not stored and only the text between them is.

```
▶<doccontent>...</doccontent>
▶<segmentdefinitions>...</segmentdefinitions>
▼<alignments>
   <alignment id="1923" notes="" status="confirmed" BaseTextSegIds="38143" VersionSegIds="38559,49272"/>
   <alignment id="1924" notes="" status="confirmed" BaseTextSegIds="38561" VersionSegIds="38560,49273"/>
   <alignment id="1925" notes="" status="confirmed" BaseTextSegIds="38563" VersionSegIds="38562"/>
   <alignment id="1927" notes="" status="confirmed" BaseTextSegIds="38107" VersionSegIds="37243"/>
   <alignment id="1928" notes="" status="confirmed" BaseTextSegIds="38108" VersionSegIds="37244,37247"/>
   <alignment id="1929" notes="" status="confirmed" BaseTextSegIds="38113" VersionSegIds="37245"/>
   <alignment id="1930" notes="" status="confirmed" BaseTextSegIds="38125" VersionSegIds="37246,37248,37250"/>
   <alignment id="1931" notes="" status="confirmed" BaseTextSegIds="38565" VersionSegIds="37251"/>
   <alignment id="1932" notes="" status="confirmed" BaseTextSegIds="38128" VersionSegIds="37252"/>
   <alignment id="1933" notes="" status="confirmed" BaseTextSegIds="38129" VersionSegIds="37253"/>
   <alignment id="1935" notes="" status="confirmed" BaseTextSegIds="38136" VersionSegIds="37259"/>
   <alignment id="1936" notes="" status="confirmed" BaseTextSegIds="38135" VersionSegIds="37258"/>
   <alignment id="1937" notes="" status="confirmed" BaseTextSegIds="38566" VersionSegIds="37256"/>
   <alignment id="1938" notes="" status="confirmed" BaseTextSegIds="38567" VersionSegIds="37260"/>
   <alignment id="1939" notes="" status="confirmed" BaseTextSegIds="38140" VersionSegIds="37262"/>
   <alignment id="1941" notes="" status="confirmed" BaseTextSegIds="38144" VersionSegIds="37265"/>
   <alignment id="1942" notes="" status="confirmed" BaseTextSegIds="38146" VersionSegIds="37266"/>
   <alignment id="1943" notes="" status="confirmed" BaseTextSegIds="38149,38152" VersionSegIds="37267"/>
   <alignment id="1944" notes="" status="confirmed" BaseTextSegIds="38153" VersionSegIds="37268,37270"/>
   <alignment id="1945" notes="" status="confirmed" BaseTextSegIds="38155" VersionSegIds="37272"/>
   <alignment id="1946" notes="" status="confirmed" BaseTextSegIds="38161" VersionSegIds="37274"/>
   <alignment id="1948" notes="" status="confirmed" BaseTextSegIds="38165" VersionSegIds="37276"/>
   <alignment id="1949" notes="" status="confirmed" BaseTextSegIds="38168" VersionSegIds="37277,37279,37281,37283,37291"/>
   <alignment id="1950" notes="" status="confirmed" BaseTextSegIds="38169" VersionSegIds="37284"/>
   <alignment id="1952" notes="" status="confirmed" BaseTextSegIds="38171" VersionSegIds="37286"/>
   <alignment id="1953" notes="" status="confirmed" BaseTextSegIds="38172,38178" VersionSegIds="37287"/>
   <alignment id="1954" notes="" status="confirmed" BaseTextSegIds="38175" VersionSegIds="37288"/>
   <alignment id="1955" notes="" status="confirmed" BaseTextSegIds="38176" VersionSegIds="37289"/>
   <alignment id="1957" notes="" status="confirmed" BaseTextSegIds="38167" VersionSegIds="37280"/>
   <alignment id="13324" notes="" status="confirmed" BaseTextSegIds="38105" VersionSegIds="37242"/>
   <alignment id="13325" notes="" status="confirmed" BaseTextSegIds="38762" VersionSegIds="49269"/>
   <alignment id="13326" notes="" status="confirmed" BaseTextSegIds="38764" VersionSegIds="49270"/>
   <alignment id="13327" notes="" status="confirmed" BaseTextSegIds="38775" VersionSegIds="49271"/>
   <alignment id="13329" notes="" status="confirmed" BaseTextSegIds="39619" VersionSegIds="37285"/>
   <alignment id="13548" notes="" status="confirmed" BaseTextSegIds="49442" VersionSegIds="37264"/>
 </alignments>
</document>
```

Figure 15: The structure of the `<alignments>` element of the Othello Data XML File. This is a screenshot of the original XML file presenting the `<alignments>` element and its nested elements and attributes.

4 Project Specification

In the project specification section, the features specification of our visualization system will be discussed. The feature specification will include the basic features that the system should have and the additional optional features. The user characteristics for the system will also be presented. In addition, the technology choices for developing the software will be discussed. However, the project specification, discussed in the interim document and will be modified and updated here in final dissertation.

4.1 Feature Specification

The main goal of my project is to develop an interactive visualization system for visualizing a group of different text documents. The result of this visualization should assist the user in identifying and discovering the variations between these documents easily and efficiently. The aim is to develop a visualization system for that purpose with the following

features.

4.1.1 Basic Features

1. An interactive user interface should be implemented in our visualization system.
2. The user interface should include an option that makes the user able to select the file that will be read by the software. This option should allow the user to browse and select the data source by using open file dialog.
3. Only the XML file format is acceptable to be opened.
4. The system should support reading and parsing XML file format.
5. After reading and parsing the XML file, the user should be able to select the different translations in the file. A drop-down list, with items names refers to different versions of translations and gives the user the ability to select between the translations.
6. When a version of translation is selected a visualization of that version will be displayed in the window in addition to the actual text of that version in a text browser.
7. The user should be able to view two different translations at the same time in parallel view.
8. Single document and multiple documents visualization will be provided.
9. For multiple documents visualization, the comparisons between documents will be based on blocks and segments.
10. The visualization represents the overall size of the document, based on the number of characters and the size of segments within that document. It is composed of coloured individual rectangular blocks, each of which represents a single Blockquote of a document and its height shows its size compared to the document size.

A further description about the visualizations that will be implemented and the techniques that will be used are presented in the project design section.

4.1.2 Additional Features

1. In the case of multiple documents visualization, the ability to visualize more than two documents in parallel will be studied. Two documents visualization will be implemented and considered as a basic feature of our visualization system.

2. View a brief summary of the selected version of the translation. This summary provides the user with basic information about the selected translation such as the author translator name, date, genre, description and so on. A summary about a translation is viewed when the translation is selected.

3. Provide alignment visualization, which is composed of numbers of lines that are painted between the corresponding segments of the base-text and the translation.

4. Add an option to view and hide the segment ids before each segment text in the text area.

5. Add On-Mouse-Over feature to the visualization to show the segment id and the segment text when the mouse moves over the visualization of a specific segment.

4.1.3 User Characteristics

The idea and the requirement of this project, discussed in the introduction, indicate who will use and evaluate the developed visualization tool. The researchers in the College of Art and Humanities at Swansea University are the users of version variation translation visualization system.

4.2 Technology Choices

According to the project specification and the required features presented previously, the selection of the technology needed to develop our software was made. The programming language and the additional tools selected are best suited to the development framework. The tools needed, beside the programming language are: the GUI library to support user interaction with the program; graphics library for creating visualization; backing up tool and code commenting. In the following, our choices of technology are going to be discussed.

4.2.1 The programming Language

C++ programming language is the language chosen to be used to build our software. C++ was basically developed from C. It is a general-purpose and powerful programming language that was developed to be used in a conventional compilation and runtime environment, which is the C programming environment. C++ has the advantage of the efficiency of low-level programming; it has features of manipulating hardware facilities directly and efficiently [Stroustrup, 1997]. Moreover, it was decided to use C++ programming because of its support to a variety of libraries, especially graphics and the GUI programming libraries.

4.2.2 GUI programming

In building our system C++ programming language with the Qt library for GUI programming support will be used. Qt is a development framework, which is considered to be comprehensive. However, there are a variety of quite good features in Qt making it a more preferable framework to be used in developing our system than other frameworks. Its good features are:

- It is well documented. A helpful and inclusive documentation is available on the Qt project/ documentation website [Qt, nd].

- It is a free and open source platform.
- A lot of advanced features are provided by Qt, and all the features we need for building our project are available in Qt. For example for reading and manipulating the XML file format, there are built-in functions in Qt for reading and writing XML files.
- It has the advantage of platform independency, the graphical applications generated by Qt are using the "write once, compile anywhere" approach [Blanchette and Summerfield, 2008].
- Moreover, Qt libraries and tools are not restricted, they can be used and run on various environments and on various operating systems [Blanchette and Summerfield, 2008].

4.2.3 Additional Tools

In addition to the basic programming tools that are necessary for building the basic specification of our project, the following supplementary tools were chosen.

4.2.3.1 Doxygen: While designing and developing a software project, it is very important to take into account the source code documentation. The source code should be well commented and explained to facilitate the modification and maintenance processes.In our project, Doxygen is used for source code documentation. Doxygen is a widely used tool for generating documentation for C++ source code. It also supports many other programming languages.

Doxygen generates documentation in different formats including pdf and HTML formats [Doxygen, nd]. The Doxygen tool is provided by Qt creator. Doxygen is used to create source code documentation in HTML format and to create the collaboration and hierarchy diagram.

4.2.3.2 Git Repository: For source code backup requirement, a remote repository that is hosted on BitBucket website will be used. This helps in organizing the development process of our software and in keeping an updated version of the software available.

QT creator supports the use of the “Distributed Version Control System” tool Git that enables the user to push the project code into the remote repository, clone it, modify it, and commit some changes to the repository.

5 Visualization Using Existing Visualization Tools

Visualizing textual data is in demand in many different fields due to the dramatic increase in the size of data. Visualizing texts helps in exploring the content, analyzing the text and understanding the data generally. However, before starting to design our visualization tool. It is useful to explore and use the free available text visualization tool with the Othello data. Using these tools with our dataset and looking at the applied visualization techniques will assist us in the process of selecting and implementing the appropriate visualization techniques for our tools. In the following, the advantages and the limitations of the tested tools will be discussed.

5.1 Many Eyes Visualization Tool

ManyEyes is a web visualization tool developed by IBM. This tool provides variant types of visualization that could work with different types of data sets. The web tool allows the user either to choose from the existing data sets or upload his/her own. Using ManyEyes requires user registration for a free membership in order to benefit from its services. ManyEyes offers a range of services for users including choosing a data set, selecting a visualization method, creating the visualization and saving it. It allows the user to provide his feedback about the tools. However, this is a public tool that is available online at [IBM, nd].

5.1.1 Tag Cloud

Tag Cloud is a common type of text visualization that visualizes the words in the document according to their frequency in that document. The visualization technique depends on changing the size and the colour of the words in the document depending on their frequency. This tool was used with one German translation document of Othello. The result of the visualization is shown in figure 16 .

As presented in figure 16, the system lists the words in the document in parallel rows based on their appearance in the document. The colour and size of each word represent the word frequency in the text. The user is able to search for a word of interest within the visualized text. The system offers a feature for displaying the original context of the word and the number of its occurrences when the user moves the mouse over the word as presented in figure 17. There are no additional features provided such as elimination of unwanted words or a group of words selected.

The benefits of using this tool with Tag Cloud visualization is the fact that it provides fast exploring of the text content. The user can identify the most and least common words in the document, the number of occurrences of each word, viewing its context within the text and search for a specific word.

There are some limitations of this tool. It visualizes all the words in the document without any exclusion. The user will not be able to eliminate the words which are not of interest or the very common words. There is no feature enabling the user to select a group of words. The user can only look up one word at a time. Since the Tag Cloud visualization is a single document visualization technique, this will not be a very effective technique if the objective is to compare the different translations.

5.1.2 Customizing Word Cloud Generator

This is similar to the previous Tag Cloud visualization, but with more customization and interaction options. The layout of the visualization can be changed. The appearance and the content of the layout can be adjusted according to the user settings. For appearance changes, the user can choose: the colour theme and variance; the direction of the words in the cloud such as horizontally, vertically etc.; the font and change the words' cases.

For the content, the system detects the language of the data set entered by the user.

Figure 16: Many Eyes: Tag Cloud visualization. The visualization of Tag Cloud on Many Eyes web visualization tool [IBM, nd] using the data set (Act 1 scene 3 of Othello) of one version of the German translations. All the words in the text are listed and visualized by diiferent colours and sizes according to their frequency in the document.

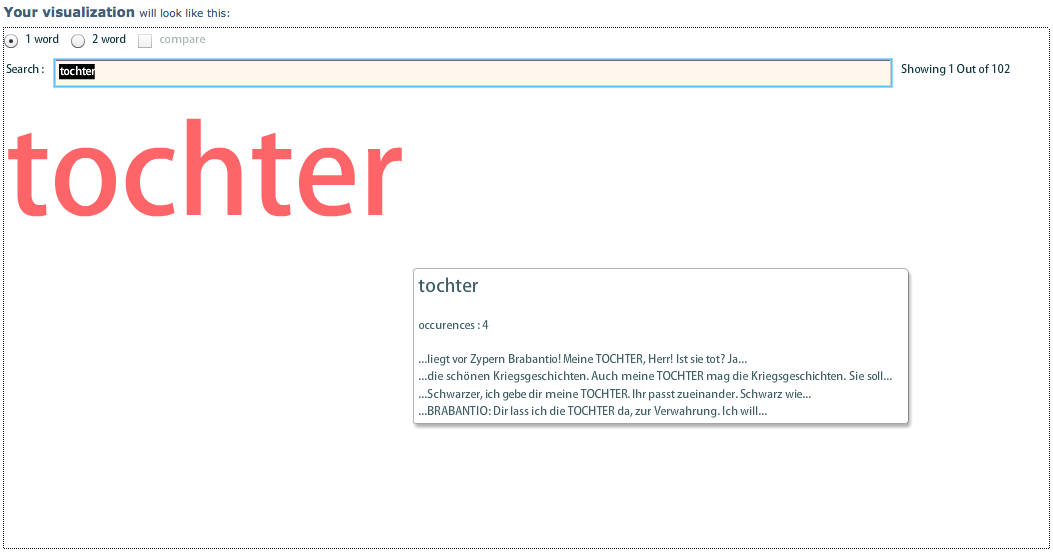


Figure 17: Many Eyes: Tag Cloud visualization. The visualization of Tag Cloud on Many Eyes web visualization tool [IBM, nd] using the data set (Act 1 scene 3 of Othello) of one version of the German translations. This shows the result of the user search for a specific word. The number of occurrences and the word context in the original text is shown.

It overcomes the main the limitation of tag cloud by providing a feature for excluding the most common words for each language. Furthermore, it allows the user to ban some words of his/her selection. Figure 18 examples of these options.

(a) The visualization layout with the default settings

(b) The visualization layout for the same data, but with changing the number of words, the words direction, the colour theme and the word cases.

Figure 18: Many Eyes: Word Cloud visualization [IBM, nd]. Both of the visualizations use the same data set, which is one version of the German translation of Othello.

5.1.3 Word Tree Visualization

Word tree visualization is a visualization technique that represents the frequency of a term of interest and its context within a single document. The term is represented as the root of the tree and the branches of the tree are the different contexts of this term in the document [Ward et al., 2010]. In the word tree visualization provided by Many Eyes [IBM, nd] shown in figure 19, after selecting the data set, the user has to select a specific word "a term of interest" to start the visualization process. This term becomes the root of the tree, and the branches of the tree contain the context in which that term is presented in the document. The number of occurrences of the term is displayed on the top. However, the the branches show the different contexts, while the size represents the frequency of the root term within the data set.

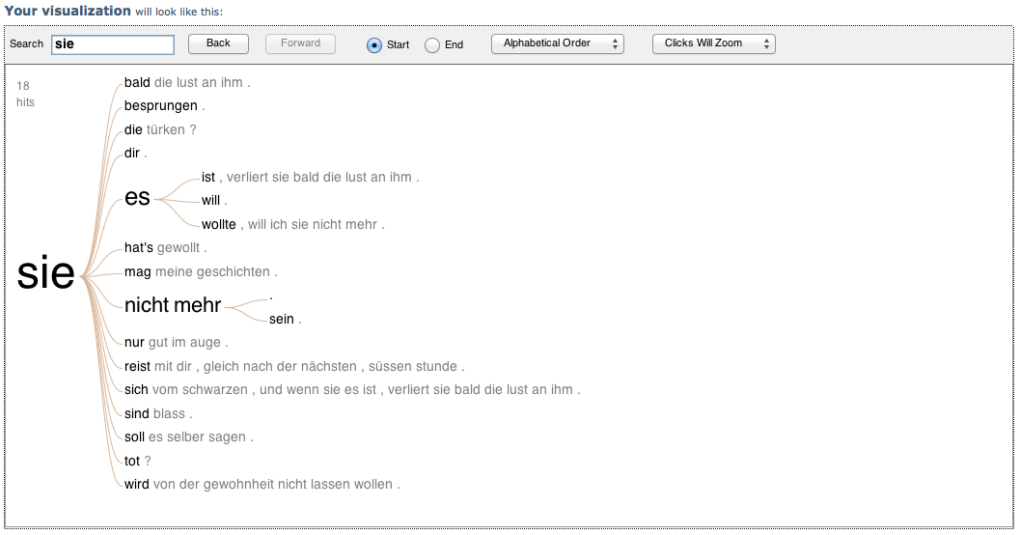
The user interaction options applied are:

- Term of interest selection i.e. selecting the root of the tree.
- Moving backward and forward through the root and its branches.
- The user is able to select the tree direction, start or end. Start means that the branches show the text, context, after the selected word. And end means that the branches show the text, context, before the selected root. The difference is clarified in figures 19a and 19b.
- Sort the branches according to their occurrences, frequency or their alphabetical order as shown in figure 20a.
- Select the view modes, the highlighter and clicks will zoom, as shown in figure 20b.

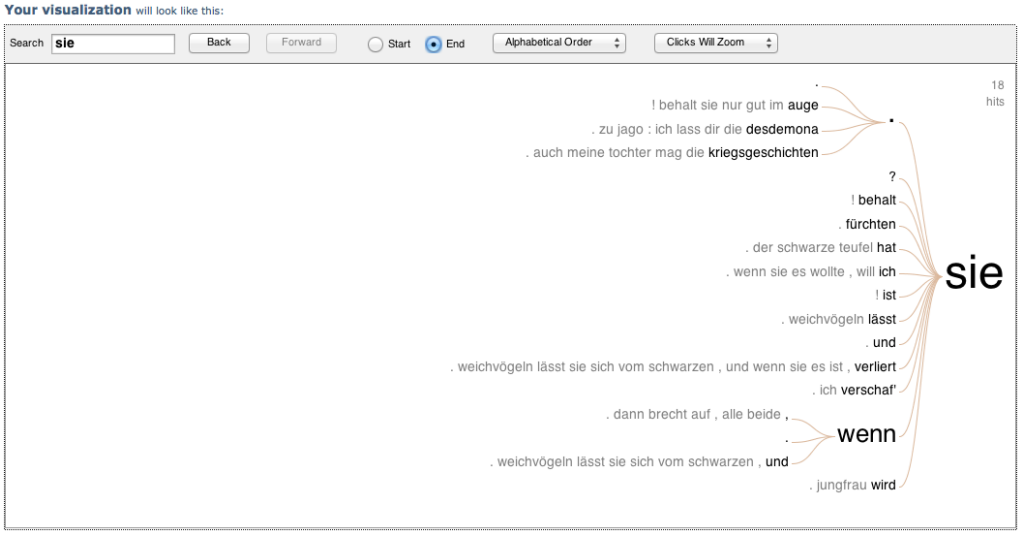
The word tree visualization in the Many Eyes web tool has many benefits. This includes: the useful user interaction options presented above; the direct visualization feature; exploring sentences that contain the word's of interest and identifying the words'

frequencies, which is a general advantage of the word tree technique.

On the other side, there are some limitations: it only shows a small part of the context of the root; the system gives no recommendation about what terms the user can start with; it does not give the most common words in the document. In general, the word tree is for single document visualization and is effective for exploring texts and identifying the frequencies of terms.



(a) Start: The root term is at the start in the context



(b) End: The root term is at the end in the context

Figure 19: Many Eyes: Word Tree visualization. The visualization of Word Tree on the Many Eyes web visualization tool [IBM, nd] using the data set (Act 1 scene 3 of Othello) of one version of the German translations. The root of the tree is selected by the user, the branches show the contexts.

(a) Branches Sort Options (b) View Modes Options

Figure 20: Many Eyes: Word Tree visualization [IBM, nd]. This figure shows the user interactive options.

6 Project design

According to the type of the data that needs to be visualized, data visualization has three main fields .These three fields are: volume and flow visualization, considered as scientific visualization, and information visualization. In scientific visualization the data is spatial data where the data in information visualization is abstract and does not exist in a spatial domain. In our project, the data that we work on is abstract data. There are two main issues which make the field of information visualization more complex and challenging. The first is the complexity of the data which means the data may come in varied forms and structures. The second is the scalability which is related to dealing with a big amount of information.

Developing a process for a computer-based visualization system according to, [Ware, 2004], goes through four basic stages: The four basic stages according to [Ware, 2004] are:

1. Data collecting and storing.
2. Data processing and transformation.
3. Produce graphics and display the visualization output.
4. Human perception and cognitive processing, observation.

These stages are joined together in a series of feedback loops, as illustrated in figure 21.

The visualization pipeline clarifies the visualization process, see figure 22, and clearly defines the stages of designing and developing the visualization system. It provides a set of organized stages for generating a visual representation of data. According to [Ward et al., 2010], the visualization pipeline is described as "*the process of starting with data and generating an image, a visualization, or a model via the computer*", this process ends

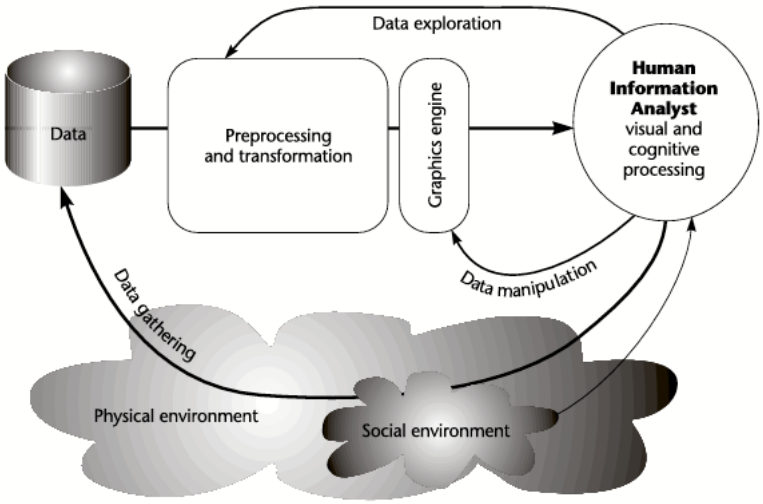


Figure 21: Visualization Process Diagram [Ware, 2004]: This schematic diagram shows the basic stages of the visualization process [Ware, 2004].

with the user interacting with the data. The stages of the visualization pipeline are illustrated in figure 22.

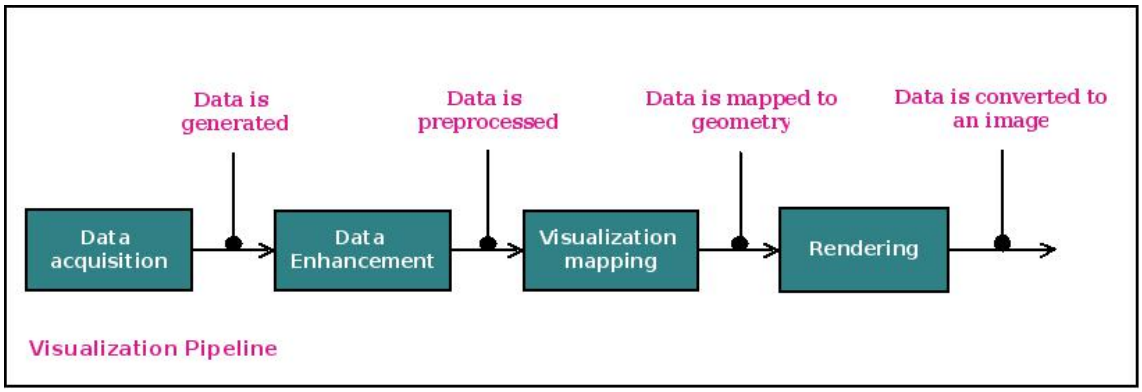


Figure 22: The Visualization Pipeline. Image Credit: based on the Visualization Pipeline in [Ward et al., 2010].

In designing and developing our visualization software two main stages are required. The first stage is reading the different translations from the data source XML File, which is described in detail in the data characteristics section, and the second is the visualization generation stage. These two stages will be discussed in the following subsection. In addition, class and collaboration diagrams of the software are included.

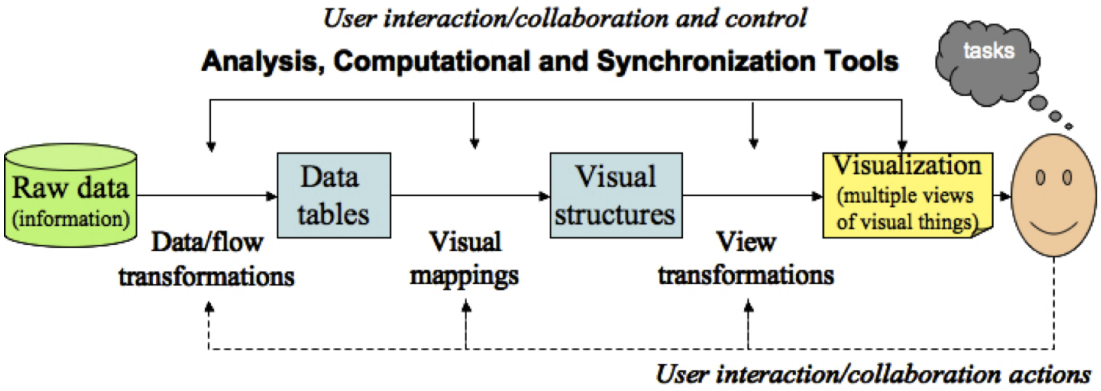


Figure 23: An example of the visualization pipeline. There are different visualization pipeline, but they are all composed of a number of steps to transform data into graphical representation and display it on the screen. Image credit: [Ward et al., 2010].

6.1 Reading and Storing the Data

As discussed in the data characteristics section, the data source is an XML file that includes thirty eight German translations of (act 1, scene 3) of Othello together with the original English text. An appropriate design using the concept of object-oriented programming has been developed to read, properly and efficiently, the required data and meta-data from the data source. This object-oriented structure facilitates maintaining, enhancing and modifying the system during the development process. In the following, the structure of the main classes that are used to read and store the data will be presented.

- **DocumentReader Class:**

  The DocumentReader class is the base class that is designed mainly for reading the translations and parsing the XML file. Document class is part of the DocumentReader class. During the process of parsing the XML file, a new Document class object is created for reading each translation's content and metadata. The structure of the DocumentReader class is presented in figure 24.

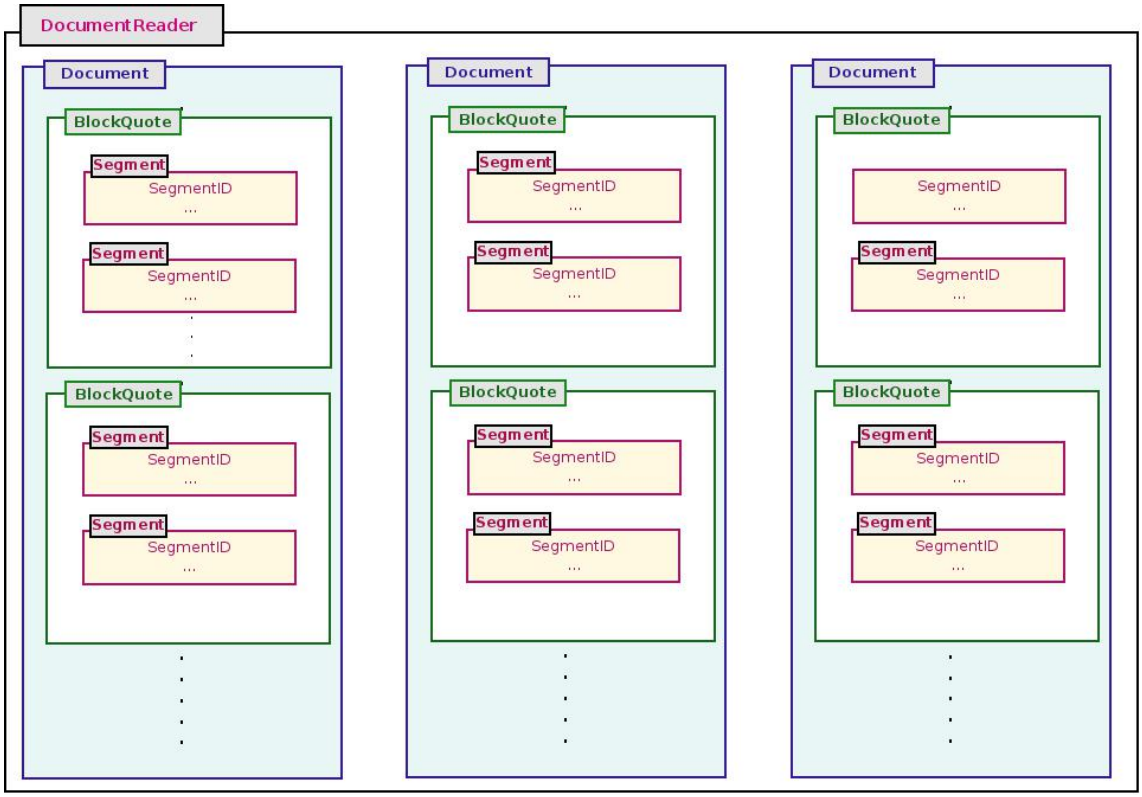


Figure 24: DocumentReader Class Structure : DocumentReader class parses the XML file and creates a Document class object for each translation. Each Document class object is composed of a number of BlockQuote class objects. Each BlockQuote object is composed of a number of Segment class objects.

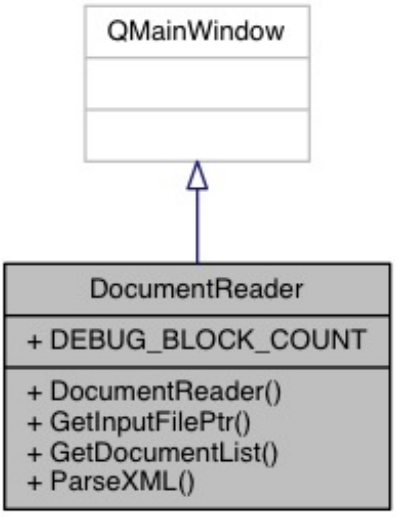


Figure 25: DocumentReader Class Inheritance Graph.

- **Document Class:**

  The Document class reads the basic information about each document such as: the

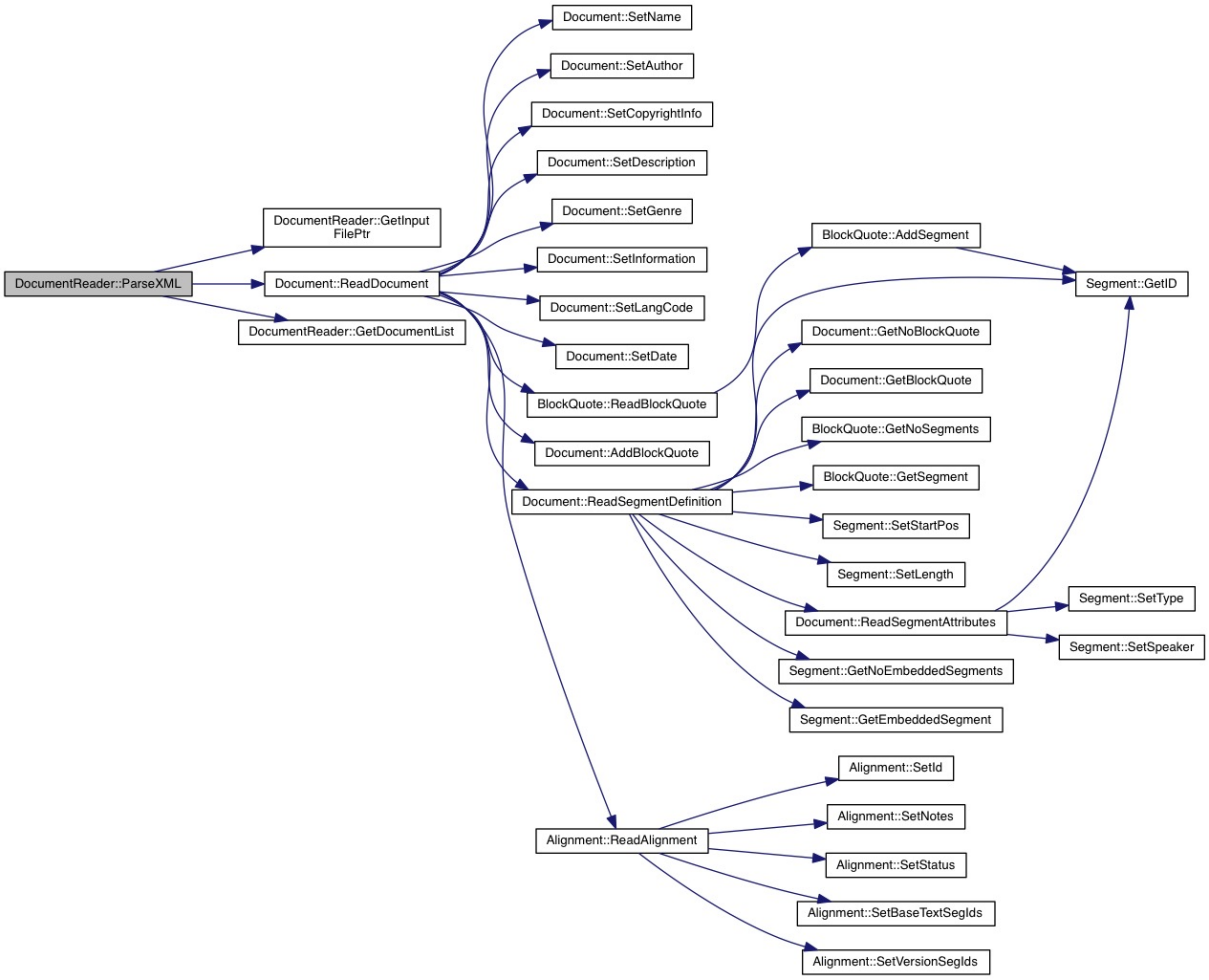


Figure 26: DocumentReader Class : The collaboration graph for ParseXML method of DocumenReader class, which is the base method for analysing the XML file content as generated by Doxygen.

document name; author translator name; copyright information; genre; reference date; language code and description. In addition, it reads the content of the document. Each document in the XML file has three main sections: the first is the document content; the second is the segment definitions and the third is the alignments section, except for the original English document, it does not contain an Alignments section. Since the document content section is composed of a group of Blockquotes, the Document class creates a new BlockQuote class object to read every Blockquote. The segment definition section is read by the Segment class, because it is additional information about the segments. For the alignments section, a new Alignment class object is created. The Document class structure is illustrated

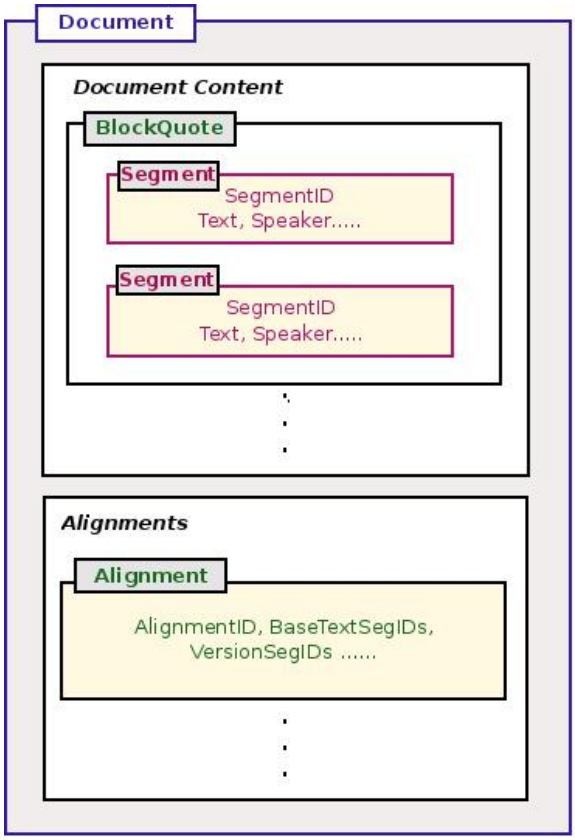


Figure 27: Document Class Structure : Document content in a group of BlockQuote objects and alignments is a group of Alignment Objects.

in Figure 27.

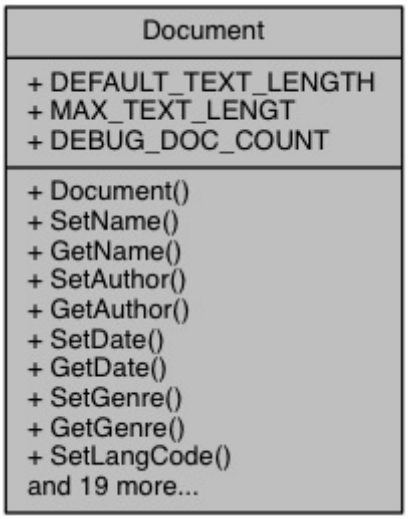


Figure 28: Document Class Diagram as generated by Doxygen.

- **BlockQuote Class:**

  The BlockQuote class is designed to read the quotes in each document and is part of the Document class. Each quote is composed of a number of segments. The

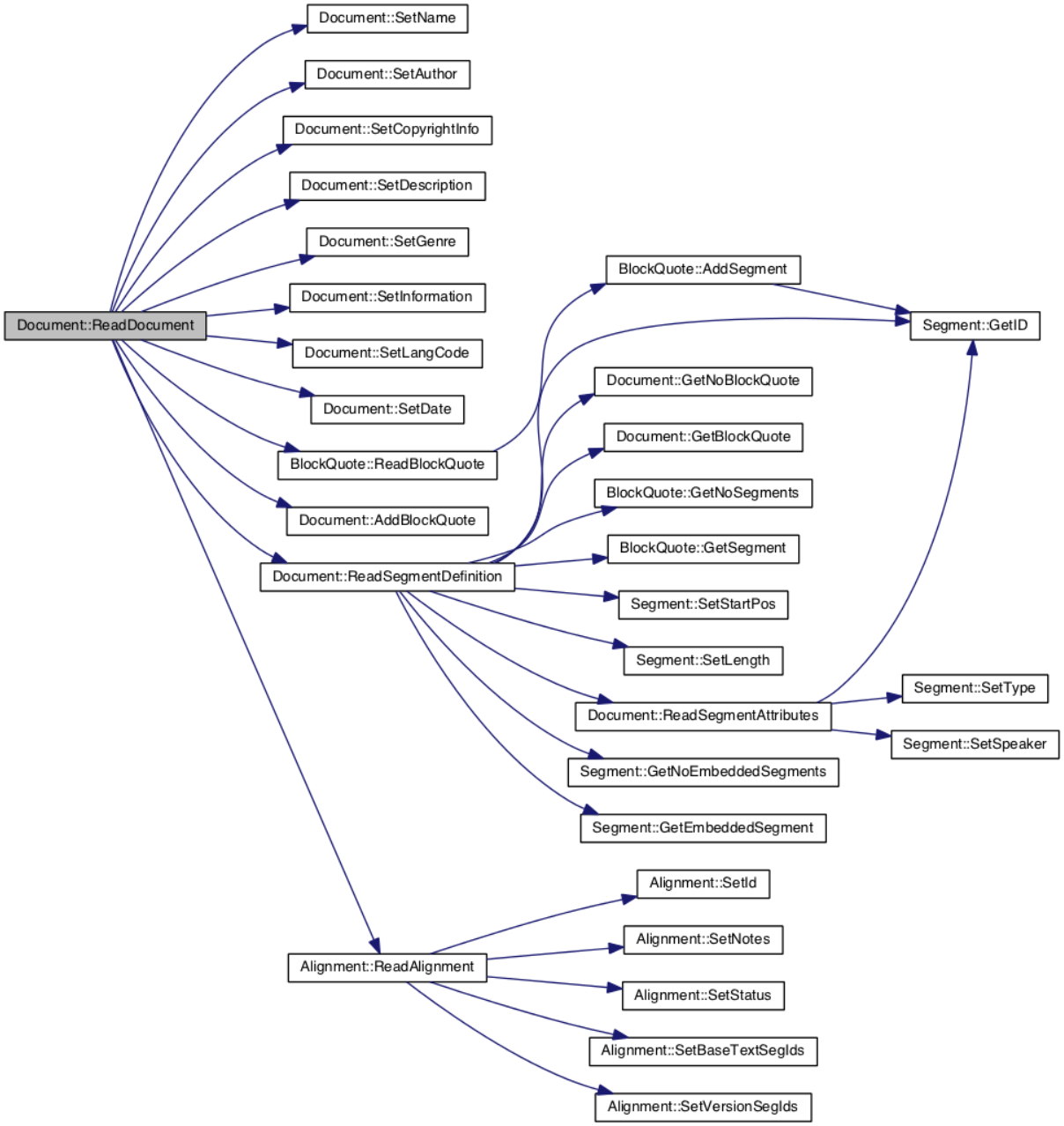


Figure 29: Document Class : The collaboration graph for ReadDocument method, which is the base method for reading each document in the XML file as generated by Doxygen.

BlockQuote class creates a Segment class object for each segment. Each segment inside the Blockquote has a text and unique Id. The segment Id is used for many purposes, including getting the additional segment information from the segment definitions section of the document and getting the segment alignment information.

- **Segment Class:**

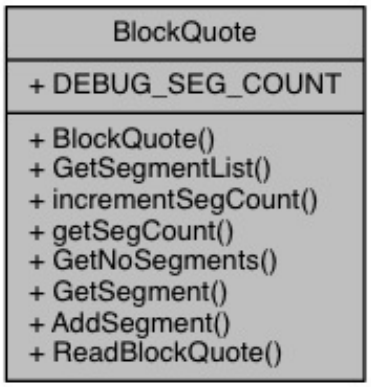


Figure 30: BlockQuote Class Diagram as generated by Doxygen.

The Segment class is part of the BlockQuote class and basically it is used to store all the information related to one segment such as segment Id, text or content, speaker, length, start position and end position. Furthermore, it stores the embedded segments information in case the segment contains embedded segments.

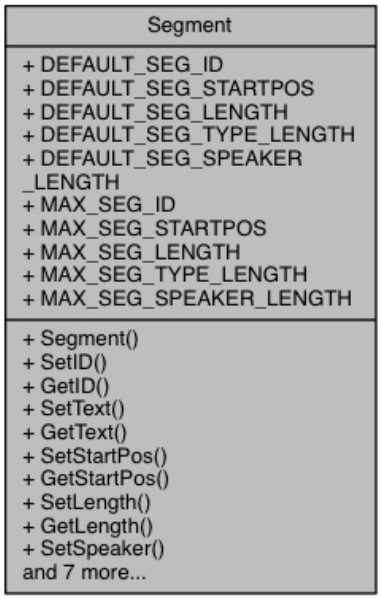


Figure 31: Segment Class Diagram as generated by Doxygen.

- **Alignment Class:**

  The Alignment class is part of Document class, since the alignments come in a separate part or element in each document. The Alignment class stores the alignment information of the document, where each “Alignments” section is composed

of a list of Alignments. Each Alignment has an Id to define the Alignment, version segment ids and the corresponding base text segment ids that are related to that particular segment. This information assists in connecting each segment of a version of translations with the corresponding segment or segments in the base English text.

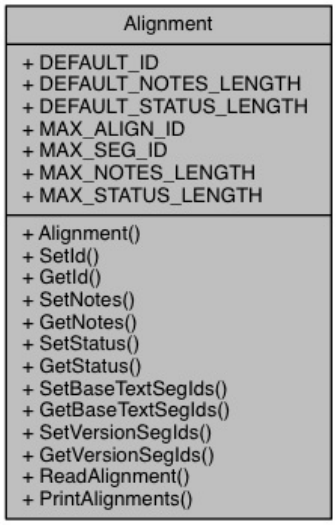


Figure 32: Alignment Class Diagram as generated by Doxygen.

6.2 Visualization Generation

The Visualization Generation Stage comes after reading and storing the data. The classes being designed for rendering the visualization are inherited from the QWidget class, which is the Qt base class for all the Qt GUI objects. Inheriting the visualization classes from the QWidget class allows them to paint themselves on the screen, which is the purpose of the visualization class, and it makes them receive the system events such as the mouse events, important to provide user interaction. The visualization widget is made to be a part of the main interface, and it will be generated and rendered after reading the data from the selected file.

However, as presented in the Project Specification Section, a parallel visualization of the translations is needed in order to show the variation between them. Our visualization method is based on the BlockQuotes and Segments of each translation. A detailed description about the visualization classes is presented in the following sections:

- **BlocksVisualization Class:** The BlocksVisualization class inherits the QWidget class of the Qt Library. It provides the parallel text visualization, which is the basic visualization in our project. The theoretical concept of this visualization is to calculate the length of each Blockquote, including all the segments contained within it, in the document i.e. the number of characters in the Blockquote and then paint them as rectangular blocks, where the height of each block is calculated according its length. Furthermore, it renders the alignments between the corresponding segments of the selected version of the translation and the base-text based on the alignment information given in the document. The class hierarchy of the BlocksVisualization class is shown in figure 34.

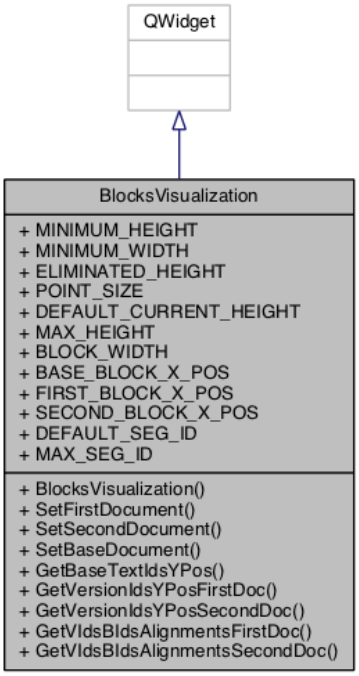


Figure 33: BlocksVisualization class inheritance diagram as generated by Doxygen.

- **CustomizedToolTip Class:** The CustomizedToolTip class inherits the QWidget class. It is designed to provide the user with the required information about the visualization when he/she moves the mouse over specific areas of the visualization widget. The class hierarchy of CustomizedToolTip class is shown in figure 34.

- **ColorsCollection Class:** This is a simple class that stores colour values to be used later in rendering the visualization. The purpose of this class is to avoid the redundancy of colour definitions and to facilitate accessing the required colour by its name rather than its actual value.

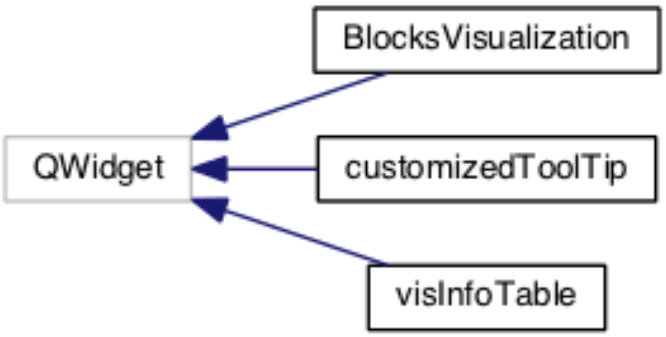


Figure 34: A class hierarchy diagram of the BlocksVisualization and the Custom Tooltip classes.

Further description about these classes is discussed in the project implementation section.

6.3 GUI

For the Graphical User Interface, there is one main class, which is the MainWindow GUI class.

- **MainWindow Class:** is a class for creating the main window in our software and it contains the essential graphical user interface. The GUI provided by the MainWindow class allows the user to select the data source to start the data analysis process and visualization rendering. This class connects the user interface with the other classes in the software such as DocumentReader class, which is responsible for parsing the XML file.

7 Project Plan and Timetable

The project plan section reviews the timeline of the project and discusses the methods used to evaluate the progression throughout the development process. It involves: the software process model; the coding conventions; project timetable and finally risk assessment. The project plan and timetable section was discussed previously in the interim document.

7.1 Software Process Model

The software process model is the development strategy that guides the software engineers during the project development. This development strategy is described as a problem-solving loop that starts with a defined problem, develops a solution and terminates with a complete system. However, choosing the proper software process model depends on: the nature of the project; the kind of application needed to be developed and the method needed to be applied [Pressman, 2001]. The process model is composed of a set of phases, each of which has to be accomplished before moving to the next phase. The software process model is an iterative process that allows each phase to be modified and edited many times until the project is completed.

There are many different software process models, each of which has different characteristics. The focus will be on the chosen model. However, in order to decide which software model is the most suitable for developing this project, the pros and cons of different types of software process models had to be taken into consideration. This resulted in using the Spiral model.

The Spiral model, which is presented in figure 35, was mainly developed to combine the iterative feature of the Prototyping model with the systematic, controlled features of the Waterfall model. It was decided to use the Spiral model because it assists in providing a rapid development process of incremental versions of the developed software, which

can be modified during the software process life cycle.

During the early stages of the software process, the incremental version could be just a prototype of the software. A complete version of this prototype is increasingly produced during the later stages of the process [Pressman, 2001].

A Spiral model consists of many tasks. The number of these tasks ranges between three to six task areas. Figure 35 shows a typical Spiral model consisting of six task areas. These tasks are: customer communication; planning; risk analysis; engineering; construction and release and customer evaluation [Pressman, 2001].

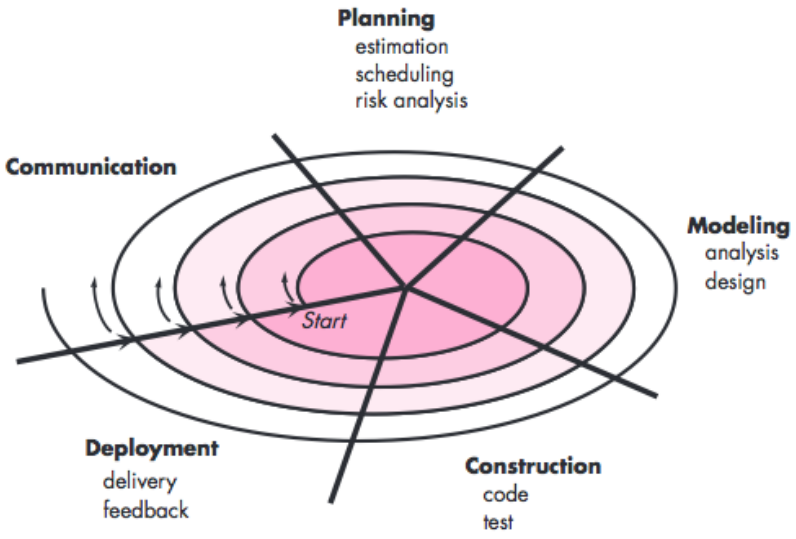


Figure 35: A typical Spiral model. Image credit: [Pressman, 2010].

7.2 Coding Conventions

It is significantly important to have a well designed and structured code. Following an appropriate coding convention helps to keep the software source code organized, and makes it much easier to manage, modify, and update the code. In coding the software system, "Bob's Concise Coding Conventions (C^3)" [Laramee, 2010a] is followed. Bob's Concise Coding Conventions is a set of ten concise rules that are simple and easy to

follow. These rules are applied during the software design and implementation processes to keep the source code in an organized format. Following these rules assists in producing a successful software project and helps in writing a professional code that is easy to read and modify.

7.3 Timetable

In planning the project, the time required for each task to be completed is specified and it is a very important aspct of the project. Bob's minutes of meeting protocol [Laramee, 2010b], which is very useful to manage, organize and give the meetings a structure, is followed. All the project's tasks are broken down into smaller tasks and assigned a deadline for them to be accomplished before the next meeting. Table 3 shows the timetable that reviews the progression of the project and the deadlines. This will help in evaluating the progression we make throughout the project's development.

7.4 Risk Assessment

Risk management is a significant field that is a requirement in different domains such as: the national security; chemical industry; nuclear power reactors; financial investments and many others. In 1989, the risk management field of software project development was acknowledged as an independent field.

The role of risk management of a software project is to continuously: indicate; analyse; trace and control the risks related to the software development process. Risk management provides a significant contribution in the success of the software development process. In contrast, perceiving risks in a non systematic manner leads to reducing the effectiveness and the sufficiency of the software project [Sarigiannidis and Chatzoglou, 2011].

Table 3: Project Timetable

	Description	***Date***
1	Initial document deadline	12 March 2014
2	Update the literature review to follow a specific structure [Laramee, 2011]	4 April 2014
3	Start writing the final report + update the literature review	11 April 2014
4	Read the translations of txt file format	17 April 2014
5	Project plan and timetable	2 May 2014
6	Final report deadline	6 May 2014
7	Study the XML file structure + using the existing visualization tools	4 June 2014
8	The project object-oriented design	16 June 2014
9	Read a single document from the XML file	23 June 2014
10	Extract specific information about the document	9 July 2014
11	Read segments and blockquotes of the document	23 July 2014
12	Read segments definitions and embedded segments	30 July 2014
13	Read all the translations and their meta-data	6 August 2014
14	Display the documents content on the GUI + add user options	13 August 2014
15	Parallel text visualization	27 August 2014
16	Read the translations alignments	19 September 2014
17	Draw the alignments between the corresponding segments	30 September 2014
18	Add some interaction features +writing the dissertation	6 October 2014
19	Add tables for testing the visualization +writing the dissertation	14 October 2014
20	Add customized tooltip to show information about the visualized text	20 October 2014
21	Proofread the dissertation	23 October 2014
22	Dissertation document deadline	31 October 2014

However, discovering and specifying the risks in the early stages of the software process is the key for an efficient risk management. Since the Spiral model for the software development process is used, it necessitates a direct consideration of potential technical risks at all stages of the software development. Implementing the Spiral model accurately should decrease potential technical risks before they become more complex [Pressman, 2001].

8 Project Implementation

In the project implementation section, the project specification section and discuss the project application of the project requirements is extended. The requirements changed during the project development process. The implementation of basic and additional features are presented in table 4. In the following, the details of the implementation of the project features are illustrateed.

Table 4: Project Implementation Table

Implementation Progression	
Basic Features	Level
1) XML File Reader	Done
2) Parallel Text Visualization	Done
3) Alignments Visualization	Done
4) Actual Text Visualization	Done
5) Single and multiple documents visualization.	Done
Additional Features	Level
1) User interaction options:	
a) Show/hide segment ids before segment text.	Done
b) Show segment Id and text on-mouse-moves over.	Done
c) Document summary on-mouse-over	Done
2) Visualize more than two documents in parallel.	Done
3) Visualization Information Tables.	Incomplete

8.1 Basic Features

This section includes a detailed description of the implementation of the must-have features of the software. It involves the implementation of the XML file reader, the visualization rendering and the implementation of the user interaction options. In addition, screen captures of the software interface are added to provide further illustration.

8.1.1 The XML File Reader

The greatest challenge in this project is reading and parsing the XML file properly. As the structure of the XML file discussed in the data characteristics section shows, there are many elements and nested elements and a lot of attributes that need to be stored correctly,

in order to move to the next step, which is the visualization rendering step. Good object-oriented design of the software definitely helps in completing this task correctly and leads to the creation of a successful software project. The proposed design for the file reader makes the process of reading and refining the data and the meta-data easier to manage and modify.

Qt, which is the framework being used with C++ programming language to build our software, includes and supports different libraries for reading data from different files format. QFile and QXmlStreamReader are the libraries used in our project. QFile is a class that supports many methods such as: opening, closing, reading and writing the different files. QXmlStreamReader class also has many methods that assist in reading the elements of the XML file. It reads the XML file as a stream of tokens, and facilitates the process of identifying the token name, attributes and type.

The main class that is responsible for opening and reading the data source file is the DocumentReader class. This class analyses the XML file and creates a Document class object for every document element. Each document represents a version of the translations and contains document content and meta-data. The object-oriented design of the file reader i.e. the DocumentReader class is illustrated in the project design section.

A GUI has been designed to allow the user to load the file and interact with the data after the file content has been processed by selecting the document to display. More details about the output of reading the data will be provided in the actual text visualization section and in the user interaction options section.

8.1.2 The Parallel Text Visualization

The main visualization in our project is the Parallel Text Visualization, which is designed to help in comparing the base text with the other translations and in comparing each

translation with the others. The concept of this visualization is to draw the document, base-text or translation, and its BlockQuotes based on their length, which is the number of characters. Three documents are visualized in parallel. The base-text document is drawn in the middle and the other two documents,selected by the user, are drawn on each side of it. Each document is drawn as a list of blocks, each visualized block represents a single BlockQuote of the document. The height of each block in the visualization is calculated from the number of characters of the related BlockQuote in the document, and expresses the percentage of that block relative to the whole document. The parallel text visualization is created by the BlocksVisualization class. The visualization generation process goes through the following steps:

1. Specify the height and the width of the widget in order to scale the visualization to be painted within the widget area.

2. Specify the document drawing area. This is to specify the maximum and minimum possible height for a document according to the height of the widget.

3. Calculate the block length. Since the block is a BlockQuote composed of a number of segments, then the block length is calculated by the equation:

$$\boxed{blockLength = \sum_{i=0}^{n} segment(i).getLength()} \quad (1)$$

   Where n = the number of segments in the BlockQuote and getLength() method return the number of characters in the specified segment.

   This process is repeated for all Blockquotes in the document. The result of each "blockLength" is then added to a list called blockLengths that stores the lengths of all the blocks.

4. Calculate the document total length from the sum of all blocks lengths by the equation:

$$\boxed{documentLength = \sum_{i=0}^{n} blockLengths(i)} \quad (2)$$

Where n = the number of BlockQuotes in the document, blockLengths(i).size().

5. Draw the blocks of the documents. This method is firstly invoked to draw the base text document blocks and then to draw the two other documents based on the user selection.

6. The visualization is changed according to the user selection of the documents.

Figure 37a and figure 37b show the result of the Parallel Text Visualization for different translations.

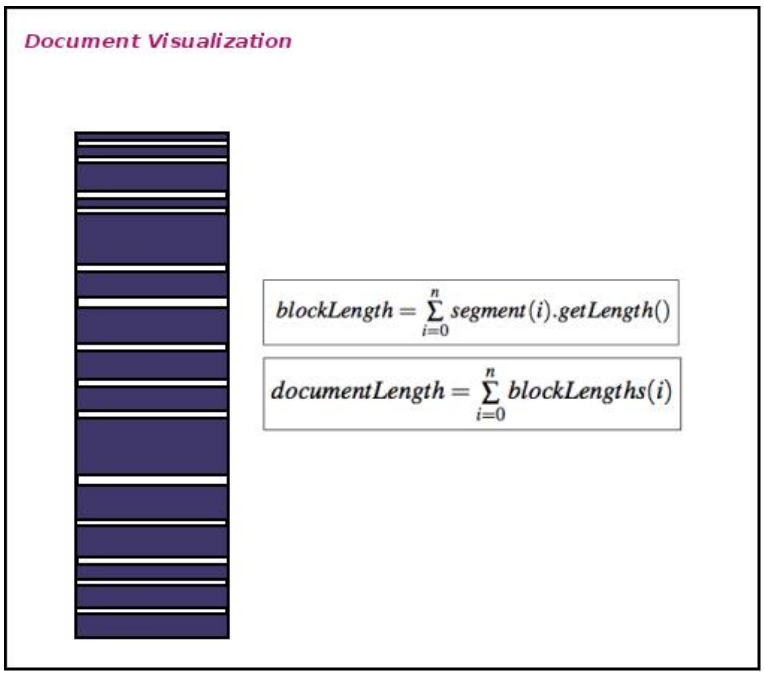
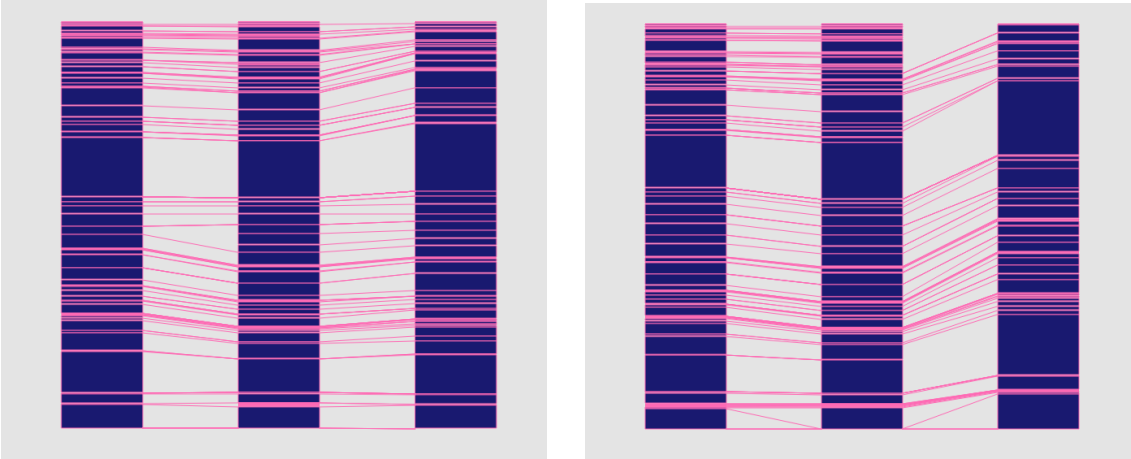


Figure 36: Illustration of Document Visualization Implementation.

The Alignments Visualization:

The Alignments Visualization is part of the Parallel Text Visualization, and is created by the same class. The Alignments Visualization draws the lines between the translations and the base-text. The implementation of this visualization is performed during the implementation of the Parallel Text Visualization. During the drawBlocks method, Qhash tables are created to store the segments Ids and their coordinates values. Qhash is a Qt class, which generates a dictionary to store the information on key and value format. These tables are used to create the alignments between the documents and they are also



(a) Parallel Text Visualization 1. (b) Parallel Text Visualization 2.

Figure 37: Screen captures of Parallel Text Visualization.

used later in the VisInfoTable class, which shows the information used in alignments visualization.

Three QHash tables are created from information based on the alignments part of each document. These tables are generated when the user selects a document from the list, and they are:

1. Table (1): is to store the version segments ids of the selected translation and the corresponding base-text segments ids. This table stores all the alignments imported from the alignments part in the selected document.

2. Table (2): is to store the version segments ids of the selected translation and the Y coordinate values of where these segments have been painted on the widget in parallel text visualization.

3. Table (3): is to store the base-text segments ids and the Y coordinate values of where these segments have been painted on the widget in parallel text visualization.

These tables are used as the follows:

1. Start with table number 1 and get the version segments id.

2. Get the Y coordinate of this segment from table number 2.
3. Get the corresponding base-text segment id for this version segment id from table number 1.
4. Get the Y coordinate of base-text segment id from table number 3.
5. Draw the alignment lines between the correspondence segments based on this information.
6. This process is repeated until all the alignments are drawn.

The result of the alignments is shown in figure 37a and figure 37b.

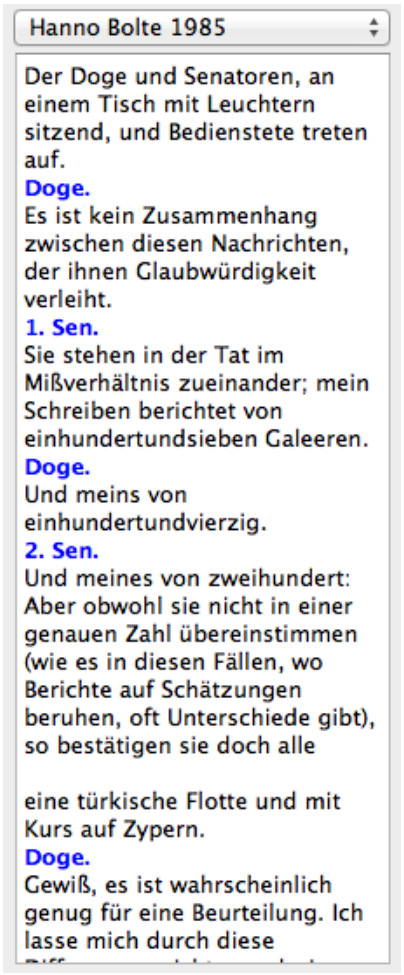
8.1.3 The Actual Text Visualization

In the main user interface, actual text of the document selected by the user tis allowed o be displayed in a text browser. The user can select the document from a drop-down list. The drop-down list items are filled directly after the user selects the data source file. This is performed by counting the number of documents in the file and adding the documents referring to the authors' names and the reference dates as items.

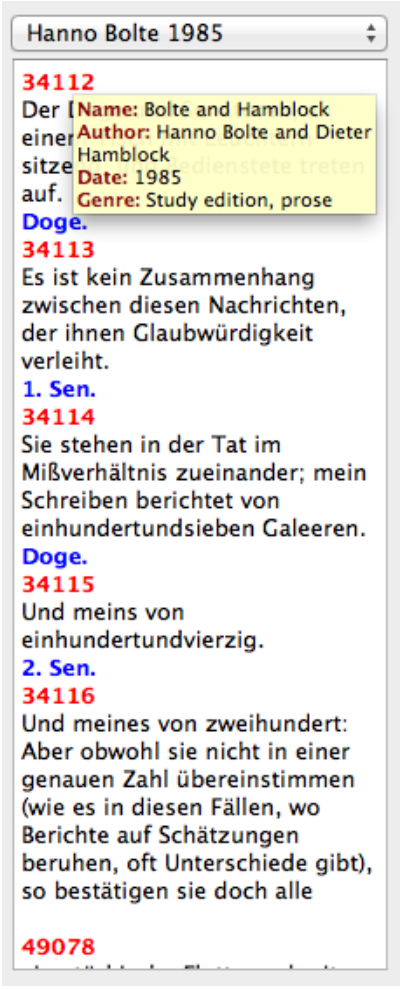
The text is displayed according to the segments in the document. The segment speaker and the segment text of all the document's segments is extracted and displayed on the text browser relative to their order in the document. There is an additional option that enables the user to show and hide segments ids as a part of the text. Moreover, additional information about the document is presented in a tooltip that is shown when the mouse moves over the text browser, presented in figure 38b.

8.1.4 Basic User Interaction Options

In this section, the user interaction options implemented in our software are illustrated. The main user interface is shown in figure 39, followed by a detailed description of all of its component.



(a) Actual Text Visualization 1: Shows the segments text and the speaker (in blue).



(b) Actual Text Visualization 2: Shows the segments text, the speaker (in blue) and the segments id (in red).

Figure 38: Actual Text Visualization

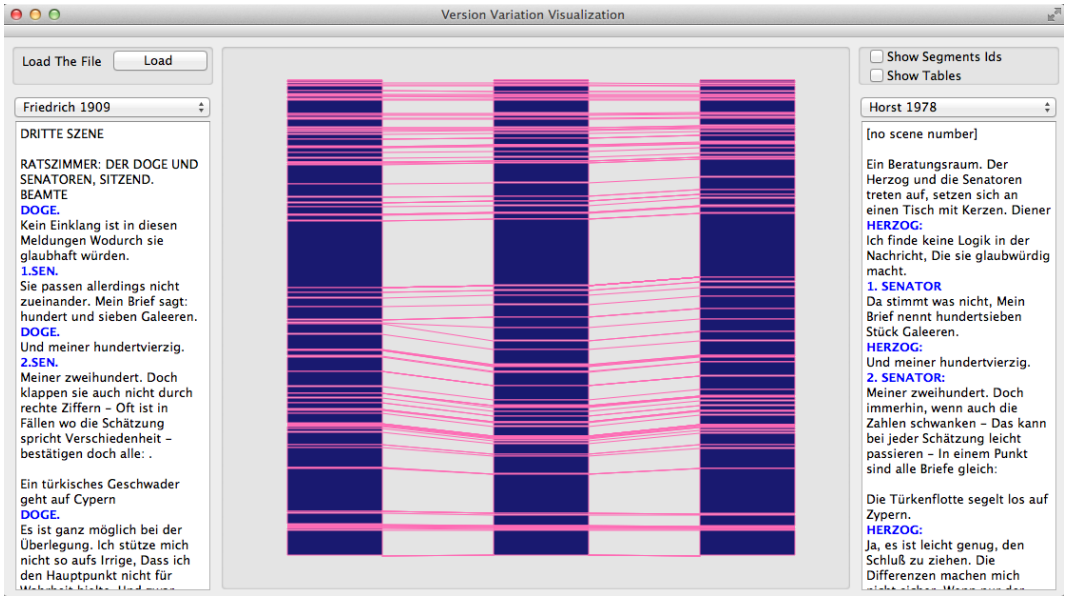


Figure 39: A screen capture of the User Interface.

1. Load Button: this push button starts the process of reading the data.

2. Drop-down List: there are two drop-down lists to enable the user to select a document, either base-text or translation, to display its content and its visualization.

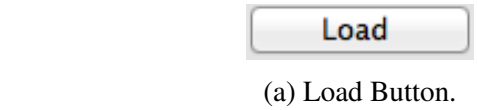
3. Scroll Area: allows the user to scroll up and down to view the text presented in the text browser.

8.2 Additional Features

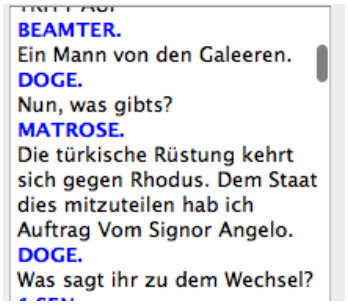
The main additional feature for the visualization is the ability to visualize more than two documents in parallel. As dicussed previously, currently three documents are visualized in parallel at the same time. In addition, the user is allowed to read and visualize all the documents in the XML file, which is composed of thirty-nine documents. These documents are the original English text and the thirty-eight German translations of that text. Beside this additional feature, the following user interaction options are implemented.

8.2.1 Additional User Interaction Options

The additional User Interaction Options are displayed in the following:



(a) Load Button.



(b) Scroll Area of the text browser.

(c) Drop-Down List that contains the list of the documents

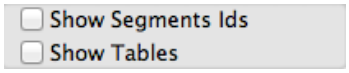
Figure 40: Screen captures of the Basic User Interaction Options.

1. Show segments ids checkbox: allows the user to show/hide the segments ids in the text browser. Each segment id is displayed before the segment text.

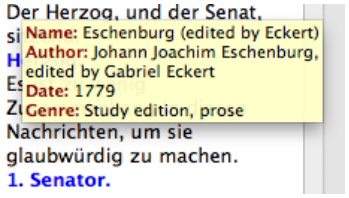
2. Show Tables checkbox: allows the user to show the visualization information tables, a description of these tables is in the verification of correctness section.

3. On the Mouse-Moves-Over the text browser that views the document's content, a tooltip appears which displays the basic information about that document i.e. the document name, author translator, date, and genre. A screen capture of this tooltip is shown in figure 41b.

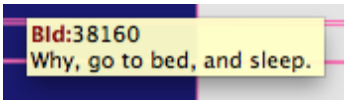
4. On the mouse moves over the visualization, a tooltip appears. The tooltip information is based on the cursor position. If the cursor moves over the visualization of the base-text document that is drawn in the middle of the visualization area, the tooltip will show the id and the text of the base-text segment that is painted in that Y coordinate. If the cursor moves over the visualization of the other two documents, the tooltip will show the id and the text of the translated segment that is painted in that Y coordinate in addition to the base-text segment id that corresponds to that translated segment.Screen captures of this tooltip are shown in figure 41b and figure 41d.



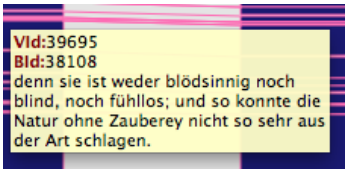
(a) Show/hide Checkbox and Show Tables Checkbox.



(b) Document Summary Tooltip.



(c) Visualization tooltip: when the cursor moves over the base-text visualization



(d) Visualization tooltip: when the cursor moves over the translation visualization

Figure 41: Screen captures of the Additional User Interaction Options.

9 Verification of Correctness

In order to verify the correctness of our visualization, two methods are proposed. The first is to show the visualization information tables that the visualization of the alignments is based on. The second is to show the visualized segment text and its id when the mouse moves over the visualized segment. In this section, the results of these two verification methods are presented.

9.1 Visualization Information Tables

The purpose of showing the basic information about the visualization is to know the correspondence segments i.e. the segment id of the selected document and the corresponding segment in the English document and the Y coordinates of each segment, which are used to verify that the alignments drawn between the documents are correct. This information is limited to the segments that exist in the Alignments section of the document data. VisInfoTable class is created to provide this information. The VisInfoTable class, with its own GUI, inherits the QWidget Qt class in order to benefits from its features. It is designed to display the tables produced during the process of parallel text visualization and it provides the following:

- Allows the user to display the required table. The class provides the following three tables:

  1. Table (1) : is to show the version segments ids and the corresponding base-text segments ids. An example of this table is illustrated in table 5.
  2. Table (2) : is to show the version segments ids and the Y coordinate values of where these segments have been painted on the widget in parallel text visualization. An example of this table is illustrated in table 6.
  3. Table (3) : is to show the base-text segments ids and the Y coordinate values of where these segments have been painted on the widget in parallel text visualization.An example of this table is illustrated in table 7

  Focus is only on the Y coordinate because it is a variable value, while the X coordinate is fixed for all the segments in the document.

- Display the selected table.

Table 5: Visualization Information Table no.1: An example of the table that shows the version segment ids and the corresponding base-text segment ids.

Version Segment Ids - Base-Text Segment Ids Table	
Version Segment Ids (VIds)	Base-Text Segment Ids (BIds)
38565	37251
38129	37253

Table 6: Visualization Information Table no.2: An example of the table that shows the version segments ids and the Y coordinate values where these segments have been painted on the widget in parallel text visualization.

Version Segment Ids - Y Coordinate Values Table	
Version Segment Ids (VIds)	Y Coordinate Value)
35008	497.311
35009	497.618

However, the implementation of showing these tables on the screen is not fully complete and therefore they are not used for the verification process. Instead, the content of

Table 7: Visualization Information Table no.3: An example of the table that shows the base-text segments ids and the Y coordinate values where these segments have been painted on the widget in parallel text visualization.

Base-Text Segment Ids - Y Coordinate Values Table	
Base-Text Segment Ids (BIds)	Y Coordinate Values)
38117	498.107
38153	389.095

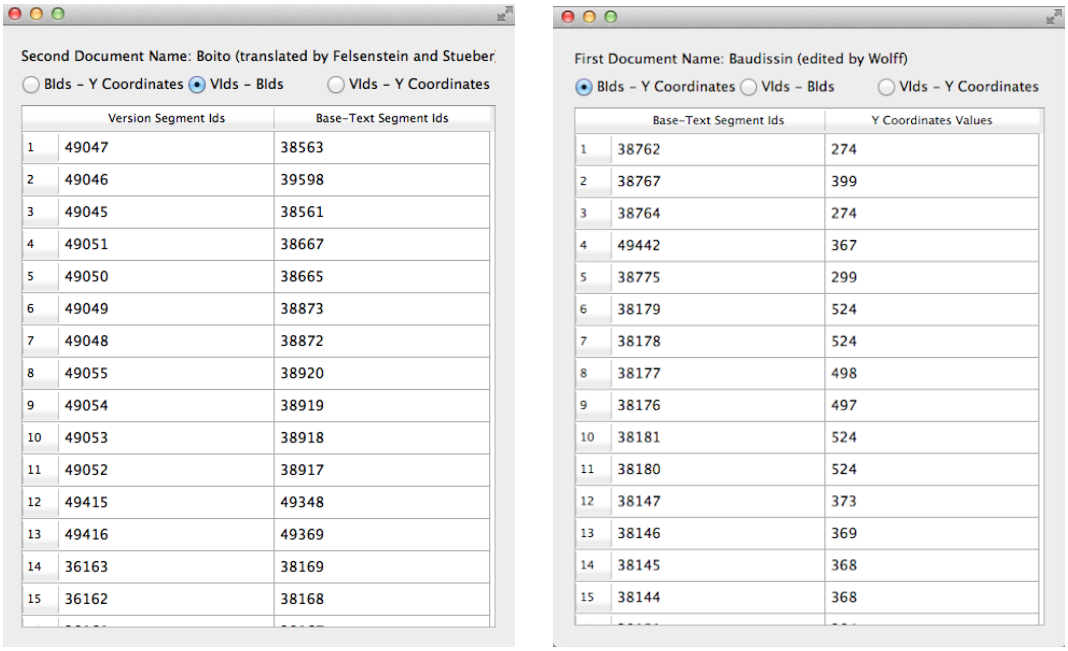
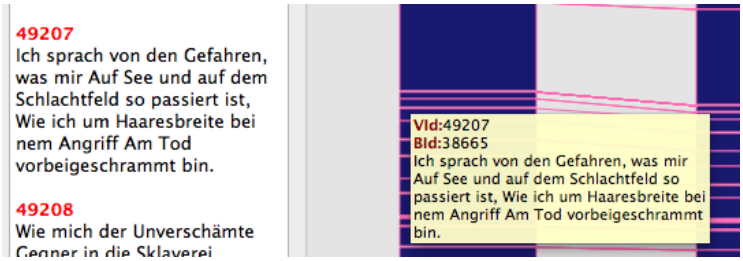


Figure 42: Screen captures of the Visualization Information Tables.

the tables is printed on the console for verification and the information of the visualization tooltip is used. Figure 42 shows screen captures of theses tables.

9.2 Information of the Visualization Tooltip

The information of the visualization tooltip displays the segment text, the segment id, and the corresponding segment id of the segment painted in the same position as the mouse. The implementation of this tooltip is discussed in the Project Implementation Section. This section presents screen captures of this tooltip from our software user interface and screen captures of the data source in order to show the conformity between the data and the visualization.



(a) The segment text of the segment with the id (49207) is the same for both the visualization and the text browser..

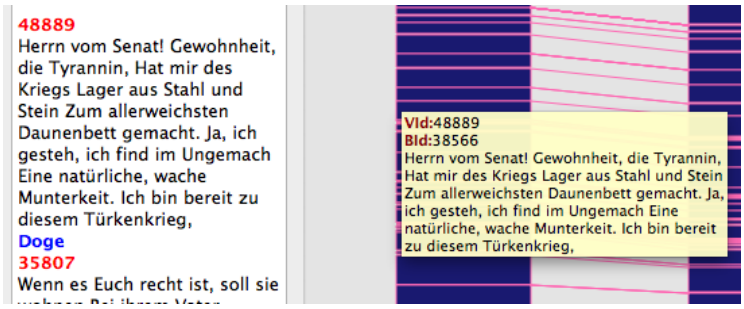
```
<br/>
<q data-eblatype="startmarker" data-eblasegid="49207">[</q>
Ich sprach von den Gefahren, was mir
<br/>
Auf See und auf dem Schlachtfeld so passiert ist,
<br/>
Wie ich um Haaresbreite bei nem Angriff
<br/>
Am Tod vorbeigeschrammt bin.
<q data-eblatype="endmarker" data-eblasegid="49207">]</q>
<q data-eblatype="startmarker" data-eblasegid="49208">[</q>
```

(b) The segment text of the segment with the id (49207) from the data source file.

```
<alignment id="13263" notes="" status="confirmed" BaseTextSegIds="38665" VersionSegIds="49207"/>
```

(c) The alignment information of the segment with the id (49207). It has the same corresponding base-txt segment id (38665) as presented in the tooltip.

Figure 43: The verification of the Visualization Correctness 1.Version name “Buhss” and authortranslator “Buhss”.



(a) The segment text of the segment with the id (48889) is the same for both the visualization and the text browser.

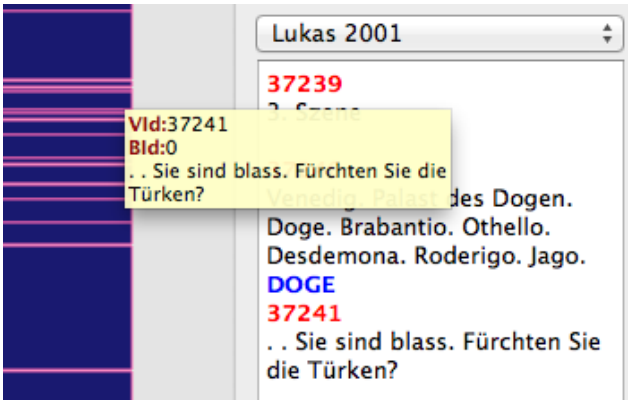
```
<q data-eblatype="startmarker" data-eblasegid="48889">[</q>
Herrn vom Senat! Gewohnheit, die Tyrannin,
<br/>
Hat mir des Kriegs Lager aus Stahl und Stein
<br/>
Zum allerweichsten Daunenbett gemacht.
<br/>
Ja, ich gesteh, ich find im Ungemach
<br/>
Eine natürliche, wache Munterkeit.
<br/>
Ich bin bereit zu diesem Türkenkrieg,
<q data-eblatype="endmarker" data-eblasegid="48889">]</q>
<br/>
```

(b) The segment text of the segment with the id (48889) from the data source file.

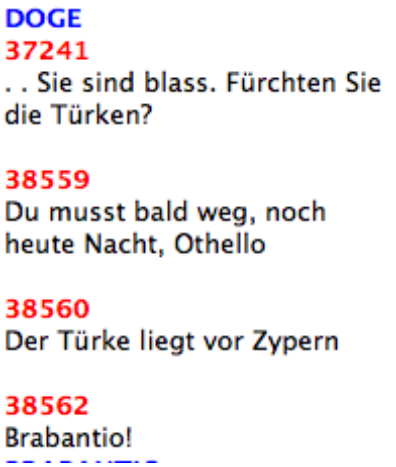
```
<alignment id="12968" notes="" status="confirmed" BaseTextSegIds="38566" VersionSegIds="48889"/>
```

(c) The alignment information of the segment with the id (48889). It has the same corresponding base-txt segment id (38566) as presented in the tooltip.

Figure 44: The verification of the Visualization Correctness 2. Version name “Fried” and authortranslator “Erich Fried”.



(a) The segment text of the segment with the id (37241) is the same for both the visualization and the text browser. This segment has no corresponding base-text segment id.



(b) The segment text of the segment with the id (37241). The segment (37241) has embedded segments.

```
▼<blockquote>
   <q data-eblatype="startmarker" data-eblasegid="37241">[</q>
   <q data-eblatype="startmarker" data-eblasegid="38559">[</q>
   Du musst bald weg, noch heute Nacht, Othello
   <q data-eblatype="endmarker" data-eblasegid="38559">]</q>
   .
   <br/>
   <q data-eblatype="startmarker" data-eblasegid="38560">[</q>
   Der Türke liegt vor Zypern
   <q data-eblatype="endmarker" data-eblasegid="38560">]</q>
   .
   <br/>
   <q data-eblatype="startmarker" data-eblasegid="38562">[</q>
   Brabantio!
   <q data-eblatype="endmarker" data-eblasegid="38562">]</q>
   Sie sind blass.
   <br/>
   Fürchten Sie die Türken?
   <q data-eblatype="endmarker" data-eblasegid="37241">]</q>
 </blockquote>
```

(c) The segment text of the segment with the id (37241) from the data source file. This segment has embedded segments. The embedded segments are presented in the text browser with their text and ids after the segment (37241). The data from the source file matches with the data from the visualization in the software.

Figure 45: The verification of the Visualization Correctness 3. Version name “Bärfuß” and authortranslator “Lukas Bärfuss”.

10 Conclusion

To conclude, data visualization is defined as “the use of computer graphics to illustrate information”. Nowadays, it is becoming a fundamental necessity in different fields due to the need to understand the massive growth in the amount of information. Although this project concentrates on visualizing textual data, its main goal is to develop an interactive visualization system for visualizing the different German translations of Shakespeare’s play, Othello. A group of researchers in the College of Art and Humanities at Swansea University has collected different German translations, which have evolved over a long period of time, of Shakespeare’s play, Othello. The aim of their research is to study the variations between these different translations and understand how the translations have differed throughout this period of time. Furthermore, how they vary from one author translator to another. The goal of the of the visualization of version variation project is to develop an interactive visualization system that applies effective visualization techniques in such a way that helps the researchers in the College of Art and Humanities to analyze, explore and identify the variations of this large textual data.

Working on this project was a great challenge for the author, especially considering that the field of data visualization was entirely new to her. It has been a very informative experience with many learning outcomes. Working with C++ programming language, the Qt GUI library, and following Bob’s Concise Coding Conventions was very helpful and contributed massively to enhancing the author’s programming skills.

For future development, Parallel Text Visualization could be applied to visualize all the segments of the document rather than the BlockQuotes; visualizing more than three documents in parallel and, in addition, to include more interaction options.

11 Supplementary Files

The URL below provides a video demonstration of the developed software and the Doxygen documentation of the source code.

http://cos-ugrad.swansea.ac.uk/715803/

References

[Blanchette and Summerfield, 2008] Blanchette, J. and Summerfield, M. (2008). *C++ Gui Programming with Qt 4, Second Edition*. Prentice Hall Press, Upper Saddle River, NJ, USA, second edition.

[Cheesman et al., 2012] Cheesman, T., Laramee, R., Hope, J., Flanagan, K., and Thiel, S. (2012). Version variation visualization @ONLINE. http://www.delightedbeauty.org/vvv/. [last accessed April 2014].

[Collins et al., 2009a] Collins, C., Carpendale, S., and Penn, G. (2009a). Docuburst: Visualizing document content using language structure. In *Proceedings of the 11th Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis'09, pages 1039–1046, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Collins et al., 2009b] Collins, C., Viégas, F. B., and Wattenberg, M. (2009b). Parallel tag clouds to explore and analyze faceted text corpora. In *IEEE VAST*, pages 91–98. IEEE.

[Daily Kos, nd] Daily Kos (n.d.). Daily Kos @ONLINE. http://dailykos.com/. [last accessed September 2014].

[Dörk et al., 2012] Dörk, M., Riche, N. H., Ramos, G., and Dumais, S. T. (2012). Pivotpaths: Strolling through faceted information spaces. *IEEE Trans. Vis. Comput. Graph*, 18(12):2709–2718.

[Dou et al., 2011] Dou, W., Wang, X., Chang, R., and Ribarsky, W. (2011). Paralleltopics: A probabilistic approach to exploring document collections. In *IEEE VAST*, pages 231–240. IEEE.

[Dou et al., 2013] Dou, W., Yu, L., Wang, X., Ma, Z., and Ribarsky, W. (2013). Hierarchicaltopics: Visually exploring large text collections using topic hierarchies. *IEEE Trans. Vis. Comput. Graph*, 19(12):2002–2011.

[Doxygen, nd] Doxygen (n.d.). Doxygen website @ONLINE. http://www.stack.nl/~dimitri/doxygen/. [last accessed April 2014].

[Fellbaum, 1998] Fellbaum, C., editor (1998). *WordNet: an electronic lexical database*. MIT Press.

[Furnas, 1986] Furnas, G. W. (1986). Generalized fisheye views. *SIGCHI Bull.*, 17(4):16–23.

[Geng et al., 2013a] Geng, Z., Cheesman, T., Laramee, R. S., Flanagan, K., and Thiel, S. (2013a). Shakervis: Visual analysis of segment variation of german translations of shakespeare's othello. *Sage: Information Visualization*.

[Geng et al., 2013b] Geng, Z., Laramee, R. S., and Cheesman, T. (2013b). Visualizing Translation Variation of Othello: A Survey of Text Visualization and Analysis Tools. Technical report, University of Wales, Swansea, UK, Department of Computer Science.

[Geng et al., 2011] Geng, Z., Laramee, R. S., Cheesman, T., Ehrmann, A., and Berry, D. M. (2011). Visualizing translation variation: Shakespeare's othello. In Bebis, G., Boyle, R. D., Parvin, B., Koracin, D., Wang, S., Kim, K., Benes, B., Moreland, K., Borst, C. W., DiVerdi, S., Chiang, Y.-J., and Ming, J., editors, *ISVC (1)*, volume 6938 of *Lecture Notes in Computer Science*, pages 653–663. Springer.

[Griffiths et al., 2004] Griffiths, T. L., Jordan, M. I., Tenenbaum, J. B., and Blei, D. M. (2004). Hierarchical topic models and the nested chinese restaurant process. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 17–24. MIT Press.

[Hoque and Carenini, 2014] Hoque, E. and Carenini, G. (2014). ConVis: A Visual Text Analytic System for Exploring Blog Conversations. *Computer Graphics Forum*, 33(3):221–230.

[IBM, nd] IBM (n.d.). Many Eyes By IBM @ONLINE. http://www-958.ibm.com/software/data/cognos/manyeyes/. [last accessed August 2014].

[Laramee, 2010a] Laramee, R. S. (2010a). Bob's concise coding conventions (C^3). *Advances in Computer Science and Engineering (ACSE)*, 4(1):23–26. (available online).

[Laramee, 2010b] Laramee, R. S. (2010b). Bob's Minutes of Meeting Protocol: Incentive and a Description @ONLINE. . [last accessed April 2014].

[Laramee, 2011] Laramee, R. S. (2011). How to read a visualization research paper: Extracting the essentials. *IEEE Computer Graphics and Applications*, 31(3):78–82.

[Lee et al., 2010] Lee, B., Riche, N. H., Karlson, A. K., and Carpendale, M. S. T. (2010). Sparkclouds: Visualizing trends in tag clouds. *IEEE Trans. Vis. Comput. Graph*, 16(6):1182–1189.

[Munzner et al., 2003] Munzner, T., Guimbretière, F., Tasiran, S., Zhang, L., and Zhou, Y. (2003). TreeJuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. SIGGRAPH 2003, pages 453–462. ACM Transactions on Graphics 22, 3.

[Pressman, 2010] Pressman, R. (2010). *Software Engineering: A practitioner's Approach*. McGraw-Hill higher education. McGraw-Hill Higher Education, seventh edition edition.

[Pressman, 2001] Pressman, R. S. (2001). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, New York, NY, fifth edition edition.

[Qt, nd] Qt (n.d.). Qt Project @ONLINE. http://qt-project.org/doc/. [last accessed September 2014].

[Sack, 2001] Sack, W. (2001). Conversation map: An interface for very large-scale conversations. *J. of Management Information Systems*, 17(3):73–92.

[Sarigiannidis and Chatzoglou, 2011] Sarigiannidis, L. and Chatzoglou, P. D. (2011). Software development project risk management: A new conceptual framework.

[Shapiro, 1971] Shapiro, S. C. (1971). A net structure for semantic information storage, deduction, and retrieval. In *Proc. 2nd Int. Joint Conf. Artificial Intelligence*, pages 512–523.

[Slashdot, nd] Slashdot (n.d.). Slashdot @ONLINE. http://slashdot.com/. [last accessed September 2014].

[Stroustrup, 1997] Stroustrup, B. (1997). *The C++ programming language (3. ed.)*. Addison-Wesley-Longman.

[Tufte, 2006] Tufte, E. R. (2006). *Beautiful evidence*. Graphics Press, Cheshire (Conn.).

[van Ham et al., 2009] van Ham, F., Wattenberg, M., and Viégas, F. B. (2009). Mapping text with phrase nets. *IEEE Trans. Vis. Comput. Graph*, 15(6):1169–1176.

[Viegas et al., 2006] Viegas, B., F., Golder, S., and Donath, J. (2006). Visualizing email content: portraying relationships from conversational histories. In *Proceedings of ACM CHI 2006 Conference on Human Factors in Computing Systems*, volume 1 of *Visualization 2*, pages 979–988.

[Ward et al., 2010] Ward, M., Grinstein, G., and Keim, D. (2010). *Interactive Data Visualization: Foundations, Techniques, and Applications*. A. K. Peters, Ltd., Natick, MA, USA.

[Ware, 2004] Ware, C. (2004). *Information Visualization, Second Edition: Perception for Design (Interactive Technologies)*. Morgan Kaufmann, 2 edition.

[Wei et al., 2010] Wei, F., Liu, S., Song, Y., Pan, S., Zhou, M. X., Qian, W., Shi, L., Tan, L., and Zhang, Q. (2010). TIARA: a visual exploratory text analytic system. In Rao, B., Krishnapuram, B., Tomkins, A., and 0001, Q. Y., editors, *KDD*, pages 153–162. ACM.

[Wenwen Dou and Xiaoyu Wang and Drew Skau and William Ribarsky and Michelle X. Zhou, 2012] Wenwen Dou and Xiaoyu Wang and Drew Skau and William Ribarsky and Michelle X. Zhou (2012). LeadLine: Interactive visual analysis of text data through event identification and exploration. In *IEEE VAST*, pages 93–102. IEEE Computer Society.

Appendices

A 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 31 October)				

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.

Signed: Date:

Minutes: Bob, Majedah 7th March 14 11:00

Next Meeting: Friday 14th March 14 11:00

Topic discussed:

- LATEX.
- Coursework submission.
- Large Vs Small document collections
- Concept Vs Implementation.
- Essential: concept, visualization technique used, application, single vs multiple documents.
- Direct and indirect text visualization.
- Table classification in the article [88]
- Visualization on the physical sciences
- Technology choices: QT.

Progress:

- Summary of visualization of translation variation.
- Start of the summary of chapter 9.
- Read "how to read a visualization research paper".

TODOD:

- 1 page summary of chapter 9.
- Always write in the present tense.
- Add Shakervis papers to the literature review [91]
- Read "Visualizing Variation in a Shakespeare Re-Translation Corpus".
- Submit initial document.

The above is only one meeting. "Bob's Minutes of Meeting Protocol" [Laramee, 2010b] is followed to document the meetings. The URL below contains all the other minutes of meetings.

http://cos-ugrad.swansea.ac.uk/715803/Minutes/

Appendices

B Appendix : DOXYGEN DOCUMENTATION

The first ten pages of class documentation chapter of Doxygen documentation is presented in the following pages. The URL Below contains the full Doxygen documentation:
http://cos-ugrad.swansea.ac.uk/715803/Doxygen/

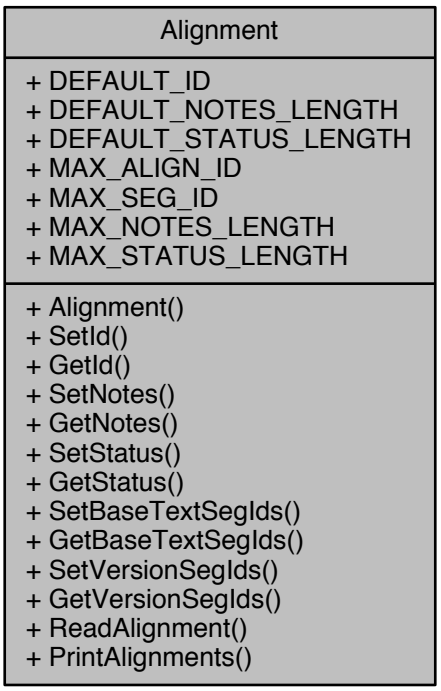
Chapter 1

Class Documentation

1.1 Alignment Class Reference

`#include <alignment.h>`

Collaboration diagram for Alignment:



Public Member Functions

- Alignment ()

  Alignment Constructor.
- bool SetId (int id)

SetId.

- int GetId ()

  GetId.

- bool SetNotes (QString notes)

  SetNotes.

- QString GetNotes ()

  GetNotes.

- bool SetStatus (QString status)

  SetStatus.

- QString GetStatus ()

  GetStatus.

- bool SetBaseTextSegIds (QString bsIds)

  SetBaseTextSegIds.

- QList< int > GetBaseTextSegIds ()

  GetBaseTextSegIds.

- bool SetVersionSegIds (QString vsIds)

  SetVersionSegIds.

- QList< int > GetVersionSegIds ()

  GetVersionSegIds.

- bool ReadAlignment (QXmlStreamReader &xml)

  ReadAlignment.

- void PrintAlignments ()

  PrintAlignments prints the information about the alignments.

Static Public Attributes

- static const int DEFAULT_ID = -1
- static const int DEFAULT_NOTES_LENGTH = 0
- static const int DEFAULT_STATUS_LENGTH = 0
- static const int MAX_ALIGN_ID = 99999
- static const int MAX_SEG_ID = 99999
- static const int MAX_NOTES_LENGTH = 999
- static const int MAX_STATUS_LENGTH = 999

1.1.1 Detailed Description

Definition at line 35 of file alignment.h.

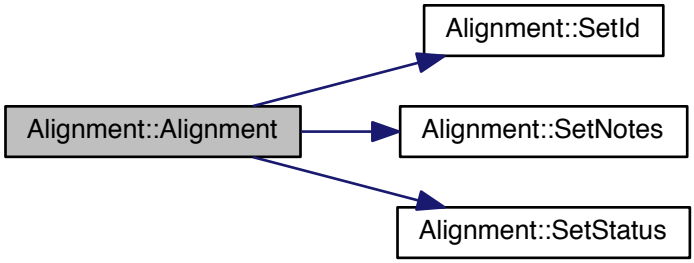
1.1.2 Constructor & Destructor Documentation

1.1.2.1 Alignment::Alignment ( )

Alignment Constructor.

Definition at line 9 of file alignment.cpp.

Here is the call graph for this function:



1.1.3 Member Function Documentation

1.1.3.1 QList< int > Alignment::GetBaseTextSegIds ( )

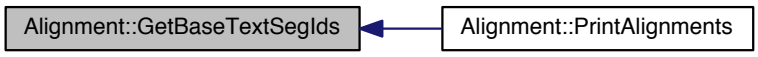
GetBaseTextSegIds.

Returns

a list of base text segment ids

Definition at line 131 of file alignment.cpp.

Here is the caller graph for this function:



1.1.3.2 int Alignment::GetId ( )

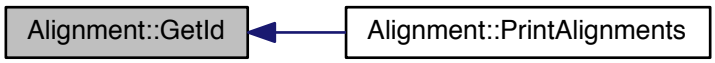
GetId.

Returns

the id of the alignment

Definition at line 37 of file alignment.cpp.

Here is the caller graph for this function:



1.1.3.3 QString Alignment::GetNotes ( )

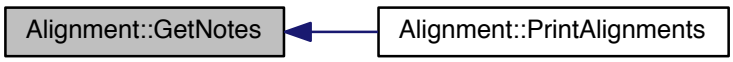
GetNotes.

Returns

notes

Definition at line 64 of file alignment.cpp.

Here is the caller graph for this function:



1.1.3.4 QString Alignment::GetStatus ( )

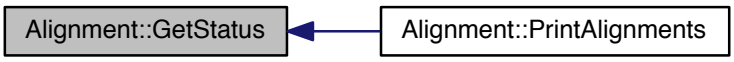
GetStatus.

Returns

status

Definition at line 91 of file alignment.cpp.

Here is the caller graph for this function:



1.1.3.5 QList< int > Alignment::GetVersionSegIds ( )

GetVersionSegIds.

Returns

list of version segment ids

Definition at line 171 of file alignment.cpp.

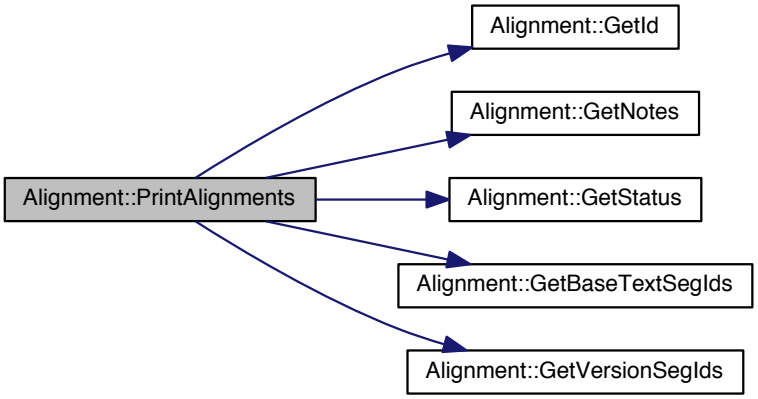
Here is the caller graph for this function:

1.1.3.6 void Alignment::PrintAlignments ( )

PrintAlignments prints the information about the alignments.

Definition at line 219 of file alignment.cpp.

Here is the call graph for this function:



1.1.3.7 bool Alignment::ReadAlignment (QXmlStreamReader & *xml*)

ReadAlignment.

Parameters

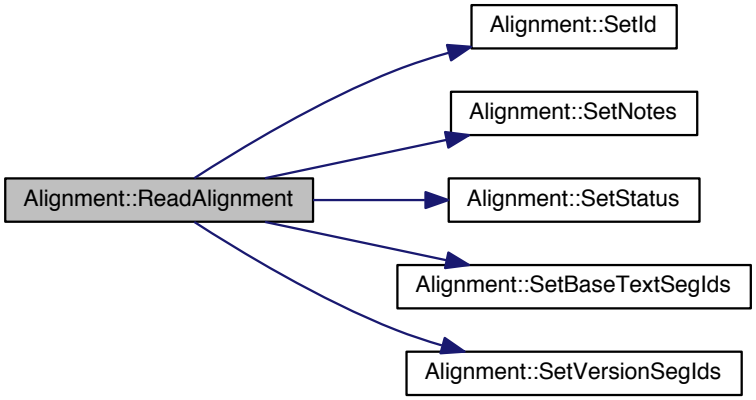
xml	

Returns

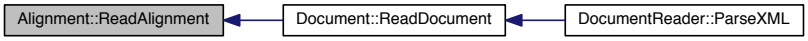
true on success

Definition at line 182 of file alignment.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



1.1.3.8 bool Alignment::SetBaseTextSegIds (QString *bsIds*)

SetBaseTextSegIds.

Parameters

list	of base text segment ids

Returns

true on success

Definition at line 101 of file alignment.cpp.

Here is the caller graph for this function:

1.1.3.9 bool Alignment::SetId (int *id*)

SetId.

Parameters

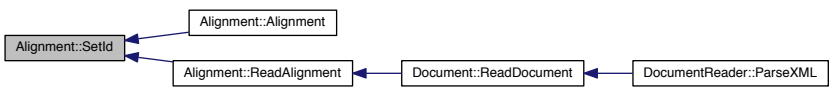
id	number of the alignment

Returns

true on success

Definition at line 22 of file alignment.cpp.

Here is the caller graph for this function:



1.1.3.10 bool Alignment::SetNotes (QString *notes*)

SetNotes.

Parameters

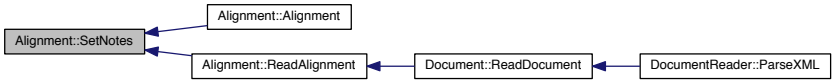
notes	

Returns

true on success

Definition at line 47 of file alignment.cpp.

Here is the caller graph for this function:



1.1.3.11 bool Alignment::SetStatus (QString *status*)

SetStatus.

Parameters

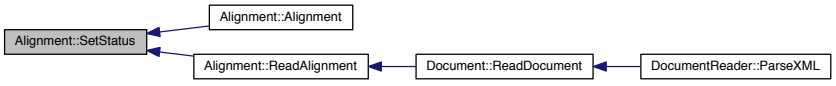
status	

Returns

true on success

Definition at line 74 of file alignment.cpp.

Here is the caller graph for this function:



1.1.3.12 bool Alignment::SetVersionSegIds (QString *vsIds*)

SetVersionSegIds.

Parameters

list	of version segment ids

Returns

true on success

Definition at line 141 of file alignment.cpp.

Here is the caller graph for this function:

1.1.4 Member Data Documentation

1.1.4.1 const int Alignment::DEFAULT_ID = -1 [static]

the default ID at initialization

Definition at line 109 of file alignment.h.

1.1.4.2 const int Alignment::DEFAULT_NOTES_LENGTH = 0 [static]

the default notes length at initialization

Definition at line 111 of file alignment.h.

1.1.4.3 const int Alignment::DEFAULT_STATUS_LENGTH = 0 [static]

the default status length at initialization

Definition at line 113 of file alignment.h.

1.1.4.4 const int Alignment::MAX_ALIGN_ID = 99999 [static]

the maximum possible alignment ID

Definition at line 115 of file alignment.h.

1.1.4.5 const int Alignment::MAX_NOTES_LENGTH = 999 [static]

the maximum possible notes length

Definition at line 119 of file alignment.h.

1.1.4.6 const int Alignment::MAX_SEG_ID = 99999 [static]

the maximum possible segment ID

Definition at line 117 of file alignment.h.

1.1.4.7 const int Alignment::MAX_STATUS_LENGTH = 999 [static]

the maximum possible status length

Definition at line 121 of file alignment.h.

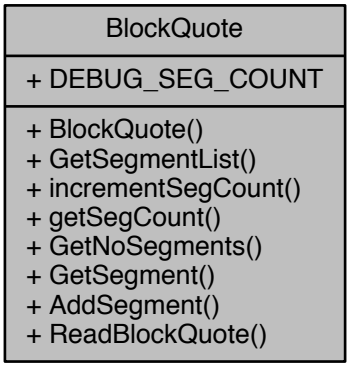
The documentation for this class was generated from the following files:

- /Users/majedahalrehiely/ReadingTranslations/alignment.h
- /Users/majedahalrehiely/ReadingTranslations/alignment.cpp

1.2 BlockQuote Class Reference

`#include <blockquote.h>`

Collaboration diagram for BlockQuote:



Public Member Functions

- BlockQuote ()

  BlockQuote constructor.
- QList< Segment * > & GetSegmentList ()

  getSegmentList
- bool incrementSegCount ()

  increment the current segment count -used for debugging
- int getSegCount ()
- int GetNoSegments ()

  GetNoSegments.
- Segment * GetSegment (int i)

  GetSegment gets a segment object from the list of segments of the BlockQuote by returning a pointer to that segment.
- bool AddSegment (Segment *segment)

  AddSegment add segment object to the segment list.
- bool ReadBlockQuote (QXmlStreamReader &xml)

  ReadBlockQuote.

Static Public Attributes

- static const int DEBUG_SEG_COUNT = 30

1.2.1 Detailed Description

Definition at line 36 of file blockquote.h.