THE UNIVERSITY OF
SYDNEY

A Data Mining Toolbox for Collaborative Writing Processes

By

Vilaythong Southavilay

THESIS

Presented to the Graduate School of Engineering at

The University of Sydney

in Fulfilment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF SYDNEY

2013

Thesis Supervisor: Dr. Kalina Yacef

ABSTRACT

Collaborative writing (CW) is an essential skill in academia and industry. The context of this research focuses particularly on collaborative forms of writing in an academic environment for the purpose of learning, in which writing activities are collaboratively performed by groups of students in a period of multiple writing sessions. CW combines the cognitive and communication requirements of writing with the social requirements of collaboration. Cognitive studies show that these requirements make CW a challenging endeavour. Providing support during the process of CW can be useful not only for achieving better quality documents, but also, more importantly, for improving the CW skills of the writers.

In order to properly support collaborative writing, it is essential to understand how ideas and concepts are developed during the writing process, which consists of a series of steps of writing activities. These steps can be considered as sequence patterns comprising both time events (as used in other process mining research) and the semantics of the changes made during those steps. Two techniques can be combined to examine those patterns: process mining, which focuses on extracting process-related knowledge from event logs recorded by an information system; and semantic analysis, which focuses on extracting knowledge about what the student wrote or edited.

This thesis contributes (i) techniques to automatically extract process models of collaborative writing processes and (ii) visualisations to describe aspects of collaborative writing. These two techniques form a data mining toolbox for collaborative writing by using process mining, probabilistic graphical models, and text mining. First, I created a framework, WriteProc, for investigating collaborative writing processes, integrated with the existing cloud computing writing tools in Google Docs. Secondly, I created new heuristic to extract the semantic nature of text edits that occur in the document revisions and automatically identify the corresponding writing activities. Thirdly, based on sequences of writing activities, I propose methods to discover the writing process models and transitional state diagrams using a process mining algorithm, Heuristics Miner, and Hidden Markov Models (HMM), respectively. My thesis compares two models of HMM: a Heuristic Markov Model and a Hidden Markov Model. The discovered process models and

transitional state diagrams are used in the process analysis that quantitatively and graphically identifies patterns in the text edit sequences that were performed by the writers as they worked on their documents. Finally, I designed three types of visualisations and made contributions to their underlying techniques for analysing writing processes: (1) the *revision map*, which summarises the text edits made at the paragraph level over time during the course of the writing; (2) the *topic evolution chart*, which uses probabilistic topic models -- especially Latent Dirichlet Allocation (LDA) and its extension, DiffLDA -- to extract topics and follow their evolution during the writing process; (3) the *topic-based collaboration network*, which allows a deeper analysis of topics in terms of author contribution and collaboration, using a novel algorithm DiffATM in conjunction with a DiffLDA-related technique.

All components of the toolbox are validated against annotated writing activities of real documents and a synthetic dataset. I also illustrate how the automatically discovered process models and visualisations are used in the process analysis with real documents written by groups of graduate students. I discuss how the analyses can be used to gain further insight into how students work and create their collaborative documents; and ultimately to help students write more efficiently and effectively, and to assist teachers with monitoring writing groups, providing information that can facilitate early detection of problems during the writing process.

ACKNOWLEDGEMENTS

During the time of doing my research and writing this thesis, I received support and help from many people. I would like to express my great appreciation to the following people and resources, which are not listed in any particular order.

With all of my heart, I am very grateful to my supervisor Dr. Kalina Yacef, and to my associate supervisor Associate Prof. Rafael A. Calvo for all the guidance, constant encouragement and support that they gave me to complete this thesis. All the advice given by them has been a great help to me in my research. I would like to thank them for assisting me in getting an Australian Research Council scholarship and Google Research Award so that I could work on this project under their supervision during the past three years. In particular, I am profoundly indebted to my supervisor, who was very generous with her time and knowledge and assisted me with each step in completing this thesis.

I would like to offer my special thanks to all administrative staff members of the school of information technologies (IT), especially the head of the school, Prof. David Dagan Feng; the post graduate research director, Anastasios Viglas; and the post graduate research officer, Evelyn Riegler, for their support so that I could finish writing this thesis. I would like to thank all technical staff members of the school of IT, headed by Greg Ryan, for their technical assistance and support during the experiments conducted for my research.

My special thanks are extended to my colleagues at the school of IT, especially all researchers and students of Computer Human Adapted Interaction (CHAI) for sharing their ideas and experience in doing research. In addition, I would like to thank my fellow friends in Learning and Affect Technologies Engineering (LATTE), especially Dr. Jorge Villalon, Stephen O'Rourke, and Dr. Liu Ming for their help with text mining library (TML), Google Document List API and for their advice on this research and the experiments.

I am grateful to thank Prof. Michael J. Jacobson for his understanding of my circumstances and his generosity in giving me an opportunity to work part-time for a project that investigates the learning of scientific knowledge about climate change through computational models at the Centre for Research on Computer Supported Learning and Cognition (CoCo). In addition, I am thankful for the experience of

working closely with Dr. Lina Markauskaite, who assisted me in developing my research skills. I also would like to thank my colleagues Nick Kelly, Kate Thompson, and Poly for their technical and administrative support for the project.

I would like to acknowledge the help provided by Prof. Peter Reimann and Anindito Aditomo (Nino) in allowing me to make use of their course so that I could conduct experiments for my research. I also appreciate their assistance in preparing materials for the experiment such as mock-up visualisations and questionnaires; and for distributing visualisations of writing processes to students. In addition, they provided me with very valuable comments and suggestions with regard to the visualisations.

I would like to acknowledge the Australian Research Council DP0986873 and a Google Research Award for funding my research.

I am thankful to Maria Cristina Beato-Lanz for her wonderful assistance in proofreading all chapters in this thesis, except chapter 3 (theoretical background) in a timely manner. She also proofread two appendices (B and E). I really appreciate her assistance.

I am grateful to my mother, who was the first teacher who taught me how to read and write a word. Although she passed away about seven years ago and was not here to see me complete this PhD thesis, she is always my inspiration that leads me to work hard and to love science and technologies.

Special thanks is reserved for my wife who has been travelling and lived with me during my candidature in Sydney for three years. After the birth of our twin sons, we had to make a hard decision to separate for about six months so that I could work on this thesis, and she travelled back to my home country where she has lot of support with looking after our children. I will join her and my children after I submit this thesis.

I am grateful to all my family, especially my father and my parents-in-law, for providing financial support and assisting my wife in taking care my children while I am in Sydney writing this thesis.

Finally, I would like to dedicate this thesis to all scientists, researchers and educators from developing countries, especially from my home country, Laos. Since I was young, I always dreamed of becoming a scientist. I was awarded an AusAID scholarship to study for my bachelor's degree in Canberra, Australia. After that I was

awarded a Fulbright scholarship to study for my master's degree in Chicago, Illinois before returning to Australia to pursue my PhD in Sydney. Along the way, I have worked really hard to fulfil my dream to become a computer scientist. I hope my journey will inspire many young generation students in their developing work to seek an opportunity to work in the field of science and technologies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

The availability of the Internet has made collaborative writing very easy to implement in schools and at work. One result of this circumstance has been the development of new forms of writing, such as blogging and wiki writing. In addition, the emerging of "cloud computing" tools and Web 2.0 applications, such as Google Docs, have led to the creation and access of near desktop-quality online writing environments.

Writing can be used not only as a method of acquiring better writing skills, but also as an important tool for learning subject matter (Bereiter & Scardamalia, 1987; Galbraith, 1999). The context of this research focuses particularly on collaborative forms of writing in an academic environment for the purpose of learning, in which writing activities are collaboratively performed by groups of students in a period of multiple writing sessions. Lowry et al. (2003) describes collaborative writing (CW) as "..an iterative and social process that involves a team focused on a common

objective that negotiates, coordinates, and communicates during the creation of a common document". Cognitive studies demonstrate that CW presents a challenge with regard to all these aspects: negotiation, coordination, and communication (Flower & Hayes, 1981). Because of the complexity of the CW process, both explicit and scaffolding support need to be provided; these two types of support generally fall into one of three classes: Specialised writing and document management tools; document analysis technologies; and team process support. This thesis addresses the last two of these three classes, as the first one is provided by commercial vendors (e.g. Google) who provide the writing tools and store the documents written by students.

Even though the use of cloud computing tools, such as the collaborative writing tool Google Docs, is spreading in workplaces and classrooms, CW is not explicitly taught in school or higher education systems. Providing support on the processes of CW can be useful not only for improving the quality of the documents produced by this process, but also – and more importantly – for improving the CW skills of those involved. This research posits that in order to effectively support Higher Education students in writing together and learning from the collaborative writing process, it is necessary to develop computational support to provide visualisations of and/or feedback with regard to the students' activities during the process. The visualisations of and/or feedback on writing processes can be analysed in order to extract the patterns of text edits and writing activities that are performed during the course of writing. Analysing the discovered patterns of writing activities can assist in understanding of how certain sequence patterns of writing activities (i.e. the steps followed by a group of authors) lead to high quality outcomes and sequence patterns that may lead to low quality outcomes.

Computer-supported writing has been studied for decades in the field of Education. After ten years of collecting empirical data, Goldberg et al. (2003) found in a meta-study "that when students write on computers, writing becomes a more social process in which students share their works with each other". The study also noted that when using computers, students prefer to make revisions while producing text, rather than afterwards; they also tend to make more revisions between initial and final drafts and to produce longer passages.

Review feedback, especially from peer reviews, has been recognized as another effective tool for learning writing (Carlson & Berry, 2008; Cho & Schunn, 2007). When students use computers to write, they engage in the revising of their work throughout the writing process; they more frequently share and receive feedback from their peers; and they benefit from teacher input earlier in the writing process than they do when writing manually. Although the studies show that computer-supported writing, including automatic feedback tools, efficiently assists students in writing and reviewing, it is still crucial to further understand the writing process in order to develop support technologies for CW.

In order to properly support collaborative writing, it is essential to gain an understanding of how ideas and concepts are developed during the writing process, which consists of steps of writing activities. These steps can be considered as sequence patterns comprising both time events and the semantics of changes made during those steps.

Therefore, there are two effective techniques that can be combined and used to obtain insight into students' collaborative writing: Process mining, which focuses on extracting process-related knowledge from event logs recorded by an information system; and semantic analysis, which focuses on extracting knowledge about what the student wrote (or edited). The field of process mining covers many areas, such as process discovery (discovery of the control flow), performance characteristics (e.g. throughput times), process conformance (checking if the event log conforms to specifications), and social networks (e.g. collaboration) (Bozkaya et al., 2009; van der Aalst, 2011). Process mining analysis, in particular, is necessary to understand group awareness and writers' participation and collaboration. Text mining combines indexing, clustering, latent semantic analysis (Landauer et al., 2007) and several probabilistic topic modelling techniques (Blei & Lafferty, 2009).

For two decades, process mining techniques have been successfully applied to extract process-related knowledge from event logs recorded by business information systems; however, the techniques have only recently been applied to educational data. For example, Pecheniskiy et al (2009) used process mining tools to analyse data from online multiple choice examinations and demonstrated the use of process discovery and analysis techniques; but the area of interest in this research was the individual students' activities related to answering online multiple-choice questions

during assessment, not student activities related to writing and editing texts collaboratively.

There is prolific disparate research in text mining with regard to improving support for quality writing, such as tools for automatic scoring of essays (Shermis & Burstein, 2003), visualisation of documents (O'Rourke et al., 2011; Villalon & Calvo, 2011), automatic question generation (Liu & Calvo, 2011) and document clustering (Andrews & Fox, 2007); however, these existing approaches all focus on the final product, unlike the work in this research which examines the writing process itself in an effort to provide insight on how students write their documents.

In addition to the above studies, one important benefit of cloud computing tools, beyond allowing authors to edit text anywhere at any time and to collaborate seamlessly, is their capacity to store all the document revisions and revision histories (i.e. timestamps and authorship), providing unprecedented historical data of all the text edits made by authors as they write. By exploiting this data, researchers can gain insight into the processes that authors follow to write their documents, and investigate and extract information about collaborative writing that may prove useful for teachers and students.

Abundant research has recently been conducted with regard to exploiting Wikipedia's revision history for several tasks, such as Natural Language Processing applications (Ferschke et al., 2013), including a number of studies that were particularly interested in analysing the lifecycles and evolution of Wiki articles. The article evolution extraction was based on human evaluated quality classes in Wikipedia, aiming mainly at automatically assessing the quality and trustworthiness of the articles. Although these techniques can be applied to extract collaborative writing processes not only for Wikipedia articles but also for any jointly-authored documents, the methods used in these studies do not provide adequate means for coding the writing behaviours logged in the revision histories or to sequentially analyse the collaborative writing processes observed. For this reason, these techniques can not be used directly in supporting collaborative writing.

When students engage in collaborative writing processes, they produce higher quality text (Palincsar & Brown, 1984; Scardamalia & Bereiter, 1996). However, research shows that groups tend to choose approaches that result in members working more on an individual than a collaborative basis (Ede & Lunsford, 1992). In

order to understand why groups tend to write individually rather than collaboratively and to discover the factors that affect group collaboration on writing tasks, it is crucial to employ techniques that identify writing activities and that model the actual sequences of these activities.

To support collaborative writing skills, feedback about the writing processes can be provided to students and teachers in the form of mirroring visualisations (Erickson et al., 1999; Kay, Maisonneuve, Yacef, & Reimann, 2006; Upton & Kay, 2009) which provide an awareness of the group's writing activities to individual students, thus enabling them to perform their collaborative writing tasks more efficiently and effectively. In addition, teachers can use the support as tools to help them monitor groups effectively and detect problems early.

In summary, providing support on the processes of CW can be useful not only for improving the quality of the documents produced by this process, but also – and more importantly – for improving the CW skills of those involved. In order to properly support collaborative writing, it is essential to gain an understanding of how ideas and concepts are developed during the writing process. Although there is unprecedented historical data including all the document revisions and revision histories (i.e. timestamps and authorship), provided by cloud computing tools, there are problems with existing techniques for investigating into the development of ideas and concepts during the course of collaborative writing:

- The existing approaches all focus on the final product of writing, not on the process of writing.
- There is no adequate ways for coding and automatically identifying the writing behaviours logged in the revision histories; and sequentially analysing the collaborative writing processes observed.
- There is no appropriate feedback and/or visualisation for analysing the development of ideas and concepts during collaborative writing processes and mirroring the group's writing activities to individual students and teachers.

1.1 Summary of Contribution

This thesis aims to develop techniques that automatically extract process models of writing processes and provide visualisations that describe aspects of students' collaborative writing. The outcomes of this work are the following:

- To identify the text features -- e.g. text editions, topics, and cohesion -- that can be used to detect the purpose of text edits made to a document.
- To extract the corresponding collaborative writing activities or events based on these text edits; the theories of cognitive models of writing processes; and the taxonomy of collaborative writing activities and revisions.
- To create techniques for building a range of process models and representation by using those collaborative writing activities which provide different views of the collaborative writing processes.
- To design several visualisations that reflect important syntactic and semantic changes made to a document during the writing process.

1.2 Outline of the Chapters

This chapter provides an introduction to this thesis, describes the overview and motivation for the research, and outlines the contributions made by the research.

Chapter 2 reviews the current literature on the theoretical framework of cognitive models of writing processes, taxonomies of collaborative writing activities, models of analysing revisions, related process mining, and text mining works.

Chapter 3 reviews the theoretical background of the text mining and process mining algorithms and techniques used in this thesis.

Chapter 4 explains the approach of this thesis, consisting of a framework for extracting revisions and revision histories; and methods for automatically identifying writing activities, extracting process models, and visualising collaborative writing processes.

Chapter 5 introduces WriteProc, the framework for extracting revisions and revision histories, provides an overview of the framework, the writing environment, along with the approach for extracting revisions and revision histories, and describe the dataset used in the following chapters: 6 and 7.

Chapter 6 presents my techniques for automatically identifying collaborative writing activities, and the validation of these techniques, using real documents written by groups of graduate students.

Chapter 7 presents my techniques for extracting process models: causal dependency diagrams and transitional state diagrams by using process mining algorithms and hidden Markov models, respectively. It also provides the process

analysis of writing processes of real documents written by groups of students outlined in the studies in Chapter 5.

Chapter 8 presents my techniques for visualising collaborative writing processes; introduces three visualisations; demonstrates the validation of the techniques used for producing the visualisations based on a synthetic dataset; and provides a prototype experiment to illustrate how the visualisations are used, what information they provide, and whether they are useful.

Chapter 9 discusses the limitations of the techniques presented herein and offers suggestions for future research, as well as summarizing the approach of this thesis.

1.3 Publication Related to This Thesis

This thesis is derived from the following publication:

Southavilay, V., Yacef, K., & Calvo, R. A. (2009). *WriteProc: A Framework for Exploring Collaborative Writing Processes.* Paper presented at the Australasian Document Computing Symposium, Sydney, Australia. – incorporated in Chapter 5.

Southavilay, V., Yacef, K., & Calvo, R. A. (2010). *Process Mining to Support Students' Collaborative Writing.* Paper presented at the the third International Conference on Educational Data Mining, Pittsburgh, PA, USA. – incorporated as in Chapter 6 and 7.

Southavilay, V., Yacef, K., & Calvo, R. A. (2010b). *Analysis of Collaborative Writing Processes Using Hidden Markov Models and Semantic Heuristics.* Paper presented at the Proceedings of the third International Workshop on Semantic Aspect of Data Mining, Sydney, Australia. – incoporated as in Chapter 7.

Southavilay, V., Yacef, K., Reimann, P, & Calvo, R. A. (2013*). Analysis of Writing Processes Using Revision Maps and Probabilistic Topic Models.* Paper presented at the Proceedings of the third International Conference on Learning Analytics and Knowledge, Leuven, Belgium. – incorporated as in Chapter 8.

CHAPTER 2
LITERATURE REVIEW

The subject of Computer-Supported Collaborative Work (CSCW), particularly Collaborative Writing (CW), has received attention since computers were first used for word processing. The ever increasing availability of the Internet has resulted in a corresponding increase of people writing collaboratively by sharing their documents in a number of ways. Relatedly, since writing individually and collaboratively are considered essential skills in most industries, academia, and government, there has also been an increase in research on how to support the production of better documents.

Over the past two decades, an abundance of text-mining research has been conducted with the purpose of improving the support of quality writing. Shermis and Burstein (2003) described four different methods, as follows: (1) Project Essay Grade (PEG), which used a large collection of surface features such as instances of average sentence length, frequency of certain transitional words, number of

semicolons, and work rarity; (2) the Bayesian approach, which examined the probabilities of each token (typically a word or a stemmed word) being used in essays in each score group; (3) Intelligent Essay Assessor (IEA), which examined content, style, and mechanics, with content expressed as independent measures of semantic quality and the amount of such content; and (4) e-rater, which examined discourse structure, syntactic structure, and vocabulary usage. The PEG and Bayesian approaches are simpler to develop for real applications, although IEA and e-rater have much deeper linguistic features.

Another active section of research in improving the support of quality writing is the area of automatic question generation, exemplified by the work of Liu and Calvo (2011), which used text-mining and natural language processing techniques to provide feedback as types of questions to students based on their documents (i.e. literature reviews). In addition, the works of O'Rourke et al (2011) and Villalon and Calvo (2011) concentrated on visualising the cohesion of texts and concept maps of students' essays, using text mining techniques respectively. Nevertheless, these studies all focus on the final product, not on the writing process itself. An investigation of how ideas and concepts are developed during the actual process of writing could be used to improve not only the quality of the documents but also -- and more importantly -- the writing skills of those involved.

Analysing the process of writing requires an understanding of how certain sequence patterns (i.e. the steps followed by a group of writers) lead to high quality outcomes. The sequence patterns of writing processes are comprised of time events (as used in other process mining research) and the semantics of the changes made during that step. Two techniques can be combined to examine the patterns: Process mining, which focuses on extracting process-related knowledge from event logs recorded by an information system; and semantic analysis, which focuses on extracting knowledge about what the student wrote or edited.

This chapter begins with a discussion of the theoretical framework of the cognitive models of writing processes and discusses the taxonomy of collaborative writing and models of analysing revisions, followed by a presentation of related works in the fields of text mining and process mining. It then reviews tools to support collaborative writing.

2.1 Theoretical Framework

2.1.1 Cognitive Models of Writing Processes

In order to understand the writing process, it is important to review the theoretical frame of cognitive models of writing processes and the taxonomy of collaborative writing activities and revisions.

There are three classic cognitive models of writing processes: Knowledge-telling, knowledge-transforming, and knowledge-constituting models (Galbraith, 2009). The first original model, proposed by Hayes and Flower (1980), was developed based on an experiment with writers thinking aloud as they wrote. This original model considers a writing process as an idea generation process that retrieves content from long-term memory by using resources from the writing task environment (including the writing assignment and the text produced so far) and from the writer's long-term memory, in which a "*monitor*" or central executive is responsible for deciding which activities (tasks) should be carried out and when (Galbraith, 2009).

According to this model, in order to work out what to write next about a topic, the writer starts by using the specification in the writing assignment to construct a set of cues with which to probe long-term memory. If content is successfully retrieved and then positively evaluated, it is then either transformed into text immediately on paper or stored mentally in the memory for later translation. This content then acts as a new probe for memory, so that each retrieval episode consists of associated chains of content being retrieved from memory. If appropriate content cannot be retrieved, the "monitor" has to decide what to do next; it may decide to pursue a different goal -- for instance, to read more books about the topic, or to read the assignment more closely – or it may also carry on generating content by probing memory again with a different set of cues. It is important to note that the central part of this model is the "monitor", which controls the decision process and the writer's overall writing strategy. Because the content was retrieved periodically from memory, this original model was called the "knowledge-telling" model (Bereiter & Scardamalia, 1987).

In 1987, Bereiter and Scardamalia extended the original model of idea generation and proposed the "knowledge-transforming" model of writing. Although this model also includes the same basic mechanism of generating ideas (i.e. the retrieval of

content from long-term memory), it is focused on the rhetorical nature of goals towards which the writing is directed. The model claims that the writing process is not simply an evolution of a knowledge telling, but also a redefinition of writing goals, in which content is formulated as the text develops. In other words, according to this model, content is not only retrieved in response to a more elaborated representation of the assignment as a rhetorical problem, but it is also formulated in the context of, and as a contribution to, the series of rhetorical acts that emerge gradually in the text.

Although this knowledge-transforming model captures important features of the writing process, Galbraith argued that it has two problems (Galbraith, 2009). First, although one of the attractive features of the model is the claim that it accounts for the common experience of writing as a source of discovery, this is only implied in the model, and was not directly tested during its development. Second, the knowledge-telling model, which is embedded within the knowledge-transforming model as its account of how content is generated to satisfy goals, does not explain how novel content is formulated during writing. According to the knowledge-transforming model, generating an idea is a matter of accessing pre-existing content in the memory. Although this can account for the fact that the content retrieved is different when the rhetorical context drives a memory search than it is when retrieved associatively, it does not explain how new content that develops the writers' understanding is generated. Consequently, the idea implied by the model that writing develops understanding cannot be justified without empirical testing.

Subsequent research has investigated this more directly, and examined the condition under which writers discovered new ideas through writing. In 1999, Galbraith presented writing as a knowledge constituting process. Rather than presenting knowledge as static data stored in a memory system (as assumed by Hayes and Flower), Galbraith considered that language production draws on a different semantic memory system, which is represented by a network of units analogical to neuron networks. This network is flexible in that units have different patterns of activation for different inputs, and only develop these patterns of activation in the presence of a particular input. Galbraith's model did not include a long-term memory unit or a monitor, like the two models previously discussed. According to the knowledge constituting model (Galbraith, 1999), the author's

semantic memory is represented by the network of units, and the author's knowledge is represented by the weights connecting the units. Galbraith did not deny that content sometimes represents individual events which can be accessed via a process of retrieval, a component which is similar to the principles of the Hayes and Flower model; and also similar to the knowledge-transforming model characteristics, the inputs in Galbraith's model consist of a specification or a goal, and a series of rhetorical acts that gradually emerge in the text. However, the weights of connections and the activation of the units in the network of the knowledge constituting model were reconfigured through a series of learning mechanisms similar to the neuron networks of humans.

2.1.2 Taxonomies of Collaborative Writing Activities

In addition to the above cognitive models, writing processes can also be described by using a taxonomy of writing activities, as proposed by Lowry et al. (2003). This taxonomy of group writing activities that occur in collaborative writing fall into six categories, as follows (Lowry et al., 2003):

1. Brainstorming: Developing new ideas for a paper draft.
2. Outlining: Creating a high-level direction in which the document will be going, including major sections and subsections.
3. Drafting: Writing the initial incomplete text of a document (this is typically synonymous with the term "writing", but the term "drafting" is used to convey incompleteness in the writing).
4. Reviewing: Having a participant or an editor read and annotate the document draft section for improvements in content, grammar, and style.
5. Revising: Responding to the above comments by making changes in the draft that reflect the feedback provided in the review.
6. Editing: Making final changes that are universally applied to a document to make it more consistent (such as copy edits, grammar, and logic).

It is important to note that generally these six activities do not occur in a linear sequence. In the process of writing a document, reviewing activities may be done not only by the authors of the document, but also by instructors or editors or peers who read and annotate the document for with regard to improvements in content, grammar, and style. To support authors during collaborative writing, it is important

to focus on those activities that are performed by the writers, and not by reviewers. Toward that end, it is crucial to concentrate on automatically identifying the following five collaborative writing activities that are performed by writers -- brainstorming, outlining, drafting, revising, and editing -- and leave the reviewing aside.

Faigley and Witte (1981) categorized text revisions into surface changes and meaning changes. Surface changes, as opposed to meaning changes, do not alter the meaning of the text. Surface changes are further subcategorized into formal changes, such as spelling, grammar, and punctuation; and meaning-preserving changes, which "paraphrase the concepts in the text, but do not alter them"(Faigley & Witte, 1981). Meaning, or text-based changes, are subcategorized into macrostructure or microstructure changes. Macrostructure changes include text revisions that would alter the summary of a text, while microstructure changes would not. The results of Faigley and Witte's study showed that experienced writers made a lot of meaning changes, while novice writers made mainly surface changes (Faigley & Witte, 1981). Boiarsky (1984) argued that Although Faigley and Witte's taxonomy provides "a means for describing the changes in the text based categories, which are for analysing how writers make text-based changes, they do not provide a description of why writers make such changes." She further developed a model for analysing semantic changes in the writing process (Boiarsky, 1984) in which she identified the following 11 types of revision functions and operations: Alteration of form; reorganization of information; improvement in coherence; deletion of information; expansion of information; emphasis of information; subordination of information; creation of immediacy; improvement of prosody; improvement in vocabulary; and correction of grammar and mechanics. Authors used these text change operations in their writing activities for different purposes in order to produce final pieces of writing.

Boiarsky's study examined three phases of writing -- during drafting, between drafting, and rehearsal – and involved interviews with writers and examinations of their drafts, including revisions made during the rehearsal phase. She concluded that these 11 functions and operations could provide a comprehensive and discriminating means of describing writers' processes of revision, as well as a valid set of criteria for analysing revisions. Boiarsky also observed that writers did not engage in every

form of revision function and operation in every work, which she suggests might be a reflection of the writer's previous knowledge.

The nature of the changes made by authors can be analysed by using two techniques: text mining and process mining. Work that relates to these two methods of analysis is described in the following section.

2.2 Text mining for Detecting Cohesion and Topics

During the process of writing, authors go through several revisions of text edits before the final work is completed. Particularly, authors introduce ideas (topics) by creating and editing sentences and paragraphs in the written text. In order to develop and convey topics to readers, authors refer to the same topics across several sentences and paragraphs and make connection between these topics in the content text. These overlapping and connection of the topics make the text cohesive. Therefore, detecting changes in cohesion and topics is important to understand how authors develop the content text during their writing processes. Text mining techniques can be used to extract semantic meaning from these editing processes, particularly from textual features such as cohesion improvement and topic (concept) changes. This subsection presents significant works related to the use of text mining techniques for analysing these features of the written texts.

McNamara and her colleagues at the University of Memphis used Coh-Metrix (Graesser et al., 2004; McNamara et al., 2010) to analyse writing quality (Crossley & Mcnamara, 2007; McNamara et al., 2009; Ozuru et al., 2010; Weston et al., 2011). Coh-Metrix is a computational tool that is used to assess text on more than 600 linguistic and lexical indices. These indices are related to conceptual knowledge, cohesion, lexical difficulty, syntactic complexity, and simple incidence scores (Weston et al., 2011). Coh-Metrix provides the following interesting indices:

- *Syntactic complexity* which computes the mean number of words before the main verb and the mean number of high level constituents (sentences and embedded sentence constituents) per word and per noun phrase.
- *Connectives and logical Operators* which measure the density of connectives. These connectives are associated with positive additive (*also, moreover*), negative additive (*however, but*), positive temporal (*after, before*), negative temporal (*until*), and causal (*because, so*) measures. The logical operators

measured in Coh-Metrix include variants of *or*, *and*, *not* and *if-then* combinations.

- *Causality* which measures causal cohesion by calculating the ratio of causal verbs to causal particles. The causal verb count is based on the number of main causal verbs identified through WordNet (Fellbaum, 2005).
- *Lexical overlap* which considers four forms of lexical overlap between sentences: noun overlap, argument overlap, stem overlap, and content word overlap.
- *Cohesion* which measures semantic coreferentiality using Latent Semantic Analysis (LSA) (Landauer & Dumais, 1997; Landauer et al., 2007), a mathematical technique for representing deeper world knowledge based on large corpora of texts. Weston et al. (2011) explained that "Unlike lexical overlap, LSA measures associations between words based on semantic similarity, which can be used to assess the amount of semantic coreferentiality in a text".
- *Spatiality* which measures spatial cohesion using motion verbs and location nouns.
- *Word characteristics* which reports on a variety of lexical indices such as hypernymy and polysemy.
- *Word frequency* which indicates how often particular words occur in the English language.

In light of the above, Coh-Metrix seems to fulfil the need of discourse psychologists and other researchers to have access to one computational linguistic tool that analyses various linguistic features of texts. In particular, Coh-Metrix has been used to detect a wide variety of differences in text and discourse (Crossley & Mcnamara, 2007; McNamara et al., 2009; Ozuru et al., 2010; Weston et al., 2011) based on an investigation of the final writing product. Although the indices of Coh-Metrix described above can be used in analysing the writing process, not all of those indices are suitable for use at the writing stages during which some words and/or sentences have not been completely written yet. Nevertheless, this thesis uses a similar technique to compute the semantic similarities between sentences and paragraphs in order to measure the cohesion, which is the overlapping of words and topics, as explained in the following section.

2.2.1 Cohesion Measure

Cohesion refers to the presence or absence of explicit cues in the text that allow the reader to make connections between ideas in the content. For example, overlapping words and concepts between sentences indicate that the same ideas are being referred to across sentences. Likewise, connectives such as "because", "therefore", and "consequently" inform the reader that there are relationships between the ideas expressed and the nature of those relationships. Both cohesion and coherence are two similar terms used in natural language processing, particularly discourse analysis. Cohesion differs from coherence in that cohesion refers to the explicit cues found in the text, whereas coherence refers to the understanding that the reader derives from these cues in the text, which may be more or less coherent depending on a number of factors, such as the reader's prior knowledge and skill (McNamara et al., 1996; O'Reilly & McNamara, 2007). For the purposes for this research, only cohesion is taken into account in analysing the writing process, leaving coherence aside.

McNamara and her colleagues, interested in the roles played by cohesion with regard to writing quality (Crossley & Mcnamara, 2007; McNamara et al., 2010; Ozuru et al., 2010), applied the techniques as elaborated in the beginning of this subsection to examine that factor; but again, their work was focused on the final writing product. In order to acquire insight on how authors carry out their activities during the course of the writing process, one must analyse changes in cohesion over time from one revision to another, beginning by first measuring cohesion in the text; to achieve this, the following text mining techniques proposed by researchers for computing cohesion can be considered:

Latent Semantic Analysis (LSA) (Landauer & Dumais, 1997; Landauer et al., 2007) has been widely used to measure deeper quality patterns in texts, especially discourse cohesion. Traditionally, LSA semantic spaces were normally created from large corpora that reflect an assumed background knowledge. However, Villalon and Calvo (2009) proposed an elegant technique for creating a semantic space using a single document and no background knowledge. Their technique measured the semantic distance between consecutive sentences and paragraphs of a document in order to identify possible breaks in cohesion in the text. For a particular document comprised of several revisions produced during the writing process, the technique of

building LSA semantic space can be used for each revision of the document in order to explore the improvement of cohesion during the writing of the text.

Standard text similarity measures (based on term frequency) perform poorly on computing the distance between consecutive sentences and paragraphs for a single document because of the lack of common words between the consecutive sentences and paragraphs. Yin and Meek (2007) achieved an improved Web-relevance similarity measure for calculating similarity between short segments of text with an approach that extended the terms of text segments using information from the Web (search engine), and computed the similarity scores based on the extended representation of those text segments. This technique can be integrated into the technique proposed by Villalon and Calvo (2009) to improve the computation of cohesion measures in the text.

Although it is common for writers to repeat words to emphasize concepts in the text, good writers usually use synonymy and pronouns to avoid annoying repetition; and this issue was not taken into account in the technique proposed by Yin and Meek (2007). Varelas et al. (2005) introduced a semantic similarity measure using Wordnet as the underlying reference ontology. This method of measuring similarity as well as those previously mentioned (Villalon & Calvo, 2009; Yih & Meek, 2007) can be used to measure the distance between consecutive sentences and paragraphs in single document semantic space in order to investigate the progressive improvement of cohesion during writing processes.

The works discussed above were all focused on computing cohesion in the text of the final written product; however, not much research exists with regard to analysing cohesion changes made during the entire course of the writing process, except for one interesting work by Thomas and Sheth (2007) on automatically identifying *semantic convergence* in Wikipedia articles. The researchers define semantic convergence as a notion of article stability (Thomas & Sheth, 2007). For a particular Wikipedia article comprised from several revisions, they created a vector space representation of the article's *revision milestones* using TF-IDF as a term-weighting scheme (Salton & McGill, 1983). The vector space is computed using all the words occurring in all revisions of the article. A revision milestone is a combination of all revisions made in one week, with the word count for milestones calculated as medians. Thoms and Sheth then computed two kinds of semantic distances, one

being the cosine distance between every pair of consecutive revision milestones and the second being the cosine distance between every revision milestone and the final revision. Based on these two computed semantic distances, the authors examined how particular articles became mature or semantically stable despite the ongoing text edits that were performed on them.

Using the technique developed by Thomas and Sheth, one could infer that text edits that produced stable cohesion through revisions were intended to revise and/or edit content without significantly changing the flow of ideas and the concept of the text. Although the vector space model used in their work is similar to the previously mentioned works using LSA such as Villalon & Calvo (2009), Thoms and Sheth (2007) did not use singular value decomposition, which is the mathematical technique underlying LSA, to transform the vector space. Overall, their work shed light on possible methods of analysing cohesion during the writing process.

Another important feature of the writing process is topic evolution, which represents how ideas and concepts have been developed during writing. Relevant works in this area are the subject of the following subsection.

2.2.2 Topic Extraction and Topic Evolution

Blei and Lafferty (2009) define topics as the "collections of words that co-occur frequently in a text collection, and can be used to provide structure to an otherwise unstructured collection of text". Topics can be discovered through the application of two different techniques: (1) using probabilistic graphical models such as latent Dirichlet allocation (LDA) or topic modelling (Blei et al., 2003), and (2) using a document clustering algorithm such as Lingo (Osinski & Weiss, 2005). Each of these two techniques has its own strengths and drawbacks when used for extracting topics and discovering topic evolution.

Latent Dirichlet Allocation (LDA) is a popular probabilistic topic modelling technique which, at the time of this research, has never been used to extract the evolution of topics during the writing of a document. The closest method used for this purpose is DiffLDA (Thomas et al., 2011), which has been applied for extracting topic evolution in software repositories.

There has been an increase in research related to analysing the evolution of topics in software development. Analysing topic changes over time in regard to software

source codes has several aspects in common with analysing topic changes in regard to the writing process. Similar to jointly authored documents, software source codes, are usually updated incrementally from one revision to another revision as programmers developed the software. Although sometimes there can be lots of changes occurring in one revision, there still exists some overlap of text contents between the revision and the previous one.

Topic models such as LDA (Blei et al., 2003) -- statistical models used to automatically extract the topics from a given corpus -- have proven to be an effective tool for analysing, understanding, and describing software project artefacts (Hall et al., 2008; Linstead et al., 2008; Thomas, 2011). The Hall model was originally developed (Hall et al., 2008) to analyse topic evolution using conference proceedings as the corpus. Linstead et al. (2008) and Thomas et al. (2010b) used the Hall model on software repositories by simply applying LDA to all versions of all documents at once and performing post hoc calculations based on the observed probability of each document in order to map topics to software versions. The main advantage of this approach is that no constraints are placed on the evolution of topics, which results in flexibility for describing the large, seemingly random changes to a corpus that are typical in software development.

However, topic models based on LDA assume that there are no duplicated documents in the corpus (Thomas, 2011; Thomas et al., 2010a). In other words, LDA treats each document as unique. This assumption holds for all kinds of texts used in the topic modelling literature such as journals, blog posts, and newspaper articles. During writing processes, however, documents are usually updated incrementally, and there is a tremendous amount of text content overlap between revisions; consequently, LDA can only be applied to the non-overlapping portions of two consecutive revisions.

In 2010, Thomas et al proposed a new model, called *DiffLDA*, which addressed LDA's sensitivity to document duplication by operating on the difference (i.e. non-overlapping portions) between versions of a source code document, resulting in a more accurate, finer-grained representation of topic evolution (Thomas et al., 2010a). Specifically, DiffLDA relied on a pre-processing step that used only the changes between consecutive versions, instead of all versions, of a document. Each version of source code document was considered to be a plain text consists of codes as words

(terms). For each source code document, DiffLDA first computed the edits between consecutive versions using the standard UNIX *diff* utility, resulting edited texts (difference between successive versions). Each text edit text was then classified as either an *add*, *change*, or *delete*, depending on whether the edit resulted in more, the same, or fewer lines of code, respectively; and an existing line that was changed was considered to be deleted and then added again (Thomas et al., 2011) so that only two types of text edits remained -- addition and deletion. For each version of the document, DiffLDA created two *delta* documents to store these two types of edits, after which LDA was applied to the entire set of *delta* documents at once, which revealed a set of topics and membership values for each *delta*. Finally, the output of LDA was examined to compute the metrics of interest such as the *topic assignment metric*, which shows the distribution of topics in each version, and the *hotness metric*, which represents how much edit activity a topic has received in each version (Thomas et al., 2010a).

The *DiffLDA* model manifested a limitation in that when the two versions of a source code document were compared, and the edits were saved into the *delta* documents as explained above, although duplication was eliminated and the document edits were captured, the context of the original documents was destroyed. In other words, the words in the delta documents were no longer contextualized by surrounding words and paragraphs as they were in the original document (Thomas et al., 2010a). As a result, it was difficult to infer and interpret the topics (labels) in order to understand the actual nature of the topic changes.

Other factors that must be taken into account when using topic models and LDA-based models are the parameter setting and the exact inference problem. When applying LDA, one needs to specify the number of topics as an input to the model, along with document and topic smoothing parameters α and β, of the two Dirichlet distributions: document topic distribution and topic-specific word distribution. At present, there is no standard method for selecting the values for these input parameters beforehand. One approach is to use the well-established values that have been shown to work reasonably well (Griffiths & Steyvers, 2004; Wallach, 2008). Another approach is to first learn the number of topics using algorithms such as the Hierarchical Dirichlet Process (Teh et al., 2006). In addition, LDA is a generative probabilistic model in which exact inference is intractable, and Gibbs sampling is

often used to sample the posterior probabilities of documents and topics. Consequently, different sets of sampling iterations will produce slightly different results (Thomas et al., 2011).

The link model, proposed by Mei and Zhai (2005) and first used on software repositories by Hindle et al. (2009), took a different approach than the Hall model by applying LDA to each version of the repository separately, followed by a post-processing phase to link topics across versions. Once the topics were linked, the topic evolutions could be computed in the same way as in the Hall model. This technique involved the use of similarity thresholds to determine whether two topics were similar enough to be called the same, since LDA is a probabilistic process and it is not guaranteed to find identical topics in different versions of a corpus. As a result, at each successive version, some topics are successfully linked while others are not, causing past topics to "die" and new topics to be "born".

Because of these limitations, this thesis also considers a deterministic approach using a document clustering algorithm to extract topics. One such technique for extracting topics was based on the document clustering algorithm, Lingo (Osinski & Weiss, 2005). The Lingo algorithm was created originally for the purpose of clustering web search results or snippets (Osinski & Weiss, 2004). Unlike other document clustering algorithms, which determine description (labels) after discovering the actual cluster content, Lingo emphasised indentifying clustering description first, before allocating cluster content. In addition, unlike the previously discussed topic modelling algorithms in which each document was considered to be a probability distribution over some topics and each topic was presented as a probability distribution over a number of words, Lingo builds a TF-IDF vector space model from the input snippets and uses LSA to discover cluster labels or topics. Finally, the algorithm identified cluster members by matching the input snippets against a series of queries, each of which is a single cluster label. Particularly, Osinski and Weiss (2005) also used the *cosine* distance to calculate the similarity between the input snippets and the cluster labels.

Lingo created overlapping clusters and the "other topics" cluster for the input snippets that did not match any of the cluster labels, and provided several advantages over its counterpart, probabilistic topic modelling algorithms. First, Lingo could automatically extract labels or topics with minimum human expert interaction and

was able to handle short text forms such as sentences. Second, unlike the topic modelling algorithms, which required training several models and validating the models in order to obtain the number of topics, Lingo could automatically discover the number of topics during its execution time by using the singular value decomposition technique in LSA. Consequently, the Lingo algorithm will perform better in terms of running time if one wants to extract and compare topics between several revisions for the purposes of analysing writing processes. Therefore, in this thesis Lingo algorithm was selected to extract topics and compute topic overlapping for semantic analysis in order to identify writing activities and discover writing process models.

Table 2-1. Summary of topic extraction algorithms.

Features	Topic Extraction			
	Probabilistic graphical models (based on LDA)			Document clustering (based on LSA)
	Hall model (Hall et al., 2008)	Link model (Mei & Zhai, 2005))	DiffLDA (Thomas et al., 2011)	Lingo (Osinski & Weiss, 2005)
Sensitivity to document duplication	Y	Y	N	N
Automatic discovery of the number of topics	N	N	N	Y
Automatic discovery of topic labels	N	N	N	Y

The works discussed so far are involved with extracting features of text content, such as cohesion and topics, in order to identify the semantics of text edits based on these features. Particularly, Latent Semantic Analysis (LSA) is used to compute cohesion measure. In addition, topics can be discovered through the application of two different algorithms: (1) probabilistic graphical models such as latent Dirichlet allocation, and (2) a document clustering algorithm such as Lingo. Table 2-1 summarises topic extraction algorithms by comparing their features and highlighting similarity and differences. The semantics of text edits can be defined as

the actions or activities that authors intend to perform during their writing tasks. In order to gain a deeper understanding of the way that authors develop their documents, it is possible to examine the sequences of the identified actions or activities and process mining techniques, as described in the following discussion.

2.3 Process Mining

The aim of process mining techniques is to discover the underlying patterns of various processes by extracting knowledge from process observation data, such as recorded event logs in organisational management systems, student interactions with each other or software captured in learning software logs (Trčka et al., 2010; van der Aalst, 2011). Process mining techniques have three broad uses: the first is the *discovery* of process models without using any *a priori* information; the second is the *checking conformance* of the observed behaviour to an *a priori* process model or behaviour workflow model; and the third is the *extension of a priori process models* by projecting discovered patterns back on the initial models and adjusting processes accordingly (Rozinat et al., 2007; van der Aalst, 2011). This thesis focuses on extracting process models of the entire writing procedure and on using these models to perform an analysis for the purpose of discovering the patterns of text edits that are carried out during writing processes. Toward this end, a review of the literature relevant to discovering and analysing process models is presented below.

The open source process mining framework ProM (ProM, 2013) has been widely used in extracting business process models. ProM provides several algorithms for discovering process models, such as the α-algorithm (van der Aalst et al., 2004), Heuristic Miner (Weijters & Ribeiro, 2010; Weijters & van der Aalst, 2003), and genetic process mining (Medeiros et al., 2007) as well as plug-in utilities such as Dotted Chart Analysis utility (Song & van der Aalst, 2007) and Performance Sequence Analysis (Bozkaya et al., 2009). Of these process mining algorithms, the Heuristic Miner algorithm (Weijters & Ribeiro, 2010; Weijters et al., 2006) has been successfully employed in several applications for a number of reasons. Particularly, the algorithm can be used to exploratory mine less structured process data when *a priori* workflow model is not known. In addition, this mining algorithm can handle event logs with various kinds of "noise", such as diversions from common sequences or incomplete traces of process information. Such noise is common in event logs,

particularly when event logs are derived from online Web 2.0 applications supporting collaborative learning tasks. Overall, Heuristic Miner is considered to be appropriate for mining processes that require flexibility and cannot strictly be predefined in advance, such as writing process models.

Other types of works have focused on the automatic analysis of student learning processes and patterns (Romero & Ventura, 2006; Xiaoli et al., 2010). Most of this research primarily analysed student log files that were automatically generated from the students' interactions with software and/or each other using various statistical, data and text mining techniques, such as hidden Markov models (HMM), process mining, time series and sequential pattern mining (Jeong et al., 2010; Pechenizkiy, Trcka, Vasilyeva, Aalst, & Bra, 2009; Southavilay et al., 2010b). For example, Kay et al.(2006) analysed student interaction sequences when they worked collaboratively on software development, for the purpose of detecting learning patterns that are indicative of team problems and success. Jeong et al. (2010) analysed student self-learning behaviours in asynchronous learning environments for adult learners using HMM; the analysis revealed that successful students had more linear learning behaviours that remain consistent across different models than unsuccessful students. Many other e-learning studies used different data mining algorithms for exploring student learning in e-learning systems, such as analysing student navigational behaviour in virtual campus environments and identifying gifted students' learning paths (Romero & Ventura, 2006). The event logs of these studies, however, considered student interaction sequences as sequences of activities, but did not explore the semantics of activities (the nature of the interactions) that occurred in the studies.

Recently researchers have used both semantic (content) analysis (i.e. text mining) and workflow analysis (i.e. process mining) in their works to extract activities (events) embodied in emails and other artefacts. For instance, Kushmerick and Lau (2006) tried to discover process models from an email dataset of e-commerce transactions. They first used identifiers, such as transaction numbers, to determine the activities of each transaction. They then employed a hierarchical agglomerative clustering method to establish the transitions of the activities and derive the process model. In another study, Buffet and Geng (2010) proceeded to refine an initial process model -- a *priori* information -- based on additional evidence (i.e. email

correspondence). They accomplished this by using text classification for labelling events of email correspondence, and iteratively refined the existing process model based on the labelled events.

Despite all the prior research described above, to the best of my knowledge, this thesis constitutes the first example of using both semantic and process analyses to identify (or label) writing activities based on text edits, cohesion and topic overlap, and to extract writing process models for process analysis.

2.3.1 Analysing Writing Processes

The last thirty years have manifested an increasing interest in discovering writing strategies and exploring the various stages of the writing process. During the 1980s, researchers studied how authors went about writing and revising the many drafts of their work, and proposed methods for analysing these procedures. Initially these analytical methods were performed by hand, by tediously collecting hardcopies of revisions and manually analysing them, as in the studies conducted by Faigley & Witte (1981) and Boiarsky (1984). More recently, new software and advances in computational linguistics have allowed researchers to collect revisions in electronic formats, as well as securing logs or revision histories that include timestamp and authorship information, thereby assisting by partially automating the collection of revisions produced by authors and allowing researchers to concentrate on the analysis of the writing process. Currently, there are many software applications used in the study of writing processes, ranging from key-stroking, single-user logging applications such as InputLog (Leijten & Van Waes, 2006) to version controlled document applications which support collaborative writing, like Google Docs (Google Docs, 2013), EtherPad (EtherPad, 2013), and Wikipedia (Wikipedia, 2013).

Several researchers (Caporossi & Leblay, 2011), (Leijten & Van Waes, 2006), and (Tillema et al., 2011) have used InputLog to study and analyse the writing processes of individual authors. Of particular interest is the analysis performed by Tillema et al. (2011) from a study conducted to investigate whether the *(meta)cognitive* activities (i.e. reading the assignment, planning, text production, revising, etc) of secondary school students during writing tasks, as measured by thinking aloud techniques and key-stroke logging, could be predicted by their individual writing styles -- planning or revising. The researchers assumed that

writing style was determined by the temporal distribution of (meta)cognition across the writing process. A multilevel regression model was employed to model the occurrence of the (meta)cognitive activities over the period of the writing process. The results showed that among all activities, the online temporal distribution of reading the assignment and planning were different for different degrees of the students' writing styles. Although this study investigated single authors, the analysis technique can nevertheless be applied for analysing collaborative writing by computing the temporal distribution of (meta)cognitive activities across individual students' writing processes.

Although recently there has been a great deal of research using Wikipedia's revision history for applications in Natural Language Processing (Ferschke et al., 2013), these studies used the revision data and its history record as the basis for practical applications such as spelling correction, vandalism detection, automatic article quality assessment, or trustworthiness. Research on extracting and analysing collaborative writing processes automatically is still scarce.

Among the various individuals who used Wikipedia's revision history to analyse the evolution of Wikipedia articles, there was one particular researcher -- Han et al. (2011) – who applied a Markov model technique to analyse the lifecycle of the Wikipedia articles. In this study, the authors defined six stages through which an article usually passed before reaching a convergence state. These states were identified as 1) building structure, 2) contributing text, 3) discussing text, 4) contributing structure and text, 5) discussing structure, and 6) text/content agreement. Three features were used as observation variables to determine these states: 1) Update type, including insertion, deletion, and modification; 2) content type, including structure, text, format, structure + text, text + format, and structure+format; and 3) revision granularity, including heading level, word level, sentence level, paragraph level, section level, and link level. A sequence of these observation variables was used to build a Markov model of a particular article, and revision cycle patterns were extracted based on this model in order to find correlations between human evaluated quality classes and revision cycle patterns to automatically assess the quality of an article. It should be noted that although the authors made reference to having used hidden Markov models, the hidden states applied were nevertheless the six states mentioned above. A learning algorithm was not used to obtain the

hidden states; in fact, the Markov states are predefined based on the values of the three features used as observation variables. For example, inserting a heading was determined to be part of the "building structure" state, whereas inserting words in a paragraph was deemed part of the "contributing text" state.

Nonetheless, the Han et al. approach outperformed the previous results in the study by Dalip et al. (2009) that worked on the same task and data, but without using features based on revision history data. Hence the features based on revision history proved to be helpful elements for not only predicting quality of Wikipedia articles but also analysing the history of jointly authored documents such as the Wikipedia articles.

Another study also using Wikipedia revision history and types of text edits was performed by Zeng et al. (2006), who were the first researchers to develop and evaluate a model of article trustworthiness based on revisions histories. Their model was based on author reputation, edit type features and the trustworthiness of the previous revision. The edit type features chosen for use were addition and deletion; the number of deleted and/or inserted words was measured. Interestingly, the authors applied a Dynamic Bayesian network (DBN) based on these features to estimate the trustworthiness of a revision based on a sequence of previous states, i.e. revisions. Although this work was not related to the writing process, the proposed techniques, especially DBN, could also be employed in analysing writing processes, using revision history and different types of features.

Although the works discussed above all involved collaborative writing by a web community like Wikipedia, small scale Wikis in classrooms, which also provide revision histories, can also be used to identify students' collaborative writing patterns. Heeter and Jeong (2012) conducted a study to extract students' collaborative processes in Wikis for the purpose of discovering whether group members preferred to work individually (sequentially or in parallel) rather than collaboratively (or reciprocally) in wikis. Interestingly, the authors systematically generated a coding scheme and then manually coded text edits captured in the revision histories of a Wiki. Based on sequences of coded text edits, they built Markov models and identified patterns in the action-sequences that students performed in a Wiki. The authors' result was consistent with prior research, which found that students preferred to work on an individual rather than a collaborative

basis (Heeter & Jeong, 2012). Although this study analysed sequentially the individual and collaborative writing actions observed in the Wiki, the proposed coding scheme was based on raw student text edit data captured in the Wiki's revision history and did not explore the semantic nature of those text edits.

This concludes the overview of representative modelling techniques for analysing both individual and collaborative writing processes, such as multilevel regression models, hidden Markov models, and Dynamic Bayesian networks. The following section presents existing techniques that involve the use of graphs and visualisation methods for analysing writing processes.

2.3.2 Visualising Writing Processes

The process mining tool ProM (ProM, 2013) includes several means of visualisation, such as Dotted Chart Analysis (Song & van der Aalst, 2007) and causal dependency diagrams (van der Aalst, 2011), but these charts and nets were tailored for business workflow models. To model writing processes, some researchers use Markov models representing transitional state diagrams. Particularly, Heeter and Jeong (2012) used Markov models as forms of visualisation to study writing processes. Figure 2-2 depicts a transitional state diagram of individual author's actions on own Wiki pages.

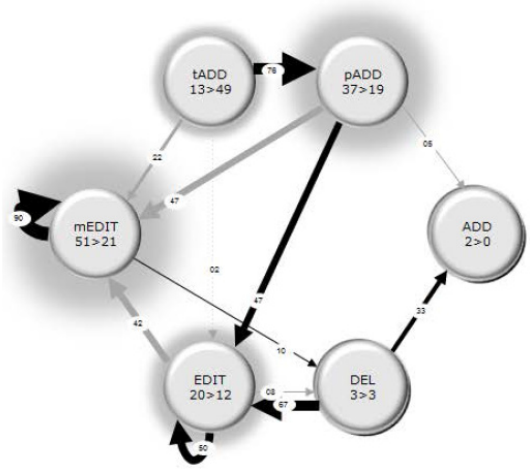


Figure 2-1. Transitional state diagram of a writing process of an individual Wiki author (from Heeter and Jeong (2012) – Permission has been authorised).

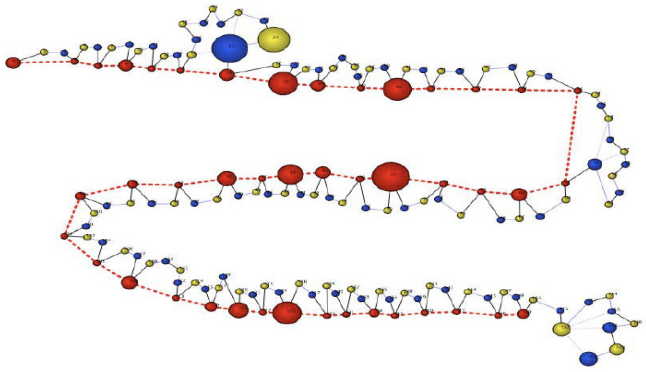


Figure 2-2. Graph visualisation of writing process: an example of novice writer (from (Caporossi & Leblay, 2011) - permission has been authorised).

Another visualisation tool specifically for the writing process was proposed by Caporossi and Leblay (2011), based on the graph theory that captures the viewpoint to understand the writing process. Their technique used nodes and links to create a graph of a writing process, with the size and colour of each node indicating the number of elementary events (i.e. text edits) that it represents and their nature, respectively. For instance, yellow nodes represent additions that have later been removed; red nodes depict additions that remain until the final text; and blue nodes indicate deletions. These nodes are connected by links or edges representing a spatial or temporal relationship, indicated by the shape and colour of the edges. Figure 2-2 shows an example of a graph visualisation of novice author's writing process, taken from (Caporossi & Leblay, 2011). By examining the graphs, Caporossi and Leblay (2011) were able to analyse the writing process and discovered that the graph of an expert writer was interconnected, whereas the graph of novice writer was quite linear.

Although this technique for representing the writing process was able to take into account changes in the position of the text (moving the words around) and allowed researchers to identify the portion of the document that was modified by the writer, it did not distinguish the various writing activities as defined by Faigley & Witte (1981) and Boiarsky (1984). It also did not indicate the time and duration corresponding to each text edit represented by the nodes. In addition, comparing to a transitional state diagram as shown in Figure 2-2, it is difficult to gain insight how individual text edits depend on each other (e.g. which text edits were likely to immediately follow another text edit).

Recent work by Perrin and Wildi (2010) investigated the dynamics of cursor movement during the process of writing and presented the movement in *progression graphs*. Based on these graphs, they proposed a statistical method to discover the stages of writing through which authors progress. Specifically, the *progression graphs* were used to represent time series consisting of large "*bursts of signals*", and statistical signal extraction was used to decompose the series into sequences of interesting features. Writing stages were then identified based on the changes of the features. The Perin and Wildi analysis, however, focused only on the temporal dynamics of cursor movements, not on the edits in the content of the text. This research differs from that work in that this thesis is concerned with investigating how the text content changes over time throughout the writing to derive an understanding of the entire process.

Kim and Lebanon (2010) proposed a novel representation of document versions based on the location of the content words. They built a matrix in which columns correspond to the position of each word and rows represent versions (time), using the space-time smoothing method to extract and visualise changes in words over time, discovering revision patterns based on these changes. Although this method can determine which parts (i.e. word content) of the documents change over time, it cannot discern the intended writing activities, the stages of the process.

Although all the graphs and the underlying techniques that were discussed in this subsection proved to be useful for analysing the writing processes of documents written by single authors, they could not be applied in the context of investigating collaborative writing.

2.4 Tools to Support Collaborative Writing

Aside from the above visualisation, there are other multiple aspects of collaborative writing (CW) that can be supported by technology. Lowry and Nunamaker (2003) were the first researchers to introduced an internet-based CW tool, called Collaboratus to support enhanced coordination and group awareness. Some of the key CW activities directly supported in Collaboratus that could be conducted by group members simultaneously included group brainstorming, group voting, group outlining, group writing, and group annotations that allow multiple levels of group discussions. Having different screens and features, according to the

activity a group was working on, greatly increases coordination by focusing team members on the appropriate task at hand. Lowry and Nunamaker also conducted a one-month-long experiment to compare its features and support with those of Microsoft Word. Their result showed that Collaboratus groups generally experienced better outcomes than Word groups, in terms of productivity, document quality, relationships, and communication, but not in terms of satisfaction. From the study, Lowry and Nunamaker concluded that specialized collaborative writing (CW) tools could improve group coordination, group awareness, and CW activities.

The research mentioned above was conducted in 2003. Since that time, technology, especially machine learning and natural language processing (NLP) have been advanced gradually. Recently, Calvo et al. (2011) created an environment, iWrite, to support students' collaborative writing. iWrite outsources the writing tool and the storage of students' documents to third party cloud-computing vendor (i.e. Google). It consists of two main elements: 1) a functionality to manage writing activities for both students and instructors in large cohorts, particularly the management and allocation of groups, peer reviewing, and assessment; 2) an intelligent feedback tool, Glosser (Villalón et al., 2008), which uses several machine learning and NLP techniques to provide feedback on the text surface level and concept level of a document, such as cohesion as well as automatic question generation (Calvo et al., 2011). Although iWrite provides information on team contribution, for instance which author contributed which sentences or paragraphs and how these contributed to the overall topics of the document, it does not provide feedback about team processes based on writing activities and their semantic significance.

Cognitive visualisation, such as a Concept Map can also be used as a feedback tool for supporting writing processes. Concept Maps (CM) were introduced by Joseph Novak as a way to assess children's understanding of science with graphical tools to organize and represent knowledge (Novak & Gowin, 1984). In a CM, concepts are represented in boxes that are linked by labeled relationships; two related concepts (including their link) form a proposition or semantic unit. Concepts are also arranged hierarchically such that more general concepts are located higher on the map and specific concepts such as examples are located lower. A concept by itself

does not provide meaning, but when two concepts are connected using linking words or phrases, they form a meaningful proposition.

Villalon and Calvo (2011) proposed a concept map miner (CMM), a tool that automatically generated concept maps from students' writing assignments. Particularly, they presented a novel approach in the educational application of CMs. Here, CMs were embedded in writing (as opposed to reading) activities and were used to summarize the students' own writing. Unlike in the more typical scenarios of using CMs to support reading, in Villalon and Calvo's work the CMs become approximate representations of students' current state of knowledge. From the students' perspective, such CMs can be used to reflect on their own knowledge and also to help students see their writing from a different perspective. From the instructor's point of view, such CMs can be used as a rapid assessment of students' conceptual understanding.

Another work on using concept maps, as is the work by Macedo et al., who presented a method for analysing concepts in the text which demonstrated a collaborative writing system that enabled students to communicate with each other and elaborate a text in a collaborative way (Macedo et al., 2009). Importantly, the system had a text mining tool that enable teachers to extract concepts from students' writings and generate a graphical representation of those concepts. Unlike the work mentioned previously, the method for extracting topics (i.e. concepts) in Macedo's work was simply based on the frequency of the appearance of compound terms in the text.

These techniques based on concept maps can be applied to investigate the way in which concepts and their semantic relationships change from one revision to another during the course of the writing process, by comparing concepts and their semantic relationships of concept maps of two consecutive revisions. However, the efficient method for comparing two concept maps and computing their overlapping and discrepancy is still an open research. In addition, the concept maps extracted from revisions at the writing stages during which some words and/or sentences have not been completely written yet, may not be meaningful.

This section provides review on tools to support collaborative writing. The first tool, Collaboratus was based on user interface to facilitate group coordination and awareness. The second tool, iWrite integrated data mining and NLP to support

students' collaborative writing. The last two tools used concept maps to provide feedback on writing processes. However, at the time of this research, there have been no collaborative writing tools that provide ways to mine and access log data as described in this thesis.

2.5 Summary

Since the first original model was proposed in 1980 by Hayes and Flower (1980), much research has been conducted relating to the theoretical frameworks of cognitive models of writing processes. These cognitive models can be categorized into three main types -- knowledge-transferring, knowledge-transforming, and knowledge-constituting models. In general, these theoretical frameworks focus on the writing processes of particular documents written by only one author.

A new type of theoretical framework is needed to examine the collaborative writing processes of groups of authors, because the collaborative writing context presents different, additional, and greater organizational and cognitive demands. In 2003, Lowry et al.(2003) proposed a taxonomy of writing activities for the collaborative environment. Two different proposed models for analysing revisions -- (Faigley & Witte, 1981) and (Boiarsky, 1984) – although based on revisions of single-author documents, can be adapted for analysing documents written by groups of authors. **Based on the proposed taxonomy of writing activities, and models of analysing revisions, this thesis creates techniques to automatically extract collaborative writing activities and discover process models of the writing processes.**

Several text mining techniques have been proposed to automatically discover document topics and compute document cohesion scores. These techniques can be categorized into two different branches: Probabilistic graphical models such as topic modelling or LDA; and document clustering algorithms such as Lingo, which in turn uses LSA. Nevertheless, these techniques are still based on the final writing product; research with regard to changes in topics and cohesion during collaborative writing remains lacking. **In this thesis, text mining techniques for extracting topics and measuring cohesion are used to discover the purpose of text edits during collaborative writing in order to identify the corresponding collaborative writing activities.**

Based on these writing activities, process mining techniques can be employed in order to discover and analyse writing process models. Process mining developed for business workflows involves three main uses: Discovering process models from event logs files; checking conformance of the observed behaviour in the event logs files to process models; and extending or adjusting process models to best fit the observed behaviour. Different process models provide different views of workflow processes and are suitable for different purpose of process analysis. These process models of business workflows use process instances consisting of events (activities or transactions) predefined in business information systems. Unlike business workflows, the writing process consists of non-predefined events of text edits. In addition, these text edits are interleaving during the period of collaborative writing. **In this thesis, several semantic levels of text edits are considered in order to identify collaborative writing activities or events of writing processes and discover writing process models representing different views of semantic levels of text edits.**

Much research exists with regard to visualising document versions and associated text edits. One type of writing process visualisation proposed by Caporossi and Leblay (2011) was based on graphs that consist of nodes as text editions and links as spatial or temporal relations of these editions. Other works, such as (Perrin & Wildi, 2010) used graph techniques to analyse writing processes. Nevertheless, these techniques for visualisation and analysis did not focus on the content of the text (the meaning of the words and phrases used in the document). Although all these types of visualisations and their underlying techniques were found to be useful for analysing the writing processes of documents written by only one author, they could not be applied for use in collaborative writing contexts. **This thesis proposes a range of visualisations for representing different views of collaborative writing processes.**

CHAPTER 3

THEORETICAL BACKGROUND

The sequence of writing activities are comprised of time events and the semantics of the changes performed during those activities. Two techniques can be combined to examine the sequence patterns: process mining, which focuses on extracting process-related knowledge from event logs recorded by an information system; and text mining, which focuses on extracting knowledge about what the student wrote or edited.

This chapter provides theoretical backgrounds of both text mining and process mining techniques used in this thesis. It is important to note that this research focuses on collaborative writing (CW) with multiple writing sessions. Therefore, it does not report any theories that distinguish action sequences of CW from those of individual one. In addition, there is no particular theory that relates sequences of collaborative writing activities to quality outcomes.

This chapter organises as the following. Section 3.1 describes text mining techniques and algorithms including Latent Semantic Analysis, document clustering algorithm, Lingo, and Latent Dirichlet Allocation. Process mining, including event logs, process mining algorithm, Heuristic Miner, other process perspectives is discussed in Section 2.3. Finally, Markov Model and Hidden Markov Model, including techniques to construct those models are reviewed in Section 3.3.

3.1 Text Mining

In this section, theoretical background of text mining techniques and algorithms used in this thesis is reviewed. First, a vector space of documents' presentation and Latent Semantic Analysis (LSA) is introduced in the following subsection. How the semantic measure, i.e. sentences and paragraph similarities are computed will be also discussed in this subsection. Then, two different types of algorithms for extracting topics are described: one is based on LSA and the other one uses probabilistic techniques.

3.1.1 Latent Semantic Analysis

A common problem encountered in information retrieval, document analysis and visualization applications is that people use words for their collective meaning and not just for the literal term. Linguistically the difficulties introduced are explained by the synonymy and polysemy problems. The former refers to the many ways of expressing the same concept, where people adapt their vocabulary based on the topic being discussed, or on the particular background knowledge (both of the writer and/or the reader). The latter refers to the many meanings that a word can have, meanings that humans are able to disambiguate using information about the topic being discussed or other contextual information. Synonymy and polysemy are known to affect the accuracy of computer systems that use terms (instead of concepts) as the main way of representing information. In information retrieval in particular synonymy affects recall and polysemy affects precision.

Latent Semantic Analysis (LSA) is a statistical dimensionality reduction technique proposed by Deerwester et al. (1990) to address these issues by indexing documents based on concepts (or topics) rather than terms. This requires a semantic representation for the corpus of documents and queries. LSA starts with a term-

document matrix and uses Singular Value Decomposition (SVD) to create a semantic space where the distances between terms and/or documents reflect a *semantic* proximity. When LSA is used for information retrieval tasks user queries are projected in the semantic space as pseudo documents.

Formally, LSA defines the semantic space of a term-by-document (or term-by-sentence) matrix $X \in \Re^{n \times m}$ (that can be called the knowledge base) by decomposing it using Singular Value Decomposition as:

$$X^k = U_k \Sigma_k V_k^T$$

where $U_k \in \Re^{n \times k}$, $\Sigma_k \in \Re^{k \times k}$, $V_k \in \Re^{n \times k}$ and $k < min(m,n)$.

In this representation the m columns X_i represent the weighted term-frequency vectors (of size n) of each of the documents used to create the *semantic space*. The column vectors of orthonormal matrices U_k and V_k are the left and right singular vectors respectively. Σ_k the non-negative diagonal matrix of the k biggest singular values sorted in descending order. The rows of U_k and V_k can be interpreted as the coordinates of points that represent terms and documents respectively in the k dimensional space.

If new documents need to be represented in this semantic space, they can be represented as $d \in \Re^n$ and projected on the k-dimensional space as:

$$\hat{d} = d^T U_k \Sigma_k^{-1}$$

The result $\hat{d}$ is a k-dimensional vector that can be compared with other documents in the original knowledge base corpora or with other query documents.

The criteria to decide the value of k still remains an open question for LSA and is usually set for individual experiments. More detail of LSA can be found in (Dumais, 1991; Haley et al., 2005; Landauer et al., 1998).

As discussed in previous chapter, cohesion refers to a way establishing connections or overlap within a discourse at all sort of different levels e.g. sections, paragraphs, sentences, and even phrases. In order to calculate the cohesion of the content text, the similarities or distances between consecutive sentences of the text are computed. This thesis first builds a semantic space for a term-by-sentence for each revision of a document using LSA technique as discussed above. Based on the

semantic space, it then computes the similarity or distance between consecutive sentences using a cosine measure, as defined below:

$$\cos(X, Y) = \frac{\vec{X} \,.\, \vec{Y}}{|X|.|Y|}$$

3.1.2 Lingo: Document Clustering Algorithm

In order to identify writing activities and discover writing process models, topics and topic evolution, which represents how ideas and concepts have been developed during writing, are discovered and semantically analysed. In this thesis, a document clustering algorithm, Lingo (Osinski & Weiss, 2005) was used to extract topics. This subsection describes Lingo algorithm for the purpose of extracting topics.

The Lingo algorithm was implemented in Carrot2, an open source search results clustering library under a BSD like license and found at http://project.carrot2.org. Lingo algorithm was created by Osinski and Weiss in 2004 originally for the purpose of clustering web search results or snippets. Unlike other document clustering algorithms, which determine description (labels) after discovering the actual cluster content, Lingo emphasizes first on indentifying clustering description before allocating cluster content.

Lingo algorithm operates in the following ways. First, it performs text-preprocessing: stemming, indentifying stop words, segmenting text into words sentences. Most document clustering algorithms remove stop words from the input entirely. However, Lingo only marks stop words and leave them in the input because they can help in understanding longer phrases (for instance, “Sydney University” versus “University of Sydney”). After text-preprocessing, Lingo uses the semantic hierarchical clustering algorithm (Dong, 2002) for extract phrases and single terms that can potentially be the descriptions (labels). To achieve this, it uses the vector space model (VSM) and singular value decomposition (SVD), which is the mathematical technique underlying the latent semantic analysis (LSA) as described above. It builds a term-document matrix from the input snippets using *tfidf* as term-weighting scheme (Salton & McGill, 1983). It then discovers labels by using SVD of the term-document matrix (please see (Osinski & Weiss, 2005) for full detail of the cluster-label induction). Finally, the algorithm identifies cluster members by using VSM-based document retrieval by matching the input snippets against a series

of queries, each of which is a single cluster label. Particularly, it uses the classic *cosine* distance to calculate the similarity between the input snippets and the cluster labels. If the similarity exceeds a predefined threshold (within the range of 0.15-0.30) (Osinski & Weiss, 2005), the algorithm allocates the snippets to the corresponding cluster. It is important to note that this assignment scheme naturally creates overlapping clusters and effectively handles cross-topic documents (Osinski & Weiss, 2005). In addition, the Lingo algorithm created the "other topics" cluster for those snippets that do not match any of the cluster labels.

Cluster labels (phrases and single terms) can be used to identify topics of interest. In order to extract topics, text (of each revision) is firstly segmented into sentences using Carrot2 library mentioned above. Then, the system uses the sentences as input snippets to the Lingo algorithm and produces cluster labels, which are listed as topics. After extracting topics of individual revisions, the list of topics of a revision is compared to the list of topics of the previous revision in order to compute topic overlap between two consecutive.

3.1.3 Latent Dirichlet Allocation

Another technique used for extracting topics was based on LDA (Blei & Lafferty, 2009), which is a generative model that models each document in a corpus as multi-membership mixture of T topics (Thomas et al., 2010b), and each topic as a multi-membership mixture of the words in the corpus vocabulary. A multi-membership mixture means that each document can contain more than one topic, and each word can be contained in more than one topic. Therefore, LDA is able to discover a set of ideas or themes that well describe the corpus as a whole (Blei & Lafferty, 2009).

As an example (adapted from (Thomas et al., 2010b)), there are three documents below:

d_1: "A student left his university to get a loan at the bank."

d_2: "University students prepare for their exams on the bank of a river."

d_3: "Banks make money by giving loans."

From the corpus of the three documents above, we could have a vocabulary of words ordered alphabetically: {bank, exam, loan, money, river, student, university}. Applying LDA with $T = 3$ would produce topics similar to:

z_1: {exam, student, university}

z_2: {bank, loan, money}

z_3: {bank, river}

Document d_1 would have a 50% membership in both z_1 and z_2 and 0% membership in z_3 since it contains words from z_1 and z_2 to an equal degree, and none from z_3. On the other hand, document d_2 would have a 50% membership in both z_1 and z_3, 0% membership in z_2. Document d_3 would have a 100% membership in z_2 because it only contains words from z_2. Therefore, we could represent each document as a vector of their topic memberships:

$$d_1 = [0.5,0.5,0.0]$$

$$d_2 = [0.5,0.0,0.5]$$

$$d_3 = [0.0,1.0,0.0]$$

In addition, based on the vocabulary listed above, each topic could be described as a vector of their word memberships, in which each element represents the normalized weight of each word (indexed according to the vocabulary) for that topic:

$$z1 = [0.0, 0.2, 0.0, 0.0, 0.0, 0.4, 0.4]$$

$$z2 = [0.4, 0.0, 0.4, 0.2, 0.0, 0.0, 0.0]$$

$$z3 = [0.5, 0.0, 0.0, 0.0, 0.5, 0.0, 0.0]$$

LDA defines the following generative process for each document in the collection:

1. For each document, pick a topic from its distribution over topics.
2. Sample a word from the distribution over the words associated with the chosen topic.
3. Repeat the process for all the words in the document.

More formally, in the generative process, LDA infers, for each of T topics, an N-dimensional word membership vector $z(\phi_{1:N})$ that describes which words appear in topic z, and to what extent. In addition, for each document d in the corpus, LDA infers a T-dimensional topic membership vector $d(\theta_{1:T})$ that describes the extent to which each topic appear in d. Both θ and ϕ have Dirichlet prior with hyper-parameters α and β, respectively. LDA performs these inferences using Bayesian techniques such as collapsed Gibbs sampling, a Markov-chain Monte Carlo (MCMC) method, which is currently in widespread use as an inference tool among topic modelers (Griffiths & Steyvers, 2004).

The extracted topics and topic evolutions provide an overview of how topics are created and the way that they evolve. Knowing how authors collaboratively write those topics assists in understanding how their written documents are developed. In this thesis, author-topic model (Rosen-zvi et al., 2003) is used to extract topics per auhtor in order to construct the topic-based collaboration networks, which are in turn used in investigating how authors collaboratively write their documents, which will be explained in Chapter 8. The Author-Topic model is described below.

The Author-Topic Model

The Author-Topic Model (AT Model) is an extension of LDA, which was first purposed in (Rosen-zvi et al., 2003) and further extended in (Rosen-Zvi et al., 2010). Under this model, each word w in a document is associated with two variables: au author, x and a topic, z. Similar to LDA, each author is associated with a multinomial distribution over T topics, denoted as θ. Each topic is associated with a multinomial distribution over words, denoted as ϕ. Differently to LDA, the observed variables for an individual document are the set of authors and the words in the document. The formal generation process of Author-Topic Model is as follows (Rosen-Zvi et al., 2010):

For each document, given the vector of authors, $\boldsymbol{a_d}$:

For each word in the document :

1. Conditioned on $\boldsymbol{a_d}$, choose an author x_{di}~Uniform(a_d).
2. Conditioned on x_{di}, choose a topic z_{di}.
3. Conditioned on z_{di}, choose a word w_{di}.

One important difference between the Author-Topic Model and LDA is that there is no multinomial distribution over T topics for an individual document. Therefore, if we want to model documents and authors simultaneously, further treatment is needed. A detailed description can be found in (Rosen-Zvi et al., 2010).

3.2 Process Mining

Process mining is a relative young research discipline that sits between machine learning and data mining on the one hand and process modeling and analysis on the other hand. The idea of process mining is to discover, monitor and improve real

processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's information systems (van der Aalst, 2011).

Learning management systems, intelligent tutoring systems and other learning software usually offer recorded event data, such as event logs. Note that this thesis does not assume the presence of a workflow management system. The only assumption is that it is possible to construct event logs with event data. These event logs are used to construct a process specification or representation, which adequately models the behaviour captured. The term process mining is used for the method of distilling a structured process description from a set of real executions.

Process mining techniques are used for a wide range of purposes, including: a) to discover new patterns; b) to check conformance of the observed processes to an *a priori* modelled pattern; and c) to extend *a priori* process models by using newly discovered patterns (Rozinat et al., 2007; Weijters & Ribeiro, 2010; Weijters et al., 2006). Figure 3-1 depicts the portioning of these three main types of process mining.

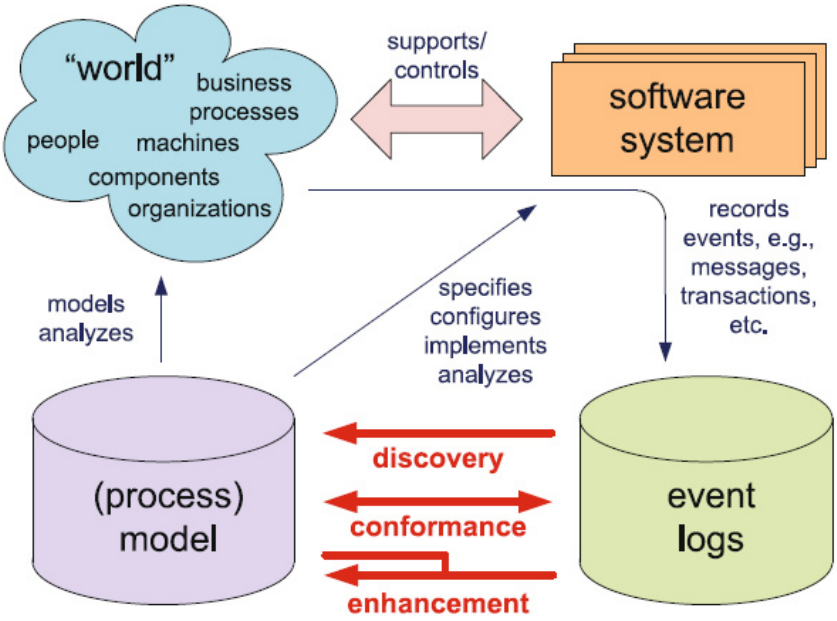


Figure 3-1. The three main types of process mining: discovery, conformance, and enhancement (from (van der Aalst, 2011) – permission has been authorised).

In the following subsections, event logs as well as several terminology used in process mining context are introduced first. After that a process discovery algorithm, Heuristic Miner, including its conformance checking technique are discussed. Finally techniques for mining additional perspectives of processes are described.

3.2.1 Event Logs

This thesis employs a process mining technique to identify and explore the structure of writing processes. We assume that it is possible to record events such that (i) each event refers to an *activity* (i.e., usually a well-defined step in the process), (ii) each event refers to precisely a *case* (i.e., a process instance), (iii) each event can have a *performer* also referred to as *originator* (the person executing or initiating the activity), and (iv) events have a *time stamp* and are totally ordered.

Event logs such as the one shown in Table 3-1 are used as the starting point for process mining. Each row of the event log shown in the table consists of an event, its process case ID and its properties such as activity, timestamp, originator, and revision ID. The recorded events are ordered by their timestamp. "Case ID" is the unique identification of process case or instance. Originator is the unique identification of user who generates the event. Note that there are more than 2 process cases in this event log; and more events of Process case 1 are listed below those of Process case 2.

Table 3-1. Example of an event log.

Case ID	Activity	Timestamp	Originator	Revision ID
1	Brainstorming	5/20/2012 7:24	S01	2
1	Brainstorming	5/26/2012 7:44	S05	101
1	Outlining	5/27/2012 0:43	S04	132
1	Outlining	5/27/2012 13:11	S05	144
1	Outlining	5/27/2012 22:31	S03	147
1	Drafting	5/28/2012 2:56	S01	196
1	Drafting	5/28/2012 3:45	S01	269
1	Drafting	5/28/2012 4:57	S01	388
1	Drafting	5/28/2012 6:30	S03	428
1	Revising	5/28/2012 6:45	S01	524
1	Revising	5/28/2012 8:39	S03	531
1	Editing	5/28/2012 11:39	S03	612
2	Brainstorming	5/29/2012 4:50	S02	2
2	Outlining	5/29/2012 7:21	S06	16
2	Drafting	5/29/2012 9:31	S02	75
….	….	….	….	….

In the context of process mining, properties of events are referred to as attributes. This thesis assumes that each event, *e* has the following standard attributes:

- Activity associated to event *e*.
- Timestamp of event *e*.
- Originator or author ID associated to event *e*.
- Transaction type associated with event *e*. There are two transaction types: start and complete.

These standard attributes are used in process discovery algorithms. Other attributes like revision ID are used in mining other process perspective.

This thesis uses an event log with the standard format of XES (eXtensible Event Stream) (Gunther, 2009), which is the successor of MXML (Mining eXtensible Markup Language). Van der Aalst (2011) fully details these two standards for storing and exchanging event logs.

3.2.2 Heuristic Mining

The purpose of this thesis is to construct a process model on the basis of an event log, as described above. Assuming that there is a set of *activity labels, A*, the goal of a process model is to decide *which activities* need to be executed and in *what order*. Activities can be executed sequentially, activities can be optional or concurrent, and the repeated execution of the same activity may be possible.

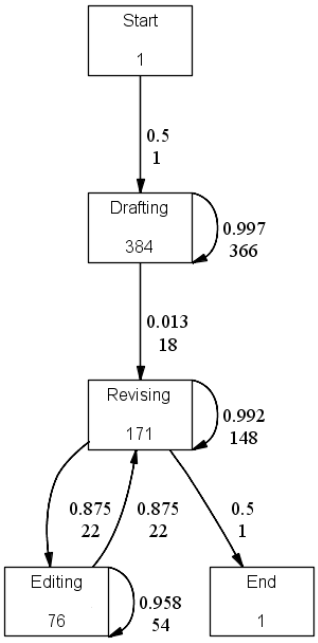


Figure 3-2. An example of dependency graphs.

This thesis focuses on process models representing causal dependencies, for instance, if an activity (event) is always followed by another activity (event) it is likely that there is a dependency relation between both activities (events). Process mining algorithms like Heuristic Miner algorithm (Weijters & Ribeiro, 2010; Weijters et al., 2006) can automatically generate these kinds of process models. Figure 3-2 shows an example of dependency graphs of writing processes. The numbers in the boxes indicate the frequency of the writing activities. The decimal numbers along the arcs show the dependency measures (described below) of transitions between two activities, and the natural numbers indicate the number of times this order of activities occurs among the five types, start, end, drafting, revising, and editing.

Heuristic mining algorithm as described in (Weijters & Ribeiro, 2010) generates dependency graphs. Moreover, this algorithm takes frequencies of events and sequences into account when constructing a process model. The basic idea is that infrequent paths should not be incorporated into the model. Both the representational bias provided by dependency graphs and the use of frequencies makes the approach able to handle noise in the log files and much more robust than most other approaches.

There are three basic relations between two activities based on the sequence of their execution. The following *a* and *b* are two activities in a sequence of an event log, *W:*

1. *a*>*b*: *a* is directly followed by *b* (direct successor)
2. *a*>>*b*: *a* is directly followed by *b* and then by *a* again (length-two loops)
3. a>>>b: a is eventually follow by b (indirect successor)

Note that length-one loops are the relations of a>a.

The heuristic mining algorithm only considers mainly the first two relations. Particularly, the algorithm uses the dependency measure, defined below. |a>b| is the number of time a > b occurs in the sequence W.

$$a \to b = \frac{|a > b| - |b > a|}{|a > b| + |b > a| + 1} \; if \; (a \neq b)$$

$$a \to a = \frac{|a > a|}{|a > a| + 1}$$

$$a \to^2 b = \frac{|a \gg b| - |b \gg a|}{|a \gg b| + |b \gg a| + 1}$$

First, remark that the value of $a \rightarrow b$ is always between -1 and 1. Some simple examples demonstrate the rationale behind the equations above. If in 5 traces, activity A is directly followed by activity B but the other way around never occurs, the value of $A \rightarrow B$ = 5/6 = 0.833 indicating that we can not be completely sure of the dependency relation (only 5 observations possibly caused by noise). However if there are 50 traces in which A is directly followed by B but the other way around never occurs, the value of $A \rightarrow B$ = 50/51 = 0.980 indicates that the probability of the dependency relation is high. If there are 50 traces in which activity A is directly followed by B and noise caused B to follow A once, the value of $A \rightarrow B$ is 49/52 = 0.94 indicating that the probability of the dependency relation is high.

A high value of $a \rightarrow b$ strongly suggests that there is a dependency relation between activities a and b. The algorithm computes the dependency measures of all relations of all pairs of activities and constructs the dependency diagrams based on the dependency measures and user-defined parameters as explained below.

3.2.2.1 Parameters of Heuristic Miner

This thesis uses Heuristic Miner (HM) algorithm, which was implemented on a process mining framework, ProM (ProM, 2013). There are two different options to construct dependency graphs: with and without “all-tasks-connected”, in which “tasks” refer to activities.

Without using the all-tasks-connected option, three threshold parameters are available in the HM to indicate that we will accept a dependency relation: (i) the dependency threshold, (ii) the length-one loops threshold and (iii) the length-two loops threshold. However, by using different parameters it is, for instance, possible to build a model without length-one loops (choose the length-one loops threshold = 1.0). With these thresholds, one can indicate what dependency relations are accepted between activities that have a dependency measure above the value of the dependency thresholds resulting in a control-flow model with only the most frequent activities and behaviour. By changing the parameters one can influence how complete the control-flow model becomes (Weijters & Ribeiro, 2010).

The advantage of using the all-tasks-connected heuristic is that many dependency relations are tracked without any influence of any parameter setting. The result is a relative complete and understandable control-flow model even if there is some noise in the log. The underlying intuition in the all-tasks-connected heuristic is that each

non-start task must have at least one other task that is its cause, and each *non-end* task must have at least one dependent task. Using this information HM builds a work flow model taking the best candidates (i.e., with the highest $a \rightarrow b$ measure).

Without the all-tasks-connected option, HM accepts dependency relations between tasks that have (i) a dependency measure above the value of the dependency threshold, and (ii) have a dependency measure close to the first already accepted dependency value (i.e., for which the difference with the best dependency measure is lower than the value of relative-to-best threshold). However, if this heuristic is used in the context of a low-structured process the result is a very complex model with all tasks and a high number of connections. Therefore, this option is not preferable for this thesis. Full detail of parameters of Heuristic Miner can be found in (Weijters & Ribeiro, 2010)

Therefore, to extract writing process model, dependency diagrams, this thesis uses the all-tasks-connected option with the default threshold parameters. All three thresholds are set to 0.9: (i) the Dependency threshold, (ii) the Length-one loops threshold 0.90 and (iii) the Length-two loops threshold. This research also added two artificial activities: start and end to all process cases in order to specify the initial and final activities of the processes.

3.2.2.2 Conformance checking

Conformance checking is a technique to relate events in the event log to activities in the process model and compares both. The goal is to find commonalities and discrepancies between the modelled behaviour and the observed behaviour. Particularly, conformance checking techniques can be used for measuring the quality of process discovery algorithms. Determining the quality of a process mining result is difficult and is characterized by many dimensions. In his book, van der Aalst (2011) refers to four quality criteria of discovered process models: fitness, precision, generalization, and simplicity. The description of these quality criteria is explained in the book. Of the four quality criteria, fitness is the most related to conformance checking. This thesis focused exclusively on fitness (i.e., the proportion of events in the log that can be explained by a process model). Process models discovered by using a process mining algorithm like Heuristic Miner are used to extract patterns of writing activities in this research. Therefore, it is important to measure how much of

the observed behavior in the event log is captured by the process model. This measurement is indicated by the fitness.

The computation of the fitness mainly relies on two data structures: (i) the process model, which is the dependency graph (DG) and (ii) the event log that contains information about the ordering of the activities. One way to measure the fitness between event logs and process models is to replay the log in the model and somehow measure the mismatch. The replay of every logical log trace starts with the marking of the initial place in the model. Then, the transitions that belong to the logged events (activities) in the trace are read one after another. While replay progresses, we count the number of tokens that had to be created artificially (i.e., the transition belonging to the logged event was not enabled and therefore could not be *successfully executed*) and the number of tokens that were left in the model, which indicate that the process was not *properly completed*. The value of *fitness(L,N)* defined in (van der Aalst, 2011) is between 0 (very poor fitness) and 1 (perfect fitness). The intuition of *fitness(L,N)* = 0.9 is that about 90% of the events can be replayed correctly. This thesis calculates the fitness of a process model using the fitness utility of ProM (ProM, 2013).

3.2.3 Mining Additional Perspectives of Writing Processes

The main focus of process discovery is on the control-flow perspective (i.e. the ordering of activities). However, event logs as described in Subsection 3.2.1 usually consist of a rich of information associated with other perspectives such as the process case perspective, and the organisational perspective. This subsection will describe these two perspectives.

A first step in any process mining project is to get a feeling for the process and the data in the event log. Dotted Chart Analysis (Song & van der Aalst, 2007) provides a way to achieve that. The dotted chart is similar to a Gantt chart (Song & van der Aalst, 2007), and shows the spread of events over time by plotting a dot for each event in the log. In other words, a dotted chart provides a snapshot of process cases. In a dotted chart, each event is depicted as a dot in a two dimensional plane. The horizontal axis represents the time of the event. The vertical axis represents the *class* of the event. Different classes of events can be viewed, for instance, resources. In addition, the shape and colour of a day may depend on other attributes. Figure 3-3 shows an example of dot charts provided by from ProM. In this figure, each row

presents a process of reviewing a document. Each dot depicts an event of reviewing activity: grey denoted an event when a document was reviewed by by authors; white by appointed reviewers, black by reviewers' group member (indicated by ovals) and brown by others (indicated by rectangles).

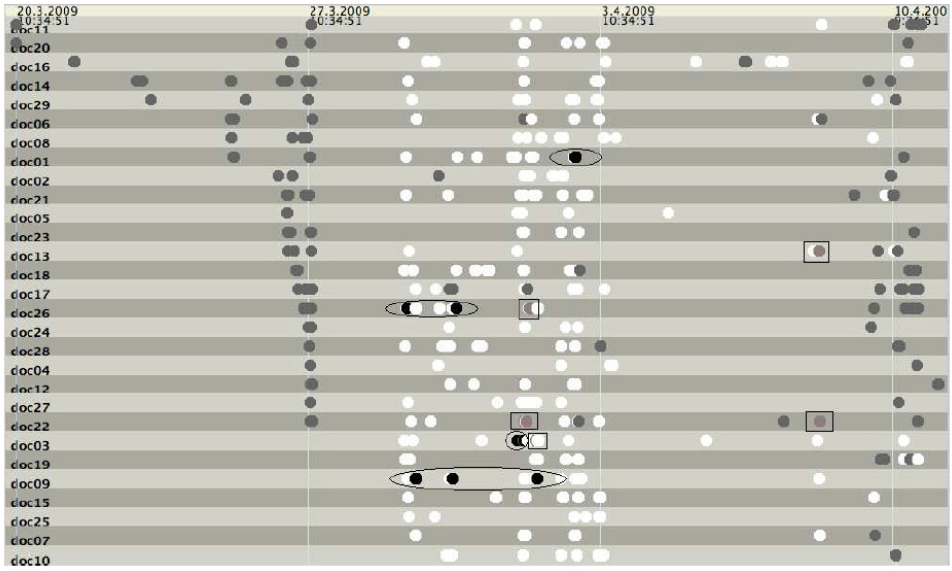


Figure 3-3. Dot chart of reviewed documents ordered by their first events' timestamps. Grey denoted events generated by authors; white by reviewers, black by reviewers' group member (indicated by ovals) and brown by others (indicated by rectangles) (from (Southavilay et al., 2009))

Another way to get an overview of the process and the data in the event log is based on resources like people and/or organizational structures (roles and departments) in order gain an insight on how work is distributed and people coordinated. Organizational mining focus on the organizational perspective (Song & van der Aalst, 2008). The organizational mining is typically based on the resource attribute present in event logs. The aim of this algorithm is to learn more about people, organizational structures (roles and departments), work distribution, and work patterns. In this thesis, organizational mining algorithm is used to extract the collaboration based on types of writing activities. More detail about organisational mining is covered in (van der Aalst, 2011).

3.3 Hidden Markov Model

This section describes the general Markov models and Hidden Markov Models (HMMs). First, the Markov property and the characteristics of Markov models (or Markov chains) will be introduced. After that, HMM and the main problems

involved with HMM will be described next. Rabiner (1989) provides a detail introduction to Markov models and HMMs. The description in the section is used in the analysis and validation of writing process, discussed in later chapters.

3.3.1 Markov Models

Markov models are used for training and recognizing sequential data, such as speech utterances, temperature variations, biological sequences, and other sequence data. In a Markov model, each observation in the data sequence depends on previous elements in the sequence. Consider a system where there are a set of distinct states, $S = \{1, 2, \ldots, N\}$. At each discrete time slot t, the system takes a move to one of the states according to a set of state transition probabilities P. The state at time t is denoted as s_t.

In a Markov model, the prediction of the next state and its associated observation only depends on the current state, meaning that the state transition probabilities do not depend on the whole history of the past process. This is called a first order Markov process. Give example.

$$P(X_{t+1} = s_k | X_1, X_2, \ldots, X_t) = P(X_{t+1} = s_k | X_t)$$

Because of the state transition is independent of time, we can have the following state transition matrix A:

$$a_{ij} = P\left(X_{t+1} = s_j \middle| X_t = s_i\right)$$

a_{ij} is a probability, therefore:

$$a_{ij} \geq 0, \forall i, j \sum_{j=1}^{N} a_{ij} = 1$$

Also we need to know the probability to start from a certain state, the initial state distribution:

$$\pi_i = P(X_1 = s_i)$$

Thus, $\sum_{i=1}^{N} \pi_i = 1$.

In a Markov model, the states from which the observations are produced and the probabilistic observation functions are known so we can regard the state sequence as the observation. Therefore the state transition probability and the initial state distribution are the only parameters.

3.3.2 Hidden Markov Model

In many real world applications, the Markov model described in the previous subsection has limited power because states of a system may not be directly observed. Therefore, we extend it to a model with greater representation power, the Hidden Markov Model (HMM). In an HMM, one does not know anything about what generates the observation sequence. The number of states, the transition probabilities, and from which state an observation is generated are all unknown.

There are many types of representations of HMM such as a time-slice view, stochastic finite-state automaton (SFSA), and dynamic graphical model (Bilmes, 2006). This thesis uses the SFSA presentation because it shows the underlying hidden Markov chain topology. One example of this presentation is shown in Figure 3-4.

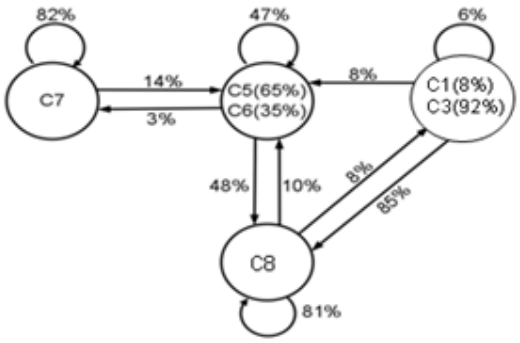


Figure 3-4. An example of Hidden Markov Model.

Instead of combining each state with a deterministic output (such as adding, deleting , and changing paragraphs etc), each state of the HMM is associated with a probabilistic function. At time *t*, an observation o_t is generated by a probabilistic function $b_j(o_t)$, which is associated with state *j*, with the probability:

$$b_j(o_t) = P\ (o_t|X_t{=}\mathrm{j})$$

In general, a HMM is composed of a five-tuple: (S,K,Π,A,B).

1. $S = \{1, 2, \ldots, N\}$ is the set of states. The state at time *t* is denoted s_t.
2. $K = \{k_1, k_2, \ldots, k_M\}$is the output alphabet. In a discrete observation density case, M is the number of observation choices. In our case, M equals the number of writing activities.
3. Initial state distribution $\Pi = \{\pi_i\}$, $i \in S$. π_i is defined as

$$\pi_i = P(s_1 = i)$$

4. State transition probability distribution $A = \{a_{ij}\},\ i,j\ \in S$

$$a_{ij} = P(s_{t+1}|s_t), 1 \leq i,j\ \leq N$$

5. Observation symbol probability distribution $B = b_j(o_t)$. The probabilistic function for each state j is:

$$b_j(o_t) =\ P\ (o_t|s_t{=}j)$$

Based on above definition, three fundamental problems have been investigated for hidden Markov models (Alpaydin, 2010):

- Given an observation sequence, how to compute the probability of the sequence given a hidden Markov model?
- Given an observation sequence and a hidden Markov model, how to compute the most likely "hidden path" in the model?
- Given a set of observation sequences, how to derive the hidden Markov model that maximizes the probability of producing these sequences?

The last problem is related to this thesis. Given a set of observation sequences, the algorithm that constructs HMMs derives an optimal set of the parameters (π, A, B) that maximizes the likelihood of the input sequences. In addition, simpler models are preferable because they are easier to interpret (Occam's razor principle). In order to achieved that, this thesis applies the technique described in Jeong et al. (2010). The technique uses an algorithm developed by Li and Biswas (2002) that employs the Bayesian information criterion (BIC) to trade off simplicity of the mode against information provided by the model. BIC (Schwarz, 1978) is defined as

$$BIC =\ -2 * \ln V + k * \ln(n)$$

k is the model size, n is the number of observations, $k * \ln(n)$ is the penalty term. The purpose of BIC is to find the model that strikes a balance between high likelihood and low complexity (Li & Biswas, 2002).

Finding the optimal HMM parameters from data is an optimization problem. Two common iterative convergence optimization schemes are the Baum-Welch (Rabiner, 1989) and the segmental K-Means (Juang & Rabiner, 1990) algorithms. The technique described in (Jeong & Biswas, 2008; Jeong et al., 2010) uses the segmental K-Means algorithm in conjunction with BIC for iterative segmentation and optimization steps to achieve the optimal model parameters including (π, A, B) and the number of states in the model, k. The segmentation step uses the Viterbi algorithm (Viterbi, 2006) for sequential decoding, while the optimization step finds a

new set of model parameters as dictated by the K-Means method (Juang & Rabiner, 1990). A chief advantage of the K-Means algorithm is its faster execution time gained by setting a restricted optimization objective.

3.4 Summary

This thesis combines two techniques to extract process models and visualisations: process mining, which extracts process-related knowledge from event logs and text mining, which extracts semantic knowledge about what students wrote or edit during their writing tasks. This chapter provides the theoretical background of both techniques. It begins with the discussion of text mining techniques, especially Latent Semantic Analysis and two different types of algorithms for extracting topics: a LSA-based document clustering algorithm, Lingo and probabilistic topic modelling or Latent Dirichlet Allocation. In addition, this chapter discusses two types of process models used in this thesis: dependency diagrams and transitional state diagrams. The dependency diagrams are generated by a process mining algorithm, Heuristic Miner implement in a process mining framework, ProM, whereas the transitional state diagrams are created by Markov Model and Hidden Markov Model. The techniques to construct these two types of process modes are discussed in this chapter. This chapter also introduces event logs as well as several terminology used in process mining context and described conformance checking technique used for Heuristic Miner algorithm.

CHAPTER 4

OVERVIEW OF THE APPROACH

The purpose of this thesis is to develop techniques that can provide insights into the process of collaborative writing, and to use these insights to give feedback to students (authors) while they are engaged in collaborative writing and to education researchers and teachers both during and after their involvement in the writing process. To achieve these aims, a range of process models and representations that offer different views of the writing processes are extracted. The process models and representations are based on basic events that are considered to be collaborative writing activities. These activities are discovered automatically, based on the semantic changes of text edits made to each document revision, using text features such as types of text edits, text structures, number of words, sentences, and paragraphs, topics, and cohesion changes.

Before describing the approach, it is important to distinguish between two terms: "revision" and "revision history". In this thesis, a revision refers to one version of the written document, whereas a revision history is a record of revisions and their

metadata such as timestamps and author IDs. Therefore, for a particular document, there can be one or many revisions. However, a document has only one revision history consisting of metadata of its revisions.

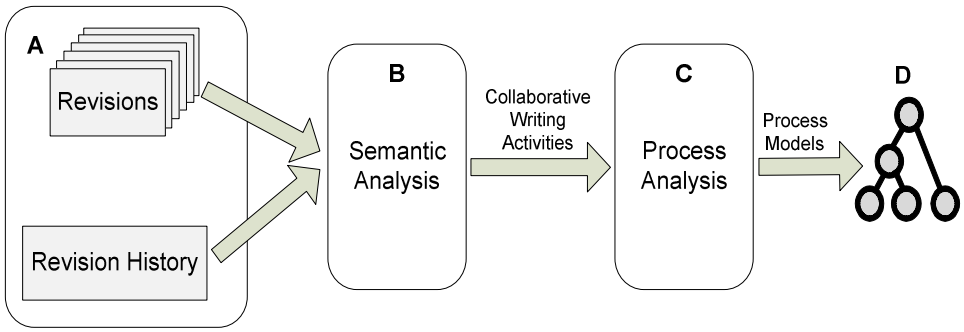


Figure 4-1. Overview of approach for extracting and analysing process models.

Figure 4-1 illustrates the steps of my overall approach:

- Automatically retrieval of content texts of document revisions and revision history from writing tools in the cloud like Google Docs **(A)**.
- Extraction of collaborative writing activities based on text features -- types of text edits, text structures, number of words, sentences, and paragraphs, topic and cohesion changes -- by using a set of heuristic. In order to extract the text features, several text mining algorithms are used extensively **(B)**. Note that text structure refers to the structure of the written documents, such as sections, and paragraphs. It does not involve syntactic analysis like part of speech, coreferentiality, etc.
- Discovery of a range of process models and representations by using the extracted writing activities and revisions, with techniques based on process mining and machine learning algorithms **(C)**.
- Creation of various types of visualisations of the process of writing activities that provide several semantic levels of text edits and topic changes made to documents during the writing processes **(D)**.

As mentioned in the previous chapter, process mining techniques have been applied successfully in extracting business process workflows and models. However, unlike business workflows, which are sequences of predefined events considered in advance to support business transactions for particular business organizations, a writing process consists of text edits (text change operations) which are not predefined. In this thesis, a document writing process is defined as a *process*

instance (or a process case) consisting of writing activities (events). Building on the research related to writing process models and a taxonomy of collaborative writing activities that were previously addressed in the literature review chapter herein, this thesis proposes techniques to automatically extract writing activities and reveal new process models based on the nature of text edits. A number of text mining algorithms and techniques are utilised to infer the semantic meaning of text edits and automatically reveal writing activities. Based on the activities discovered process mining algorithms are then employed to identify writing states and process models. Several process models can be discovered based on the semantic level of text edits.

As explored in the previous chapter of this thesis, research findings related to supporting quality writing (Villalon & Calvo, 2011) all rest on the final product, not the writing process itself; and the aforementioned studies (Caporossi & Leblay, 2011) that provide a type of visualisation of the writing process do not take into consideration the semantic meaning of text changes. In contrast with these precedents, this thesis presents the development of many types of visualisations that include several embedded semantic levels of text edits to generate process models through the use of text mining and process mining techniques already discussed. These visualisations are then offered as feedback to writers, and provide increased awareness with regard to the workings of writing activities of individual authors.

This chapter presents an overview of the approach proposed in this thesis, which involves text and process mining and can is illustrated in Figure 4-1. The first step consists of developing a framework to explore and analyse writing processes by using a Web 2.0 writing tool in the cloud, such as Google Docs, followed by the development of techniques to identify writing activities based on text mining algorithms. After finding these writing activities, process models and visualisations of writing activities are extracted and subsequently presented to the writers themselves.

4.1 Framework for Analysing Writing Processes

Several writing tools exist that provide support for logging versions of written documents. These tools can be classified according to the granularity of the logged versions. At the two opposite extremes of the classification spectrum are key-stroke logging tools and version controlled documents.

Key-stroke logging applications include tools such as InputLog (Leijten & Van Waes, 2006), which was developed specifically for capturing every key stroke (character) typed by individual writers as a method of analysing writing processes. Although InputLog logs all types of input -- keyboard, mouse, and speech recognition -- it can only run on stand-alone mode, which makes it an unsuitable tool for collaborative writing (though it has been used, as mentioned before, to analyse individual writer's processes in studies by Caporossi and Leblay (2011); Leijten and Van Waes (2006); and Tillema et al. (2011).

On the other end of the granularity spectrum are applications that support collaborative writing, such as EtherPad (EtherPad, 2013), Google Docs (Google Docs, 2013), and Wikipedia (Wikipedia, 2013), which automatically store revisions and provide revision histories including timestamp and authorship information. The cloud tool Google Docs was selected to use as a front-end collaborative writing tool for this thesis.

iWrite, as reviewed in Section 2.4 and WriteProc were chosen as the framework for exploring and analysing collaborative writing processes in this thesis, with students performing writing activities through iWrite, a system that supports the authors' collaborative writing (Calvo et al., 2011). iWrite manipulates the group and individual interactions in Google Docs; a document is created in iWrite and assigned to groups of students by administrators or instructors. WriteProc, described in detail in the next chapter, retrieves the documents directly stored in Google Docs and uses text mining and process mining algorithms to identify writing activities and extract writing processes. Along the way as students write and edit on the documents, then finally submit their final versions, they have access to a reviewing tool (Calvo et al., 2011) provided by the system should they wish to use it. The main aspect of interest for this thesis is the fact that Google Docs stores all revisions (versions) of a document that students make including a revision history with timestamps and author ID information for each revision of the document. Google Docs also provides an application programming interface (API) for developers to retrieve document revisions and their histories. Using the API, WriteProc downloads all revisions and revision histories of individual documents in order to extract process models and perform process analysis, as described below.

4.2 Identifying Writing Activities Automatically

A rule-based heuristic is used to extract the semantic meaning of text edits and to identify the types of writing activities that are performed. The heuristic developed in this thesis is based on types of text edits, cohesion changes, and topic overlap. First, a text comparison unit consisting of text difference algorithms is developed to compare the texts in two consecutive revisions and to identify the types of text edits, which are adapted from (Boiarsky, 1984). In conjunction with finding the text edits, a layer of semantics is also formed, using topic changes and cohesion changes. To achieve this, topics are first extracted for each revision, and then compared with previous revision topics to reveal the topic change. Likewise, for each revision, cohesion is first computed by averaging the similarity of all pairs of consecutive paragraphs, then compared with the previous revision to reveal the cohesion change. Based on the text edits, topic changes and cohesion changes that are uncovered, writing activities can be inferred using a heuristic set. The details and an evaluation of this technique for identifying writing activities are presented in Chapter 5.

4.3 Extracting Process Models of Writing Processes

Based on sequences of the discovered writing activities, process models of writing can be automatically derived using a process mining tool such as ProM (ProM, 2013) and a dynamic graphical model such as the Hidden Markov Models (HMM). For the purposes of this thesis, a Heuristics Miner algorithm (Weijters & Ribeiro, 2010; Weijters & van der Aalst, 2003), implemented in ProM is applied to extract writing process models presenting causal dependency diagrams. In addition, based on several layers of semantic (i.e. text edits or writing activities), dynamic graphical models embedded Markov assumption can be derived. These process models are then utilised to carry out process analysis and identify the patterns of text edits and writing activities that students perform during their tasks.

4.4 Visualising Writing Processes

The final item in the research approach of this thesis is to provide a representation and visualisation of writing processes. In addition to exploring the existing process visualisation provided by ProM, such as dotted charts (Song & van der Aalst, 2007), causal dependency diagrams (van der Aalst, 2011), and performance sequence

graphs (Bozkaya et al., 2009), this thesis also proposes the following new visualisations as feedback that can be offered to both instructors and students (writers):

- Revision maps
- Topic evolution charts
- Topic-based collaboration networks

Revision maps are first created for showing a snapshot of the text edits performed by students on their jointly authored documents. This form of visualisation depicts the development of documents at the paragraph level over a period of time. Based on the text edits made to the paragraphs, topics are then extracted through the use of several types of probabilistic topic models. The resulting topic evolution charts provide insights on how topics are created and developed during the writing process. Finally, topic-based collaboration networks are established to analyse student collaboration based on the topics found in their writing. The topic-based collaboration networks present network diagrams that show those students who share topics in common, i.e., who write about the same topics during their tasks.

Because collaborative writing is very demanding both organisationally and cognitively, it is crucial to acquire a better understanding of text edition to achieve effective and efficient collaborative writing (Lowry et al., 2003). All three types of visualisations previously nominated here are intended as a feedback mechanism about writing processes that can be given to instructors and the student writers. For instance, by examining the revision maps, students can answer three basic questions regarding text editions performed on their documents: They can see which portions (sections or paragraphs) have been edited recently, when these changes were made, and who made the modifications. Having this information helps writers to understand and follow the development of their document, and makes it much easier for them to coordinate work with their team members and carry out their tasks more efficiently and effectively. The visualisations also provides support for teachers, who can use them as a means for monitoring groups more effectively and detecting problems early in the writing process.

4.5 Summary

This thesis proposes new methods of analysing collaborative writing, with the purpose of obtaining information that can better understand the writing process. This information can be a source of useful feedback for students (authors) while they are engaged in collaborative writing, and can also be exploited by education researchers and teachers both during and after collaborative writing exercises.

The first step in achieving this goal was the creation of a framework for retrieving revisions and revision histories from the Web 2.0 writing tool Google Docs. These revisions and revision histories retrieved are then used to develop a technique for automatically extracting collaborative writing activities and to discover process models of the writing processes. This technique development and process model discovery are accomplished by employing text mining techniques for extracting topics and measuring cohesion, from which the purpose of text edits during collaborative writing can be inferred and the corresponding collaborative writing activities can be identified. Process mining and machine learning algorithms are applied to discover writing process models representing different views of semantic levels of text edits. Finally, this thesis concludes by proposing a range of visualisations that represent different views of collaborative writing processes.

CHAPTER 5

WRITEPROC: A FRAMEWORK FOR DATA COLLECTION

This chapter introduces WriteProc as a framework for collecting all the data required for semantic and process analysis as described in Chapter 6 and 7, respectively. This framework corresponds to box A of Figure 4-1 in the previous chapter, and consists of the writing tool Google Docs, which provides support for collaborative writing activities; and an application, which retrieves necessary revisions and metadata (i.e. identifications of authors who produce the revisions and timestamps indicating when the revisions were created) from Google Docs.

5.1 Overall Conceptual Description

WriteProc integrates a front-end writing tool, Google Docs, which not only supports collaborative writing activities, but also stores all revisions of documents created,

shared and edited by groups of writers. Each revision of a particular document contains the following information:

- The entire text content of the revision.
- The timestamp (date and time) that the revision was created.
- The user identification of the writer who edited the text.

In order to analyse the writing process of a particular document, the information pertaining to the three items above and all the document revisions must be retrieved and traced by using an open-source application programming interface. The content text of revisions is used for semantic analysis to specify the types of text edits, topics and cohesion changes made by the authors. The resulting semantic analysis is utilised to identify collaborative writing activities, which are explained in detail in the next chapter. Based on the information delineated above (i.e. timestamp, author identification, edited text) and on the identified writing activities, process analysis is then employed to discover process models of writing processes.

Figure 5-1 depicts WriteProc, the framework composed of the Google Docs writing environment and the Google Documents List API application used for retrieving revisions and their information as previously explained. These two components are described in detail in the subsequent section.

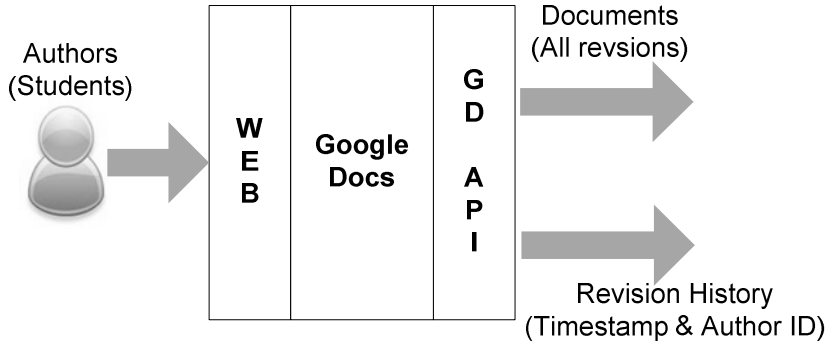


Figure 5-1. WriteProc: a frame work retrieving revisions and revision histories.

5.2 Writing Environment: Google Docs

Earlier in this thesis, it was established that in order to explore the workings of the writing process, it is necessary to have a front-end writing tool that supports collaborative writing (CW) and stores all revisions and their metadata. For this reason, tools such as Microsoft Word or OpenOffice, which do not allow CW, were not viewed as viable choices for this work, although these tools do provide some

functionality to detect changes and who produced those changes. Web 2.0 tools such as Google Docs and the incipient Microsoft Word Live allow users to write on a web application, or to write offline and synchronise the material later, with the service provider storing the different versions of the document. For this reason, Google Docs was selected as the front-end writing tool for this work.

Google Docs is a web-based utility with most of the functionalities necessary for word processing which allows users to share documents with other team members and to write synchronously. Authors can access Google Docs through their web browsers from anywhere and at anytime they choose. Each author requires a Gmail account in order to access the tool that he or she can obtain from Google free of charge; this author's Gmail account is referred to as the *author ID* in the framework of this study.

The writing process begins with the creation of a particular document that is then assigned to a group of students by course administrators/lecturers. Students work on the documents by writing and editing, after which they submit a final version. As previously noted, the crucial aspect of this particular writing process is the fact that Google Docs stores all revisions and revision histories made from beginning to end.

Each document created in Google Docs created is assigned a unique document identification number (i.e. *document ID*). Google Docs also keeps track of all versions by incrementally numbering each subsequent one (i.e. *revision ID*) every time the document is edited. Whenever an author makes changes and edits a particular document, Google Docs stores the edited content text and keeps a record of the following information in the revision history:

- The version number (revision ID).
- The identification of the author (author ID).
- The timestamp (date and time) of the changes made

There are occasions when many authors engage in editing the same content in a document at almost the same time. In this case, since Internet connection speeds are not instantaneous, when an author makes a change, he or she temporarily creates a local version of the document that is different from the versions that other collaborators see. When this occurs, Google Docs implements a mechanism to make

sure that all the text change operations eventually converge on the same correct version of the document[1].

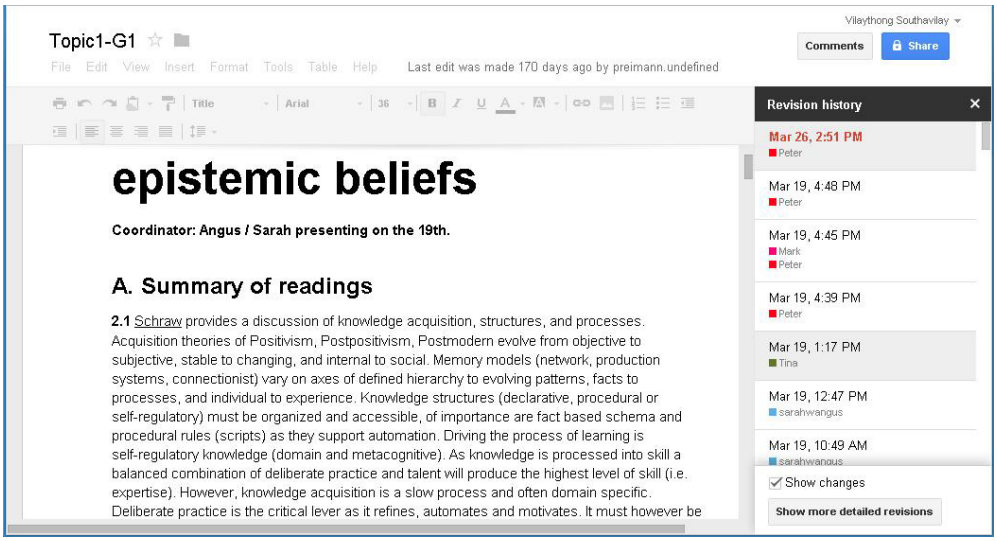


Figure 5-2. the web-based interface of revision history (on the right panel) of Google Docs, which shows a list of revisions. Each revision contains a timestamp (date and time) and an author ID (different colours for different authors).

Authors can also access the revision history of their documents by using the web-based interface via the command "*see revision history*" under the *"file"* main menu on Google Docs, as shown in Figure 5-2. Google Docs displays the web-based interface of the revision history on the right-hand panel, which includes a list of revisions containing timestamps and author IDs of the corresponding edits. Different authors are assigned different colours for identification. The web-based interface incorporates two types of revision history for each document, designated as *more* and *less detailed revisions.* For both types of revision history, an author can select a particular revision to view the edited content.

Since Google Docs automatically saves documents every few seconds even when no changes have occurred[2], there may be any number of revisions for any particular document.

[1] Since 2011, Google Docs uses a new algorithm for merging changes called *operational transformation*. It also uses the *collaboration protocol* to make sure that each author knows when there are changes that need to be merged. Please see the three white papers "what's different about new Google Docs" (Google Docs White Paper, 2010a, 2010b, 2010c) for a thorough detailed explanation of the *operational transformation* and the *collaboration protocol* (these two were originally developed as engines driving Google Wave).

[2] This framework was implemented in 2010. In 2011, Google has changed the auto-saving functionality so that Google Docs only auto-saves when there is a change in the content text of the documents.

As previously elaborated, this research uses an application programming interface (API) with the goal of automatically retrieving all authors' revisions and revision histories of documents. Before describing the API, however, it is important to first define some terminology used in this thesis.

A *writing session*, as defined for the purposes of this study, is composed of consecutive revisions that are made less than 30 minutes apart. A time threshold of 30 minutes was established to distinguish between writing sessions, as used in the data analysis for web usage mining (Markov & Larose, 2007). If two consecutive revisions show a timestamp difference of more than 30 minutes, the later revision becomes the first revision of a new writing session. Every author's writing sessions are determined according to this 30-minute cut-off. It is considered that students perform their text edits continuously during a writing session. The inactive time that occurs when students pause to read the text written so far and to think about what they are going to write next should consist of a fairly short interval (pause). If this inactive time becomes longer, it is assumed that a new writing session follows.

Since Google Docs automatically saves documents frequently, the resulting number of revisions is very high and must be reduced to a more manageable size. This is accomplished by grouping the revisions into *major revisions*, which are defined as the final revisions that end a writing session. All revisions within a major revision originate from the same author. In this thesis, the creation of major revisions is performed by WriteProc after retrieving all revisions. Figure 5-3 shows an example of revision history and major revisions.

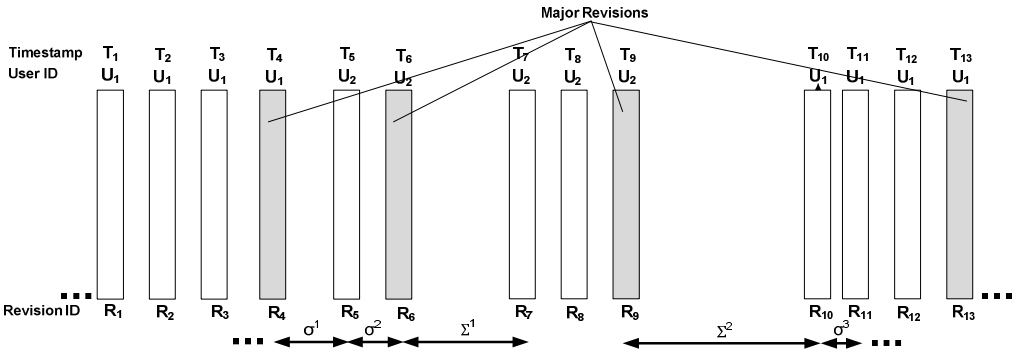


Figure 5-3. Revision history before 2011 showing 13 revisions: R1-R13 written by 2 authors: U1 and U2. Each revision has timestamp associated with it. Σ and σ are time difference of two consecutive revisions, where Σ>30 min and σ≤30 min.

From 2011, Google has changed the auto-saving functionality so that Google Docs only auto-saves when there is a change in the content text of the documents. In addition, as mentioned previously, Google Docs implements a mechanism to make sure that all the text change operations eventually converge on the same correct version of the document. Therefore, the number of revisions has been reduced significantly. For a particular document, the revision history retrieved by Google Document List API (verstion 3.0) is the less detailed revision history shown in the web-base interface, as described above. For data collected since 2011, the reduction of the number of revisions is no longer needed. This thesis considers the revisions of the less detailed revision history as major revisions, which have timestamps and authors' IDs associated with them. Although all revisions including the ones shown in the more detail revision history can be downloaded, their timestamps and authors' ID are not available. Figure 5-4 shows the revision history provided by Document List API since 2011.

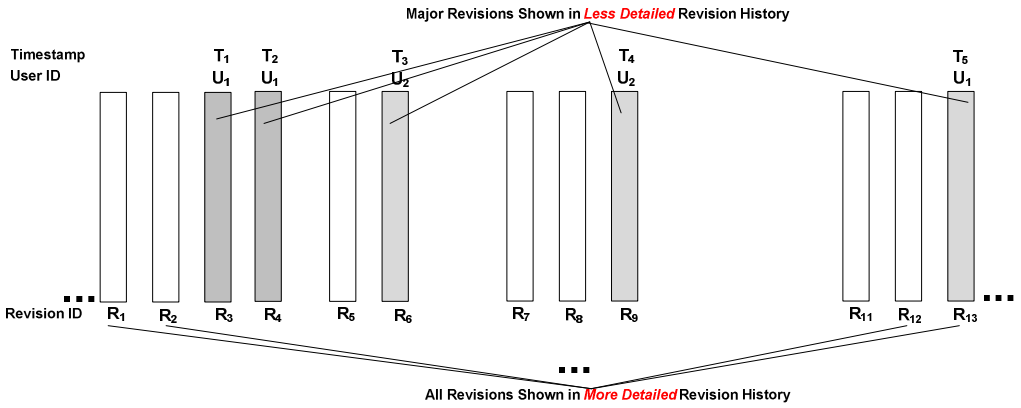


Figure 5-4. Revision history since 2011. Only revisions displayed in "less detailed" revision history have timestamp and user IDs.

5.3 Google Document List API

The last component of the framework for this research is Google Document Lists Data API (Google Documents List API, 2012) which is used to integrate Google Docs to WriteProc, as shown in Figure 5-1. The API allows the framework to retrieve and track all versions of documents that are created, shared and edited among groups of students. Every time students make changes and edit a particular document, the edited content text and the revision history of the document can be retrieved and stored at the central relational database of the framework by using the

API. This information extraction is executed seamlessly in the background; users/writers are not aware of it and are able to perform their writing tasks just as seamlessly. The API also provides the ability to build an interface to create and share documents in the collaborative writing environment. This can be very helpful for instructors or supervisors to create and assign documents to groups of writers and reviewers without accessing Google Docs. An appointed owner can edit a document, whereas an assigned "*viewer*" can only review the document.

In this thesis, the framework uses Google Document List API version 3 implemented in Java. The API allows developers to create, retrieve, update, and delete Google Docs (including but not limited to text documents, spreadsheets, presentations, and drawings), files, and collections, and also provides advanced features such as resource archives and revision history (Google Documents List API, 2012). As previously noted, for each document the Google Documents List API is used to retrieve content texts of revisions and the revision history containing metadata.

In order to access Google Docs using the API, WriteProc uses three Java classes: folder, document, and revision. An instance of folders contains one or more document instances, in which each document instance consists of one or more revisions. The attributes of these classes are described below:

- Folder: folder ID, *changestamps* (a unique integer incremented every time there is a change to the corresponding folder), and document list.
- Document: document ID, *changestamps* (similarly a unique integer incremented every time there is a change to the corresponding document), and revision history.
- Revision: revision ID, author ID, and timestamp.

In addition, WriteProc consists of a relational database containing three main tables that represent the above three main classes. This database also indexes texts of revisions using Apache Lucene (Lucene, 2013).

The procedure for retrieving revision history and revisions is described below, assuming that the document ID is known for a particular document as created by course administrators/lecturers, and that permission exists to access its revisions and revision history:

1. For each document ID, retrieve the *changestamps* and use it to identify changes made to the corresponding document.
2. If there is a change, download the updated revision history containing new record: revision ID, author ID and timestamp, using document ID and storing the record in Revision table.
3. For each record of new metadata, download (content) text using the corresponding revision ID and indexing the text using Lucene.

This procedure is executed automatically every time there is a text edit performed on particular documents monitored by WriteProc, and also operates offline without the authors' awareness, allowing them to perform their writing tasks seamlessly.

5.4 Dataset for Analysing Writing Processes

A dataset is described here as it will be used in subsequent chapters (6 and 7) to illustrate and validate techniques for identifying collaborative writing activities and extracting writing process models. In the following chapters, this dataset will be referred to as Dataset A. This is to distinguish it from the data collected during an experimental study for visualising writing processes, which will be described later in Chapter 8.

This dataset was collected during a course of E-business Analysis and Design (ELEC3610), conducted during the first semester of 2010 at the University of Sydney. In this course, 52 students were organised in groups of two and asked to write a project proposal comprising of 1,500 to 2,000 words. This writing assignment took one month to complete and was a graded assignment, which counted 30% toward the course grade. iWrite (Calvo et al., 2011) was used to manage activities and documents on Google Docs. The activities involved a draft submission after which students peer-reviewed two other proposals (from different groups). After getting feedback from their peers, students could revise and improve their documents if necessary before submitting the final version two weeks later. Figure 5-5 depicts the timeline of the course. In total there are 26 documents in this dataset.

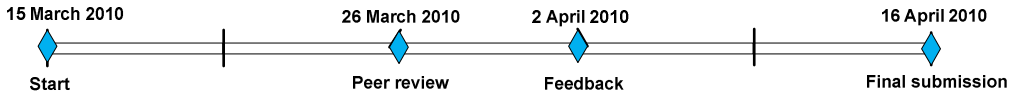


Figure 5-5. Timeline of assignment due dates in the case study.

Each document consists of five sections: 1) Introduction; 2) Background; 3) risk and opportunities, 4) total cost of ownership, and 5) conclusion. Section 1, 2 and 5 were required to be shared and written by all members of the groups, whereas each member was required to select and write either Section 3 or 4 individually. The final assessments of individual documents were added to the dataset for the analysis of the writing processes.

Table 5-1. Number of Revisions and the final marks of 26 documents of Dataset A, in which documents are ordered by the final marks.

Document	**Number of All Revisions**	**Number of Major Revisions**	**Final Mark (out of 100)**
3	2320	148	91
7	356	14	91
23	1434	101	90
16	1200	126	85
18	1276	27	78
22	1785	95	76
11	912	28	75
19	6688	432	74
4	1311	140	68
12	1690	113	68
13	2090	165	68
17	1105	42	68
25	1397	67	68
6	2172	161	65
20	2329	286	65
26	1040	79	65
1	583	46	64
8	1790	80	63
9	2354	242	61
21	2209	147	61
5	1394	81	58
14	1524	98	56
2	2111	39	54
24	1513	40	45
15	1062	15	39
10	1349	143	38
Total	44994	2955	
Mean	1730.54	113.65	
STD	1145.08	93.36	

For each document, content texts of all revisions and its revision history were obtained using WriteProc, described above. Table 5-1 shows the numbers of all revisions and major revisions and the final marks of 26 documents in the dataset. There are 2955 major revisions (M=113.65 revisions per document, STD=93.36) in total. The maximum number of major revisions is 432, whereas the minimum number of major revisions is 14. The final marks provide overall view of student's performance and will be referred to in the process analysis in Chapter 7.

5.5 Summary

This chapter describes WriteProc, the framework for retrieving revision and revision histories used in writing process analysis and visualisation. The framework uses a front-end writing tool -- Google Docs -- for collaborative writing and an application using Google Documents List API to retrieve content texts of all revisions and the revision history of the document written by groups of students. These content texts and the revision history will be used for semantic and process analysis in order to gain further insight into the way that students write their documents, which will be explained in detail in the following chapters.

CHAPTER 6

IDENTIFYING WRITING ACTIVITIES

As a first step towards process mining and visualisations, the revisions and revision histories retrieved by WriteProc (as introduced in the previous chapter) are gathered for the purpose of identifying writing activities. Different process models and visualisations provide diverse views of workflow processes that can be suitable for use according to the specific purposes of the process analysis; but as noted in prior discussions herein, these traditional systems for process models of business workflows depend on the recording of predefined events (activities or transactions), as opposed to the events that occur during the writing process. In order to achieve writing process models that represent different views of semantic levels, the semantic levels of text edits need to be taken into account. The purpose of the text edits that are made during the collaborative writing process is used as a means of identifying the corresponding collaborative writing activities. Based on these identified

activities, process mining techniques are then employed to extract and analyse writing process models and visualisations.

This chapter will describe an automatic technique for identifying writing activities based on several text features; the explanation will begin with a description of the heuristic for detecting writing activities, followed by an evaluation of the proposed heuristic, using real documents written by groups of students.

6.1 Heuristic for Determining Collaborative Writing Activities

The heuristic relevant to this study is intended to identify the nature of the writing activities performed during writing processes. The text differences between a given revision and the previous one are used to extract important indicators for estimating the collaborative writing activities performed during that revision.

The heuristic for identifying writing activities, known as CWA heuristic, is based on several features:

- Text edits (C1-C8).
- Structure of text (S1).
- Difference between the number of sentences and the number of paragraphs (S2).
- Changes in the number of words (F1)
- Changes in topics or topic overlap (F2)
- Changes in cohesion measure (F3)

Based on the taxonomy of writing activities in a collaborative environment proposed by Lowry *et al.* (2003) as described in Chapter 2, the five collaborative writing activities nominated are brainstorming (B), outlining (O), drafting (D), revising (R), and editing (E). In addition, eight types of text edits will be discussed in subsection 6.1.2: C1 – C8. Table 6-1 summarizes the proposed heuristic.

Using this table, writing activities can be identified based on text features as follows:

- **Brainstorming** and **outlining** activities are detected by examining the structure of the text (i.e. bullet-point lists consisting of single and phrasal words for brainstorming, and ordered lists for outlining). During brainstorming, authors can reorder, add, or delete items of lists of brainstorming ideas. They can also format the lists, alter all of the items on

the lists, or change selected items. Similarly, during outlining, authors can add, delete, reorder, format, and change part of or entire sections of their organised list.

- During drafting, revising and editing, text change operations become more complicated. **Drafting** activities start when the structure of the written text changes from bullet-point or structured lists to paragraphs. In other words, alteration of form (C7) after either brainstorming or outlining usually indicates the beginning of drafting activities (as depicted by **x*** in the table). During drafting, information is added and removed on an ongoing basis; hence, expansion of information (C5), deletion of information (C6), consolidation of information (C3), distribution of information (C4) and changes in micro-structure (C8) all imply drafting activities. In addition, during drafting activities, the cohesion of the written text fluctuates greatly and topic changes overlap dramatically.
- Common **revising** activities are categorised as reordering (C2), C3, C4, and C8. These text edits first occur when authors begin drafting, then recur frequently during the writing process; it is assumed that, as they draft, authors may stop writing and revise their own edits in order to improve document cohesion and effectively convey information and ideas to readers. The cohesion of the text usually remains stable and topics are generally not changed in the course of revising activities. In addition, during revising authors may also delete the entire text and rewrite it from the beginning, which represents a C7 operation
- Micro-structure change (C8), which pertains to text edits performed on existing paragraphs, can happen during both drafting and revising activities (as noted by √* in the table). If a text edit consists of appending words to an existing paragraph, it is considered to be a drafting activity; whereas if a text edit consists of inserting, deleting, moving or replacing words in an existing paragraph, it is identified as a revising activity. During drafting, authors usually append words at the end of existing paragraphs, whereas during revising they tend to insert, delete, move or replace words in the body of the paragraphs.

- For the sake of simplicity, all surface change operations (C1) including formatting, spelling, and punctuation corrections are designated as **editing** activities; and the number of words should not change during editing activities. Similar to the outcome of revising activities, editing activities do not produce a change in cohesion or topics.

Table 6-1. Heuristic for identifying collaborative writing activities based on text edits (C1 – C8), text structure (S1 – S2), and functions (F1 – F3).

Writing Activities / Features		Brain-storming (B)	Outlining (O)	Drafting (D)	Revising (R)	Editing (E)
Surface change	C1	√	√	×	×	√
Reorganization of information	C2	√	√	×	√	×
Consolidation of information	C3	√	√	√	√	×
Distribution of information	C4	√	√	√	√	×
Addition of information	C5	√	√	√	×	×
Deletion of information	C6	√	√	√	×	×
Alteration of form (Macro-structure change)	C7	N/A	N/A	×*	√	×
Micro-structure change	C8	√	√	√*	√*	×
Structure of text	S1	List	Structured List	Sections & Paragraphs	Sections & Paragraphs	Sections & Paragraphs
# Sentences (s) vs # Paragraphs (p)	S2	$s \approx p$	$s \approx p$	$s > p$	$s > p$	$s > p$
Changes in #words	F1	×	×	√	√	×
Topic overlap	F2	N/A	N/A	√	×	×
Changes in cohesion	F3	N/A	N/A	√	×	×

Abbreviation: An operation is allowed (√) or not allowed (×), and Not applicable (N/A).

From CWA heuristic shown in the table above, writing activities can be identified based on text edits (C1-C8), text structure (S1-S2) and functions (F1-F3). First, text edits of each revision are discovered before they are mapped into writing activities using several text features. For example, if a revision contains a text edit

and the number of words is not changed from the previous revision, the text edit is identified as C1. The heuristic then checks the structure (S1) of the text of the revision and compares the number of sentences and the number of paragraphs (S2). If the number of sentences is equal the number of paragraph, the structure of the text is a list. If the list is ordered, specified by S1, this C1 text edit is classified as an outlining activity. If the list is bullet-pointed, the text edit is identified as a brainstorming activity. However, if the number of sentences is higher than the number of paragraphs, the text edit is identified as an editing activity. Another sample is a revision with a text edit and no change in topic overlap (F2). In this case, the text edit associated with this revision is categorised as C7. Thus, it is identified as a revising activity. The final example is a revision with a text edit C3, which is discovered by using text comparison utility, described below. From Table 6-1, the text edit can be classified either a drafting or revising. If the cohesion of the revision is changed from the previous one, then it is identified as a drafting activity. Otherwise it is a revising activity.

The subsections that follow will describe all the features used in CWA heuristic, commencing with an explanation of the text structures employed. My prior research (Southavilay et al., 2010) analysed text edits at the paragraph level, by identifying adding, deleting, and changing paragraphs. Since that work, I have improved the granularity of the analysis to include both paragraph and word edits. Algorithms that compare the texts of two revisions to discover text edits are described in Subsection 6.1.2, followed by a description of the number of words and phrases, cohesion and topic overlap used in the CWA Heuristic (Subsections 6.1.3, 6.1.4, and 6.1.5 respectively).

6.1.1 Text Structures

Writing activities can be determined by the structure of the written texts (S1) and the number of sentences and paragraphs (S2). During brainstorming, authors normally write bullet-point lists consisting of single words or phrasal words (compound nouns). As a result, the number of paragraphs (the number of lines) is approximately equal to the number of sentences (the number of words or items). Although during an outlining phase the number of paragraphs and sentences still remain the same, the text structure becomes more organised, separated into sections and subsections.

When authors start drafting their documents, the number of sentences and paragraphs change dramatically. During this phase, the number of sentences is expected to be higher than the number of paragraphs. This is also true with regard to the revising and editing phases.

6.1.2 Text Edits

This thesis is concerned with the following eight types of text edits or text change operations, based on the revision change functions proposed by (Boiarsky, 1984): surface change (C1), reorganization of information (C2), consolidation of information (C3), distribution of information (C4), addition of information (C5), deletion of information (C6), alteration of form or macro-structure change (C7), and micro-structure change (C8). During their writing activities, authors use these text edits for different purposes during the process of producing the final document.

The technique for detecting these text edits began with my prior work (Southavilay et al., 2010), in which the granularity of the text edits remained at the paragraph level. Specifically, edits to text changes were only associated with paragraphs, as in adding, deleting and changing paragraphs. In contrast, this thesis identifies edits not only at the paragraph level, as in my aforementioned prior research (Southavilay et al., 2010), but also at the word level – i.e. adding, appending, deleting, moving, and replacing the words in existing paragraphs. A text comparison utility is expressly developed to compare the text content of two consecutive revisions and compute the difference between them; this technique is intended to discover the specific text edits that were made during the writing process in order to transform the previous revision into the current one.

This text comparison utility uses a text differencing algorithm based on two levels of text edits, the *paragraph* level and the *word* level. At the first level, the algorithm detects six types of text edits made to paragraphs: inserting (C5), deleting (C6), moving (C2), replacing (C8), merging (C3), and splitting (C4). The first four of these text edits at the *paragraph* level formed part of the research of Fong and Biuk-Aghai (Fong & Biuk-Aghai, 2010), but merging and splitting paragraphs were not acknowledged in that work. Furthermore, C8 can be interpreted as a paragraph alteration; if the existing paragraphs are replaced or altered, the algorithm detects text edits at the word level, in order to further clarify how authors alter the text.

At the second level, the text differencing algorithm identifies five types of text edits made with regard to words: inserting (C8.1), deleting (C8.2), moving (C8.4), replacing (C8.5), and appending (C8.3). The first four of these *word* level edits are also delineated in the aforementioned Fong and Biuk-Aghai research (2010). This thesis, however, differs from that study in that the algorithm used herein also distinguishes whether new words have been appended to the existing paragraphs during writing, thus differentiating between drafting (when authors usually append words at the end of paragraphs) and revising (when authors tend to insert, delete, or replace words in the paragraphs). These two differencing algorithms are explained in Appendix B.

The number of words and topic overlap are the features used to detect text edit types designated as surface change (C1) and alteration of form (C7); this aspect is described in Subsection 6.1.4 below.

6.1.3 Number of Words and Phrases (F1)

The ratio between the number of words of two consecutive revisions is computed (F1) and used in conjunction with topic overlap and cohesion measurement (discussed below) to determine writing activities.

6.1.4 Topic Overlap (F2)

The CWA heuristic also uses a topic overlap measurement (F2) in analysing the change in topics (concepts) for two consecutive revisions. If one intuits that when people write about something, they usually repeat the subject (topics) to focus the readers' attention, it follows that identifying topics and comparing them between two consecutive document revisions can expose more information about how authors develop their ideas and concepts. Intuitively during drafting, F2 changes dramatically, whereas during revising and editing, it should be constant.

Topics are extracted from each individual revision using a technique described in Section 6.2. Topic overlap is then calculated by using the topic matching rate, which is calculated for each two consecutive revisions using the formula described below (similar to the word matching rate in Appendix B):

If we denote:

T_i as the number of topics in the old revision,

T_j as the number of topics in the new revision,

$Tc_{i,j}$ as the number of common topics between the above two revisions,

$Tm_{i,j}$ as the topic matching rate between the above two revisions,

then the topic matching rate can be computed as follows:

$$Tm_{i,j} = \frac{2 \times Tc_{i,j}}{T_i + T_j}$$

The changes in the number of words and topic matching rate from old to new revisions is computed to identify surface change (C1) and alteration of form (C7). If there is a change in text (i.e. text replacement) to transform a prior revision into a new one, but the number of words of the two revisions are the same (the ratio of the number of words is 1) and all topics retrieved from the two revisions are also the same (the matching rate is 1), it is concluded that the text change operation is C1. To detect C7, it is only necessary to verify if the topic matching rate $Tm_{i,j}$ is equal to zero (i.e. no topics in common) regardless of what types of text editions have been performed. In other words, if the two consecutive revisions have totally different topics, a total change in the form of the entire text (i.e. macro-structure change) has to have taken place. Table 6-2 illustrates a summary of the process for detecting C1 and C7.

Table 6-2. Detecting surface change (C1) and alteration of form (C7).

	C1	C7
Ratio of Number of Words	1	N/A
Topic matching rate	1	0

6.1.5 Cohesion Comparison (F3)

Another measurement used by the CWA Heuristic to detect writing activities is the cohesion of the text. The cohesion of each individual revision is measured, specifically calculating the distance between consecutive sentences and paragraphs, to shed light on the development progression of particular paragraphs and of the entire text. Thomas and Sheth (2007), who worked on automatically identifying the *semantic convergence* of Wikipedia articles, suggested that "a document can be seen as being mature, if, despite ongoing changes, it is semantically stable". Text edits

performed on semantically mature revisions are considered to be revising activities. In other words, during the drafting stage, a lot of text has been added and deleted to paragraphs; thus, the semantic distance between these paragraphs is divergent. Although paragraphs are also edited during the revising stage, the semantic distance between them is stable. As a result, the cohesion of the text fluctuates significantly during the course of drafting activities, whereas the cohesion of the text usually remains stable during the course of revising activities. There should be no change in the cohesion of the text during editing activities.

This research employs the Latent Semantic Analysis (LSA) technique to measure the cohesion of the text. In particular, for each revision, average sentence similarity is computed using LSA for single documents as described in Villalon & Calvo (2009) and the result is compared with the previous document revision in order to determine if there is an improvement in cohesion from one document revision to another.

The proposed set of heuristic is based on the manner in which text edits, text structure, the number of words, topics, and cohesion have changed. The subsequent sections present a description of the pre-processing steps in computing cohesion changes and topic overlap, followed by a presentation of the validation of the heuristic.

6.2 Pre-processing: Computing Cohesion Changes and Topic Overlap

Figure 6-1 illustrates the pre-processing steps mentioned above. The first step filters out all the revisions that do not contain changes, which reflect situations in which authors may want to review their work without altering it in any way, since -- as noted in Chapter 5 -- documents are saved automatically even when no changes are made to the contents, thus creating these unaltered revisions.

This initial filtering step is followed by determining the text edit operations in two consecutive revisions, which is carried out through the use of a text comparison utility that includes text differencing algorithms for both the paragraph and the word levels (as discussed in the previous section).

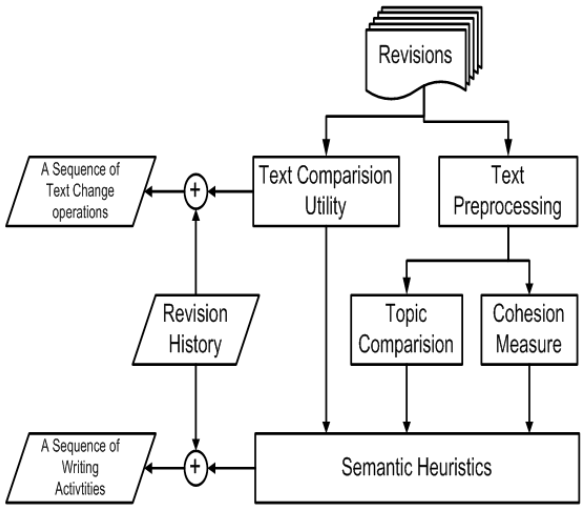


Figure 6-1. Pre-processing steps (from (Southavilay et al., 2010b))

As mentioned earlier in 6.1.5, the LSA technique is applied to measure the changes in cohesion in the text. The pre-processing step for LSA involves the extraction of terms from all relevant document revisions. Each revision is first split into paragraphs by simple matching to the newline character. Each paragraph is then divided into sentences using Sun's standard Java text library. After that step, each sentence and the entire text are indexed using Apache's Lucene (Lucene, 2013), which performs the tokenization, stemming, and stop word removal. Porter's stemmer algorithm (Snowball analyser integrated in Lucene) is utilised for stemming words, followed by the creation of a term-document matrix for each revision. Term frequency (TF) and document frequency (DF) are selected as weight terms, and terms that only appear once in each document revision are discarded. Singular Value Decomposition is then applied to reduce the space of term-document matrix; Villalon & Calvo's method (Villalon & Calvo, 2009) is adopted to reduce the dimension of the LSA space to 75% of the total number of sentences.

Based on the created semantic space, the similarity or distance between consecutive sentences is computed using a cosine measure, as defined below:

$$\cos(X, Y) = \frac{\vec{X} \,.\, \vec{Y}}{|X|.|Y|}$$

For each document revision, average sentence similarity is computed and the results are compared with those of the prior revision in order to determine if an improvement in document cohesion has taken place.

In order to compute the topic overlap discussed in Subsection 6.1.4, topics are first extracted from each document revision, using an approach based on the Lingo clustering algorithm developed by Osinski et al. (2004), with particular attention to extracting frequent phrases from each revision. The assumption and definition of the term "frequent phrase" is discussed in detail in Osinski et al. (2004). Next, using the reduced term-document matrix calculated for LSA mentioned above, any existing latent structure of diverse topics is discovered for each and every revision. The detail of Lingo algorithm is described in Subsection 3.1.2. After discovering the revision topics, they are compared between two consecutive revisions of the same document in order to calculate the topic overlap between the two revisions, by using the topic matching rate as described in the previous subsection for computation of topic overlap.

The proposed heuristic are applied by using the obtained types of text edits and the results of LSA cohesion and topic overlap calculated as above. In conjunction with timestamp and user identification information obtained from the revision history, an event log is created of the writing activities for each document.

6.3 Heuristic Validation

This section addresses validation of the CWA Heuristic by deriving the writing activities of a test set and comparing them to a gold standard of human expert tagging, beginning with a description of the dataset.

For validating the heuristic for detecting writing activities, dataset A was used, as described and explained in Section 5.5. In this section, 15 documents were selected randomly from the 26 existing documents for the purpose of conducting an evaluation with regard to extracting writing activities.

The heuristic was validated against 15 documents which contained a total of 1407 major revisions. The documents were chosen at random from our dataset (described in Subsection 5.3.1). All major revisions were downloaded from Google Docs.

Manual tagging: In each major document revision, every text change was manually tagged as either a drafting, revising editing activity as defined in Lowry et al. (2003). A total of 2335 writing activities were tagged. It is noted that since revisions may contain more than one edit, there can be more than one writing activity

for each revision (see Figure A-2 in Appendix A for an example of multiple text edits performed on one revision).

CWA heuristic tagging: After pre-processing the major revisions, the heuristic was applied to each in order to determine which of the three core writing activities (i.e. drafting, revising and editing) were involved. Since brainstorming and outlining activities occurred very rarely and mostly at the beginning of the writing process, the initial concentration centred around evaluating the detection of these three activities. An evaluation of the heuristic for identifying brainstorming and outlining activities using all revisions of five documents is described later in Subsection 6.3.6. The details of the heuristic and the pre-processing steps are described above in Section 6.1 and 6.2, respectively.

6.3.1 Matrices

For each document, a comparison of the writing activities derived from the heuristic against those found by manual tagging was achieved by computed precision, recall and F1 scores according to the following formula adapted from Olson & Delen (2008):

Precision: Of all the activities (i.e. either drafting, revising, editing, or all) discovered by the heuristic, how many are correct?

$$P = \frac{|\{Correct\ activities\}|}{|\{Correct\ and\ Incorrect\ activities|}$$

Recall: Of all the activities manually tagged, how many are discovered by the heuristic?

$$R = \frac{|\{Correct\ activities\}|}{|\{Tagged\ activities\}|}$$

F1 score:

$$R = \frac{2 * Precision * Recall}{Precision + Recall}$$

For each individual revision, it is important to note that the heuristic identification of all text changes is 100% correct for every individual revision. Each text change detected by the heuristic is designated as one of the five writing activities (i.e. brainstorming, outlining, drafting, revising, and editing). The number of activities identified by the heuristic is always equal to the number of activities selected by manual tags.

The baseline is established from the human tagging. For each writing activity (i.e. drafting, revising, and editing), the baseline is equal to the number of activities divided by the total number of changed revisions (i.e. total number of revisions – the number of pauses or no-change revisions). In other words, the baseline is the result of Zero R classifier using the human ratings as our targets.

6.3.2 Applying Evaluation Matrices to CWA Heuristic

The task of evaluating the classification performance of the heuristic is concerned with computing precision, recall, and F1 scores. Since there are several ways to measure the evaluation matrices, this subsection illustrate this problem. Table 6-3 depicts an example of four revisions (R1, R2, R3, and R4) of a hypothetical document, in which there is one activity for R1, two activities for R2, three activities for R3, and five activities for R4 (11 revisions in total). Each activity was tagged manually as either drafting (D), revisiting (R), or editing (E). In addition, each activity was classified by the heuristic as belonging to one of these three categories.

Table 6-3. An example of four revisions of a hypothetical document.

Revisions	Human Tagging	Heuristic Tagging
R1	D	D
R2	D	D
	E	R
R3	R	D
	E	E
	D	R
R4	D	D
	D	R
	R	R
	E	D
	D	E

There is a problem. It is difficult to compute the evaluation matrices. For R1, it is straight forward because there is only one activity and heuristic tagging is matched with human tagging. For R2, heuristic tagging and human tagging have different types of activities: D and R for heuristic tagging, and D and E for human tagging. One of them (i.e. D) is correctly matched. However, with the second activity,

heuristic tagging mistakenly classifies E as R. For R3, heuristic tagging derives all three different types of activities: D, E, and R, which are the same as human tagging. However, only one (i.e. E) is correctly matched with human tagging. The other two are swapped. For R4, heuristic tagging also derives three different types of activities: D, E, and R similar to human tagging. However, the numbers of individual activities are different from those of human tagging. D has three in human tagging, whereas it has two in heuristic tagging, in which only one is correctly matched. R has two in heuristic tagging, but it has only one in human tagging, which is correctly matched. E has one for both heuristic and human tagging, but it is not matched.

Therefore, for each document, there are two aspects of performance evaluation. First, the precision, recall and F1 scores are computed using all revisions (11 activities in total). Another method of measuring is to calculate the scores using each revision and then average out for all revisions to find the performance measure. These are discussed for the hypothetical example below.

First, the performance measure is calculated for the revising (R) and drafting (D) activities. The performance measure can be computed in a similar way for the editing (E) activity.

6.3.2.1 Precision and Recall for Detecting Revising (R)

- Using all revisions (11 activities in total)

	Act	Not Act
+ Predicted	1	3
- Predicted	1	6

Precision for R = 1/4 and Recall for R = ½

- Using revision R1 (1 activity)

	Act	Not Act
+ Predicted	0	0
- Predicted	0	1

Precision for R = 0/0 = infinity and Recall for R = 0/0 = infinity (I consider as 1)

- Using revision R2 (2 activities)

	Act	Not Act
+ Predicted	0	1
- Predicted	0	1

Precision for R = 0/1 = 0 and Recall for R = 0/0 = infinity (I consider as 1)

- Using revision R3 (3 activities)

	Act	Not Act
+ Predicted	0	1
- Predicted	1	1

Precision for R = 0/1 = 0 and Recall for R = 0/1 =0

- Using revision R4 (5 activities)

	Act	Not Act
+ Predicted	1	1
- Predicted	0	3

Precision for R = 1/2 and Recall for R = 1/1 = 1

6.3.2.2 Precision and Recall for Detecting Drafting (D)

- Using all revisions (11 activities in total)

	Act	Not Act
+ Predicted	3	2
- Predicted	3	3

Precision for D = 3/5 and Recall for D = 3/6=1/2

- Using revision R1 (1 activities)

	Act	Not Act
+ Predicted	1	0
- Predicted	0	0

Precision for D = 1/1 = 1 and Recall for D = 1/1 = 1

- Using revision R2 (2 activities)

	Act	Not Act
+ Predicted	1	0
- Predicted	0	1

Precision for D = 1/1 = 1 and Recall for D = 1/1 = 1

- Using revision R3 (3 activities)

	Act	Not Act
+ Predicted	0	1
- Predicted	1	1

Precision for D = 0/1 = 0 and Recall for D = 0/1 =0

- Using revision R4 (5 activities)

	Act	Not Act
+ Predicted	1	1
- Predicted	2	1

Precision for D = 1/2 and Recall for D = 1/3

Finally, the accuracy in detecting all three activities per revision is shown below:

Revision	Activity	% Frequency (Baseline) (Zero R Classifier)	Precision	Recall	F1 score
R1	D	100%	100%	100%	100%
	R	0	100%	100%	100%
	E	0	100%	100%	100%
R2	D	50%	100%	100%	100%
	R	0	0	100%	0
	E	50%	100%	0	0
R3	D	33%	0	0	- (considered as 0)
	R	33%	0	0	- (considered as 0)
	E	33%	100%	100%	100%
R4	D	60%	50%	33%	40%
	R	20%	50%	100%	67%
	E	20%	0	0	- (considered as 0)
Mean	D				49%
	R				50%
	E				58%

The accuracy across all revisions of this document is shown below:

Activities	Precision	Recall	F1 Score
D	60%	50%	54%
R	25%	50%	33%
E	50%	33%	39%

The above shows the method for computing performance scores with all revisions and per revision for only one document. If there are several documents, the performance measure can be calculated per document using similar method as per revision by calculating the scores using all activities in the document.

Therefore, there are three types of evaluation methods:

- Evaluating with all revisions.
- Evaluating per revision.
- Evaluating per document.

These types of evaluation methods are discussed in the following subsections.

6.3.3 Evaluating the Heuristic with All Revisions

Using the calculation technique described above, the F1 scores are first calculated globally for all activities of all revisions of the 15 documents and compared to the result of Zero R classifier as shown in Table 6-4. F1 scores of drafting, revising, and editing activities are 76.21%, 77.62%, and 62.50%, respectively, whereas the Zero R classifier scores are 47.06%, 43.62%, 9.31%, respectively. Therefore, for all three activities, heuristic performs better than the baseline.

Table 6-4. Evaluation using all revisions

	Drafting	Revising	Editing
Zero R classifier	76.21%	77.62%	62.50%
Heuristic	47.06%	43.62%	9.31%

6.3.4 Evaluating the Heuristic per Revision

As depicted in the example described above, for each revision, the accuracy of the heuristic used for detecting the three activities is also calculated as shown in Table 6-5. F1 scores for drafting, revising, and editing activities in this type of evaluation are 76.89%, 74.23%, and 67.70%, respectively. The scores from Zero R classifier are M=35.77%, STD=23.36% for drafting, M=26.08%, STD=25.42% for revising, and M=6.21%, STD=33.54% for editing. As a result, for all three activities, heuristic detect writing activities more accurately.

Table 6-5. Evaluation per revision.

	Drafting	Revising	Editing
Zero R classifier	35.77%	26.08%	6.21%
Heurstic	76.89%	74.23%	67.70%

6.3.5 Evaluating the Heuristic Per Document

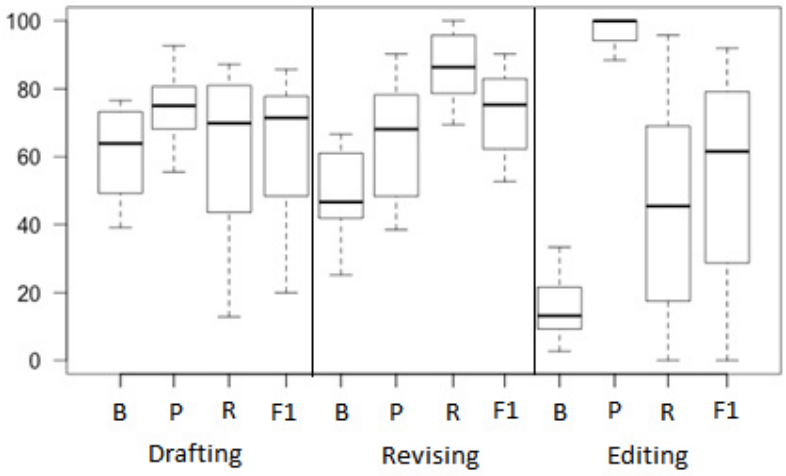


Figure 6-2. Precision (P), Recall (R), F1 score (F1) and Baseline (B) of detecting drafting, revising, and editing activities using the heuristic.

We also computed the F1 scores for each type of writing activities (i.e. drafting, revising, or editing) for all documents and compared them to the baseline. Figure 6-2 contains a summary of the results of precision, recall, F1 score, and baseline for detecting drafting, revising, and editing activities in 15 document revisions. Table 6-6 presents detailed results of the baseline and F1 scores of all 15 individual documents, which were ordered according to the number of their writing activities.

Overall, heuristic achieved higher F1 scores than the Zero R classifier for all documents in drafting and revising activities. As shown in Table 6-6, across 15 documents, we have M=72.56%, STD=10.59% for drafting and M=75.02%, STD=11.22%. The F1 scores vary considerably from 54.54% to 85.71% for drafting and from 45.45% to 90.19% for revising. One reason for this variance is that different groups of students produce different distributions of those writing activities as presented by the result of Zero R classifier (M=45.70%, STD=11.37% for drafting and M=43.67%, STD=8.35% for revising). Also, the F1 scores are quite low (less than 60%) for documents with fewer major revisions. For instance, Group 7 (with 40 writing activities in 14 major revisions) has an F1 score of 55.31% for drafting and only 45.45% for revising. Group 11 (with 65 writing activities in 28 major revisions) has an F1 score of 54.54% for drafting, and a very high F1 score of 78.05%.

Table 6-6. Heuristic performance based on Zero R as baseline and F1 score for detecting three activities: Drafting (D), Revising (R), and Editing (E) for major revisions of 15

documents. The documents were ordered according to the number of writing activities they contained.

Document	#Major revisions	Total Writing Activities	Zero R (Baseline)			F1 Score		
			D	R	E	D	R	E
15	15	39	48.57%	40.00%	11.43%	68.75%	75.68%	40.00%
7	14	40	38.46%	38.46%	23.08%	55.32%	45.45%	0.00%
1	46	60	53.45%	41.38%	5.17%	60.00%	65.57%	80.00%
11	28	65	25.42%	55.93%	18.64%	54.55%	78.05%	62.50%
2	39	91	46.51%	47.67%	5.81%	75.32%	78.16%	88.89%
17	42	101	36.08%	52.58%	11.34%	75.95%	90.20%	42.86%
25	67	107	37.76%	45.92%	16.33%	70.42%	84.31%	61.54%
26	79	107	51.43%	34.29%	14.29%	87.18%	80.00%	0.00%
24	40	109	39.18%	50.52%	10.31%	69.51%	82.40%	17.44%
18	27	116	42.73%	49.09%	8.18%	81.15%	72.75%	0.00%
10	143	178	75.69%	22.22%	2.08%	85.71%	60.98%	66.67%
4	140	184	52.08%	39.58%	8.33%	63.41%	68.42%	58.82%
21	147	185	49.66%	42.18%	8.16%	79.69%	83.92%	78.26%
3	148	235	50.52%	45.83%	3.65%	83.58%	82.56%	83.33%
19	432	718	38.02%	49.48%	12.50%	77.84%	76.92%	92.00%
Mean	93.8	155.67	43.67%	10.62%	75.16%	72.56%	75.02%	51.48%
STD	106.1	165.76	8.35%	5.74%	10.10%	10.59%	11.22%	33.17%
Total	1407	2335						

Although the overall performance of the heuristic is better than the baseline for editing activity as shown in Figure 6-2, for individual documents the performance on detecting editing does not perform well comparing to the other two activities, as shown Table 6-6. Across all 15 documents, the F1 score for editing is M=51.48% and STD=33.17%. The F1 editing score varies the most among the F1 scores for all three activities in all documents. Interestingly, the heuristic receive a score of 0 for three documents (7, 26, and 18). Similar to the effect of the total number of writing activities per document on the number of activities classified as drafting and revising, the number of editing activities per document also affects the performance of the heuristic on editing. The number of editing activities is relatively small compared to the number of other types of writing activities. The low performance of the heuristic with regard to editing can also be attributed to the fact that that the heuristic only

considers surface change operations as editing activities, so that other editing activities (such as grammatical corrections) are not detected.

6.3.6 Evaluating the Heuristic for All Five Writing Activities

In order to evaluate the effectiveness of heuristic for detecting all five writing activities, including brainstorming and outlining, all revisions were used (m'=13320) of six documents (Document 4, 10, 15, 19, 24, and 25) selected randomly from the 15 document sample. Unlike the previous performance evaluation which uses major revisions, each revision of these documents includes only one writing activity. The writing activities derived from the heuristic were compared against those identified by the manual tags along with computed precision, recall and F1 scores. Table 6-7 represents in detail the results of the baseline and F1 scores of the six individual documents, which were ordered according to the number of writing activities contained in each.

Table 6-7. Heuristic performance based on Zero R as baseline and F1 score for detecting 5 activities: Brainstorming (B), Outlining (O), Drafting (D), Revising (R), and Editing (E) for all revisions of 6 documents. The documents were ordered according to the number of writing activities contained in each.

Doc	Revisions	Zero R (Baseline)					F1 Score				
		B (%)	O (%)	D (%)	R (%)	E (%)	B (%)	O (%)	D (%)	R (%)	E(%)
15	1062	5.13	5.13	43.59	35.90	10.26	100.00	100.00	78.57	75.68	55.00
4	1311	0.00	2.70	50.68	38.51	8.11	100.00	40.00	73.41	68.87	58.82
10	1349	2.04	0.00	74.15	21.77	2.04	85.71	100.00	85.71	69.10	66.67
25	1397	0.00	2.97	36.63	44.55	15.84	100.00	80.00	82.46	76.31	59.54
24	1513	7.55	0.94	35.85	46.23	9.43	85.71	54.00	74.29	81.03	48.18
19	6688	1.03	0.00	37.63	48.97	12.37	66.67	100.00	78.31	76.92	92.00
Mean		2.62	1.96	46.42	39.32	9.68	89.68	79.00	78.79	74.65	63.37
STD		3.07	2.02	14.70	9.89	4.61	12.11	24.05	4.30	4.36	13.94

6.4 Summary

In conclusion, this chapter elaborates and evaluates a heuristic for automatically identifying collaborative writing activities. The CWA Heuristic is based on several features: the text structures, text edits, the number of words and phrases, cohesion and topic changes. In order to detect text edits, a text comparison utility is created which includes two text differencing algorithms for comparing the text content of two consecutive revisions and computing the edits at paragraph and word levels. An LSA technique is applied to compute cohesion and topic overlap is calculated through the use of a Lingo algorithm. The effectiveness of the CWA Heuristic was validated by obtaining writing activities from a test set consisting of revisions of real documents, and comparing the results to a gold standard consisting of human expert tagging. The results of the validation technique demonstrate that the CWA Heuristic performs reasonably well in identifying writing activities. The next chapter presents the use of these writing activities for extracting writing process models to use in process analysis.

CHAPTER 7
MINING WRITING PROCESSES

Based on the sequences of text edits and writing activities automatically identified using the technique explained in the previous chapter, writing processes can be analysed in order to extract the patterns of text edits and writing activities that are performed during the course of writing. The idea is to discover process models such as the ones shown in Figure 3-2 and Figure 3-4. Analysing the discovered process models can assist in understanding of how certain sequence patterns of writing activities (i.e. the steps followed by a group of authors) lead to high quality outcomes and sequence patterns that may lead to low quality outcomes. This thesis uses two techniques for this analysis: the process mining framework ProM (ProM, 2013) and one type of Markov models.

Process models can be automatically derived using the Heuristic Miner algorithm (Weijters & Ribeiro, 2010; Weijters et al., 2006) in ProM 6.2 (ProM, 2013) and Hidden Markov models (Hidden MM). The contribution of this chapter includes the

use of the Heuristic Miner algorithm to extract dependency diagrams of writing process models, as well as the use of Hidden MM (Rabiner, 1989) to extract transitional state diagrams of writing processes. In addition, based on several layers of semantics (i.e. text edits, writing activities and writing states), process models are created for analysis of writing processes.

Section 7.1 of this chapter describes writing process analysis performed by utilising a process mining framework, ProM, with several tools. Process models presenting dependency diagrams are extracted and analysed in a case study in subsection 7.1.2. Section 7.2 introduces two other types of process models based on transitional state diagrams. A case study demonstrating the capability of the two models for writing process analysis is included in subsection 7.2.3, and the chapter ends with an outline of a pilot study conducted to provide process models as mock-up visualisation to students.

7.1 Process Mining

This thesis employs a process mining technique to identify and explore the structure of writing processes. Process mining is one of the data-driven data analysis techniques that aims to discover underlying process patterns by extracting them from recorded event data, such as event logs captured by learning management systems or other learning software (Trčka et al., 2010; van der Aalst, 2011). Process mining techniques are used for a wide range of purposes, including: a) to discover new patterns; b) to check conformance of the observed processes to an *a priori* modelled pattern; and c) to extend *a priori* process models by using newly discovered patterns (Rozinat et al., 2007; Weijters & Ribeiro, 2010; Weijters et al., 2006). As the aim of this research is to identify patterns of text edits performed by students during collaborative writing tasks, a discovery technique is chosen to accomplish this goal.

7.1.1 Writing Process Discovery

7.1.1.1 Heuristic Mining

In this thesis, the heuristic miner algorithm is selected for extracting writing process models. The algorithm was implemented in the open source process mining framework ProM (Weijters & Ribeiro, 2010; Weijters et al., 2006). Heuristic Miner

has been developed for exploratory process mining of less structured data. This algorithm is appropriate for mining process data that require flexibility and cannot be strictly predefined in advance, and allows for the handling of data that contains various kinds of "noise," such as diversions from common sequences or incomplete traces of process information. Such noise is common in collaborative writing data, particularly the data derived from fine-grained text edits made by students as they write.

The heuristic mining process is based on relationship items known as *dependencies* between events (i.e., modelling actions). This frequency metric represents a certainty that there is a dependency relation between two events (i.e., an event is causally related to or dependent upon the events that precede it). This work uses the nominated *all-activities-connected heuristic* with three threshold parameters: a) the *Dependency threshold*, b) the *Positive observations threshold*, and c) the *Relative to best threshold*. (The parameters are set to the default values of "ProM".) Using these thresholds, the accepted dependency relations between events are: a) a dependency measure above the value of the Dependency threshold, and b) of a frequency higher than the value of the Positive observations threshold, and c) a dependency measure for which the difference with the "best" dependency measure is lower than the value of the Relative to best threshold. This configuration allows all distinctive events into the pattern and detects all possible causal dependencies between them. Using this algorithm, one ingoing and outgoing connection with the highest dependency value is identified and included into the model in each mining step. This process is repeated until all activities are connected. As a result, the final heuristic model does not necessary represent all possible links and dependencies between all activities, but it does depict the most strongly dependent actions. Details about this mining algorithm and how the above discussed parameters are calculated are found in Chapter 3 and in the research of (Rozinat et al., 2007; Weijters & Ribeiro, 2010; Weijters et al., 2006). Dependency graphs, i.e. the output of Heuristic Miner, are explained in Chapter 3.

7.1.1.2 Pre-processing steps

In order to discover a process model (i.e. a dependency diagram) for each document written by a group of students, pre-processing is required to create event logs for a

process mining algorithm. First, a process instance (or process case) is created for each document, using all of its revisions and its revision history. After all the text edits of all revisions are found, writing activities are identified using the heuristic described in the previous chapter. In the context of process mining (van der Aalst, 2011), each writing activity has two *transaction* types: *start* and *complete*. Each transaction has a timestamp associated with it. Timestamps belonging to the *start* and *complete* transactions of all writing activities are computed on the basis of the revision history. In this thesis, a complete transaction of a writing activity is the timestamp of its corresponding revision; however, a timestamp of a revision recorded on the revision history indicates the time when a document is automatically or manually saved, and thus produces that revision. Therefore, the time at which the author actually starts working on the document is not known. In this work, a start transaction's timestamp in a revision is a timestamp of its previous revision. The start transaction timestamp of the first writing activity (of the first revision) is the same as the transaction timestamp of the complete transaction, which is the first revision timestamp. Each process instance in a document then consists of a trace of writing activities which is ordered by their complete transaction timestamps. Process instances of all documents are transformed into an event log with the standard format of XES (eXtensible Event Stream) (Gunther, 2009). Subsection 3.2.1 provides description of event logs used in the context of process mining.

Based on the event log that is created, process analysis can be executed by using a process mining framework, ProM. Use of the Heuristic Miner algorithm extracts a process model for every document. A process model presenting a dependency diagram is then employed to discover the patterns of the authors' writing activities that are formed throughout the course of the writing. Examining the causal dependency between two activities and their frequency provides insight into the manner in which individual writing activities are carried out during the writing process, as exemplified in the following subsection.

7.1.2 Case Study

7.1.2.1 Dataset

This case study uses the same dataset (Dataset A) as the previous chapter, which is taken from a case study conducted in an engineering course of E-business Analysis and Design (ELEC3610) at the University of Sydney in 2010. As a reminder, Dataset A consists of a total of 26 student groups with two students forming each group. All the groups are asked to write a project proposal in 1,500 to 2,000 words. Details of the course and the dataset, including the structure of the documents, are given in Subsection 4.5.1. The total number of text edits and pauses performed by all 26 groups are shown (in ascending order of the groups' final marks) in Figure 7-1. The final marks are shown in Table 5-1. It is important to note that the "Pause" activity refers to an event in which students made no change to the texts.

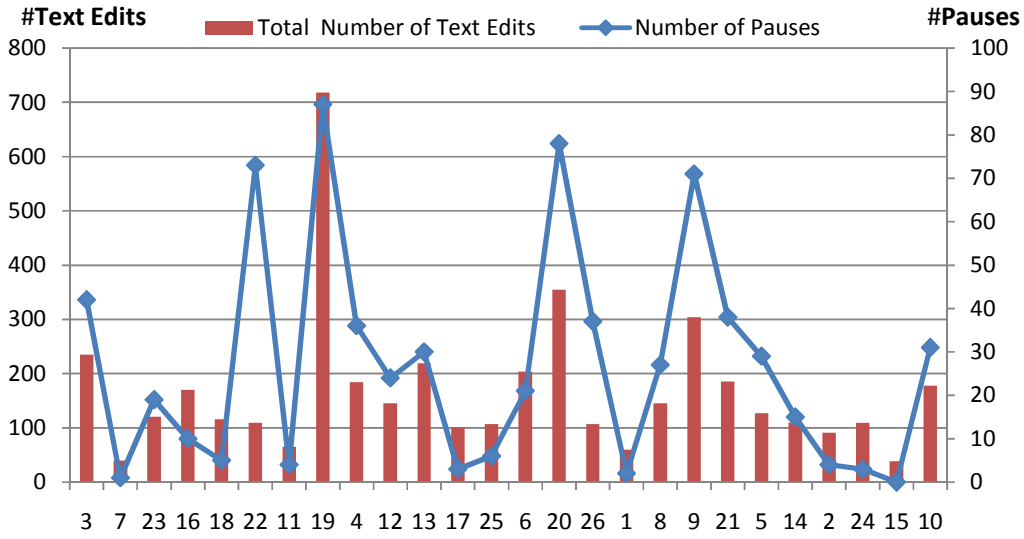


Figure 7-1. Final marks (in green), total number of inactivities (pause) (in blue) and writing activities (in red) of 26 groups in order of their final marks, lowest mark on the left.

From the information in Figure 7-1, it is difficult to distinguish between high-achieving and low-achieving groups based on the number of writing activities and pauses. The highest achieving group (Group 3) produced 235 writing activities. The lowest achieving group (Group 10) also performed numerous activities (178), whereas the second highest group (Group 7) completed only 40 activities. Because of the indeterminate nature of these results, it is necessary to extract process models and

perform process analysis of all groups in order to better understand how the author pairs developed their documents.

After deriving the writing activities in every revision of all the documents, a log file consisting of the sequences of writing activities, their timestamps and author identification is created by using the technique described in the previous subsection. Based on this log file, three types of analyses are carried out:

1. Extracting a snapshot of the overall writing activities of all groups in order to compare the start and end times of writing processes between different groups. This snapshot demonstrates when students actually start their writing tasks and how the activities are spread over time.
2. Discovering the process models of individual groups. The process models represent the sequence patterns of writing activities, which illustrates the manner in which the students undertake their writing tasks. For instance, we can determine whether the writing process of a particular group is linear (e.g. drafting is followed by revising which in turn is followed by editing) or interleaving (e.g. drafting and revising activities are interleaved most of the time).
3. Extracting information with regard to author collaboration and contribution. Irrespective of how writing activities are performed, the dynamics of collaboration over a working period remains a point of investigative interest, particularly the sequential patterns that occur throughout the writing process. Author contribution is another aspect for exploration, especially to reveal whether all authors in a group carry out all of the writing activities, or whether one of them dominates the tasks.

The result of the analyses is correlated to the final assessment of the documents. ProM 5.2 (ProM, 2010) is used to execute Dotted Chart Analysis (Song & van der Aalst, 2007), extract process models using Heuristic Miner algorithm (Weijters & Ribeiro, 2010), and conduct Performance Sequence Analysis (Bozkaya et al., 2009) and Organizational Mining (Song & van der Aalst, 2008).

7.1.2.2 Results

After pre-processing, the resulting event log records 3,720 events in total. Each process case represents one document. The average number of events per document

is 143, with a minimum of 39 events and a maximum of 631 events per document. This case study considers only three different types of events corresponding to three types of writing activities -- drafting, editing, and revising -- because brainstorming and outlining activities occurred very rarely and are done offline during group meetings before commencement of the writing tasks. Figure 7-2 depicts the number of individual writing activities for all groups. As in the analysis conducted previously of the total number of text edits and pauses, here again it is difficult to distinguish between high-achieving and low-achieving groups based on the number of writing activities produced (drafting, revising, and editing). Because of this result, dot chart analysis is implemented as described next, in order to obtain an overall snapshot of student writing activities.

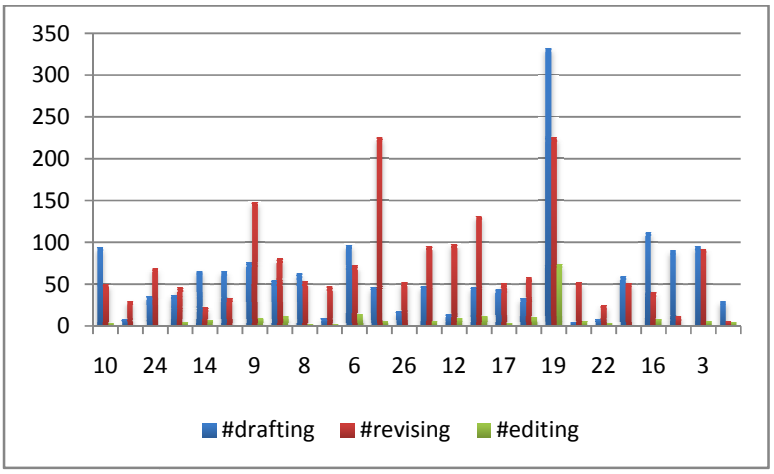


Figure 7-2. Numbers of drafting (blue), revising (red), and editing (green) activities performed by 26 groups (in order of their final marks, lowest mark on the left)

7.1.2.2.1 Dot Chart Analysis

The Dotted Chart Analysis utility of ProM (ProM, 2010) is used to examine student writing activities. Figure 7-3 illustrates the output of the dotted chart analysis of students writing their documents. All instances (one per document) are sorted by start time. In the figure, dots (points) represent writing activities occurring at a certain time. The three types of writing activities are represented by different shapes; circles are used to designate drafting, triangles for revising, and squares to indicate editing. The two authors in each group are differentiated by colour; black identifies author1, grey colour represents author2. All writers who begin the process (i.e. the first ones to perform a writing activity) are designated as author1 in each group.

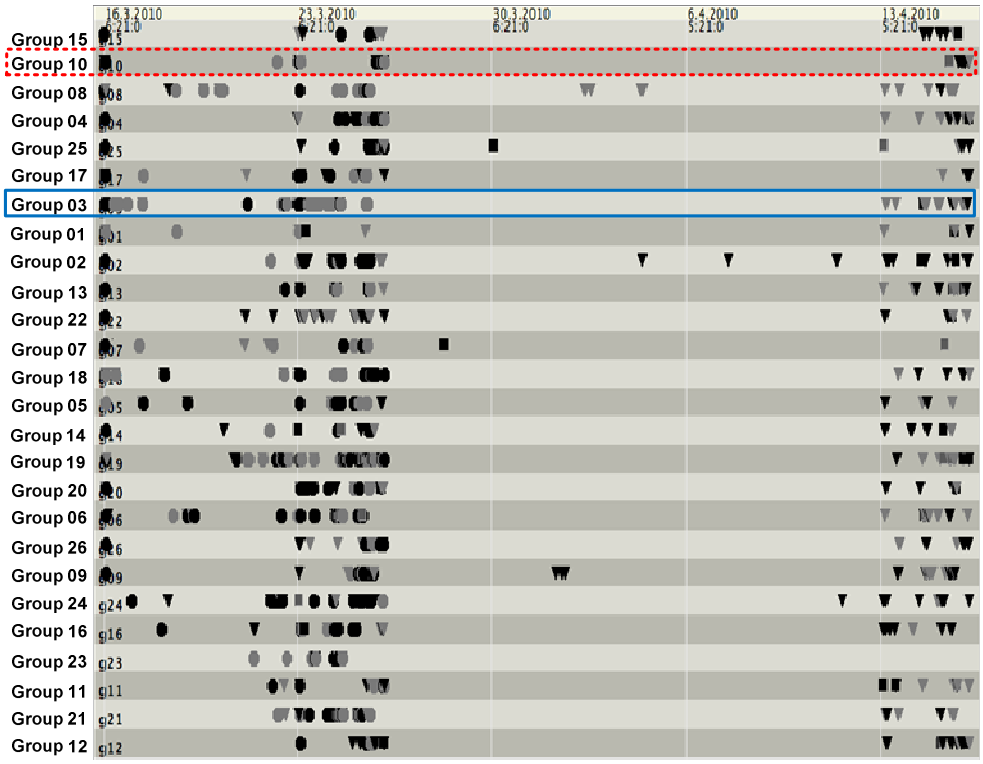


Figure 7-3. Dotted chart of 26 groups of students writing collaboratively (from ProM tool) displayed in order of starting time. Circles represent drafting, triangles depict revising, and squares denote editing. Author1 is identified by the colour black and Author2 is shown in grey.

Figure 7-3 shows most groups starting their writing tasks at approximately the same time, about ten days before the peer review submission due date 26th March 2010. Six groups begin their writing later than the others. Four groups receive above average final marks: Group 11 (75) Group 12 (68), Group 16 (85), and Group 23 (90). Two groups show below average final marks: Group 21 (61) and 24 (45). Interestingly, Group 12 begins the writing assignment three days before the due date for peer review submission. It is inferred, however, that students from these six groups these groups had actually begun the writing tasks earlier, using other word processing applications such as MS Word, because it is observed that they started writing on Google Docs with a substantial amount of text (containing sections and paragraphs) that is obviously pasted in.

Although no writing activities are expected to occur during the week of peer review while students wait for the resulting feedback, activities are recorded in Group 07 (91), Group 09 (61) and Group 25 (68) for this time period. Groups 07 and 25 only edit their texts, whereas Group 09 revises its text several times. An appraisal

of the Group 09 document revisions made during peer review week reveals that substantial text changes resulted from these activities. Furthermore, after receiving feedback from peer review (2nd April 2010), students begin revising and editing their documents before the final submission date (16th April 2010). It is observed that most of the groups revise their documents a few days before the final submission, except for Groups 02 and 08, who start working on their documents soon after receiving feedback.

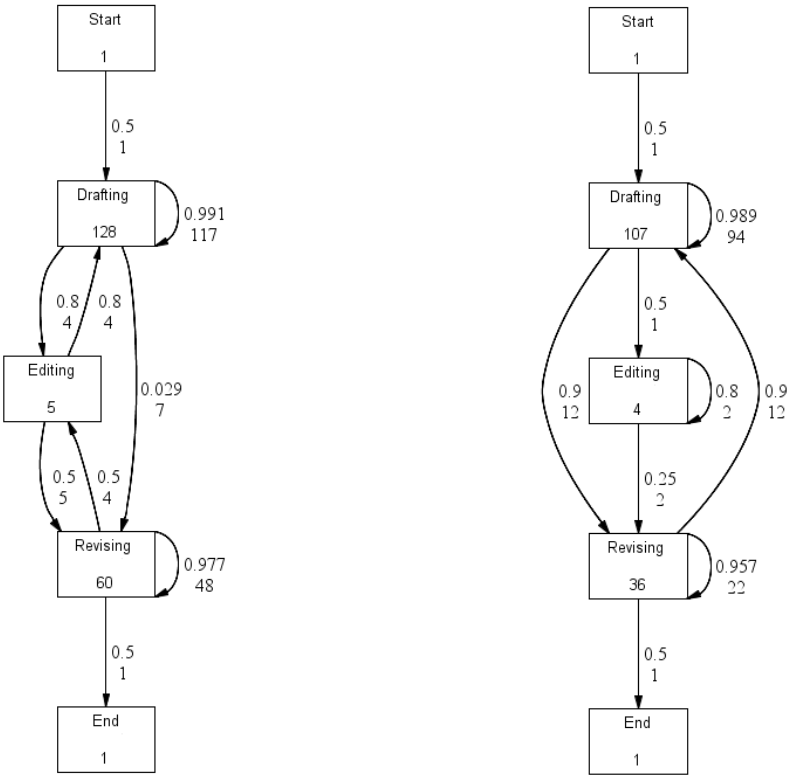
The dotted chart is further analysed in detail by undertaking an assessment of all writing activity events for each group. This assessment reveals that the writing process in most groups includes periods of time during which both authors write (perform text editions) synchronously. These events are identified by clusters of different colour dots in the chart. According to Figure 7-3, the periods of synchronous writing take place on the first day (i.e. 16 March 2010) and a few days before the deadlines (i.e. between 23 and 25 March 2010 relating to the peer review deadline, and between 14 and 16 April 2010 relating to the date of final submission). It is surmised that students collaborate frequently during these times to plan their writing assignments (on the first day) and to revise their documents (before the due dates).

The dot chart analysis provides an overview snapshot of the writing process of each group which makes it possible to compare the start time and end time of all the groups to ascertain when students actually begin their writing tasks and to see how the activities are spread over time. The chart also displays clusters of different colour dots, specifying synchronous writing periods that occur during the process. It is also of interest to know something about the writing path that students traverse in the course of their activities. These paths are discovered by using the process models as reported below.

7.1.2.2.2 Process Models of Students' Collaborative Writing

The process model of all 26 documents is for each individual group is generated by using the Heuristic Miner algorithm (Weijters et al., 2006) implemented in ProM 5.2. Figure 7-4 depicts dependency diagrams of two models: Group 03 (who received the highest final mark of 91/100) and Group 10 (who received the lowest final mark of 38/100). The numbers in the boxes indicate the frequency of the writing activities.

The decimal numbers along the arcs show the probabilities of transitions between two activities, and the natural numbers indicate the number of times this order of activities occurs among the three types, drafting (D), revising (R), and editing (E). Artificial start and end activities were added to each process instance (of a document) in the even log in order to indicate the start and end of the writing processing, respectively. Both groups started their writing with drafting activities and finish with revising activities, as shown in Figure 7-4. Figure 7-3 highlights timelines for writing processes of the two groups.



Highest Achieving Group (03) Lowest Achieving Group (10)

Figure 7-4. Process models of highest and lowest Achieving Groups (from ProM).

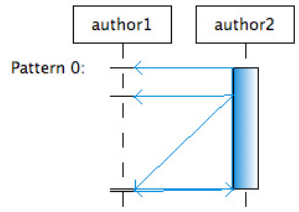
The process models shown in the figure demonstrate that Group 03 perform most of their drafting activities before revising their document, as evidenced by a one-way dependency from drafting to revising, although the dependency between the two activities is quite low (about 3%). Unlike Group 03, students in Group 10 commonly revise their document as they are drafting it. Their drafting and revising activities are interleaved most of the time as seen in the high dependency (approximately 91%) between those two activities. In Group 03, however, editing activities are carried out mostly during drafting and revising, evident from the high dependencies that exist

between editing and drafting (80%) and between editing and revising (50%); whereas in Group 10, editing takes place after drafting and before revising. In addition, it appears that whenever students in both groups start to draft or revise, they tend to continue drafting or revising, as shown by the high percentages of loops for both drafting and revising. In terms of editing activities, there is a loop for Group 10 which indicates that the group often started editing and continued to edit, as opposed to Group 3 who did not edit as often but interleaving with drafting and revising activities.

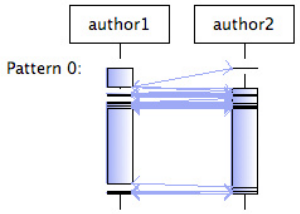
Appendix C contains the process models of all the groups. The fitness, which measures "the proportion of behaviour in the event log possible according to the model" (van der Aalst, 2011), is computed using the Heuristic Miner tool ProM 6.2. The fitness measure of a dependency diagram is described in Chapter 3. The fitness measures of the models shown in Appendix C range ranged from 0.64 to 0.99 (M=0.82, STD=0.08). Although process models are extracted and analysed in order to shed light on the manner in which each group performs their writing activities, the models do not show who (i.e. which group members) actually carry out the activities during these processes. The following subsection further explores how students collaborate and contribute during their writing tasks.

7.1.2.2.3 Performance Sequence Analysis

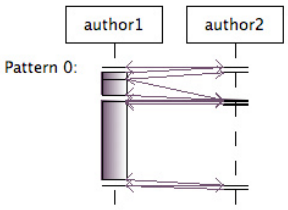
Performance Sequence Analysis (PSA), which is a plug-in to ProM, is used to study how each group collaborates during the writing process. The sequence pattern of user interaction in the event log (Bozkaya et al., 2009) is first discovered and the collaboration patterns are extracted for all groups. Figure 7-5 illustrates sequential diagrams of collaborative writing activities for four groups: Group 03, 10, 19 (final mark 74), and Group 22 (final mark 76). For the sake of confidentiality and simplicity, authors in each group are named author1 and author2. The author who starts the first writing activity is designated as author1, so that author1 and author2 represent different authors for different groups.



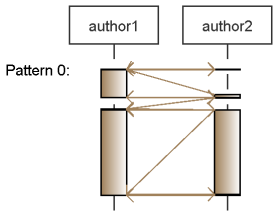
Highest Achieving Group (03)



High Achieving Group (19)



High Achieving Group (22)



Lowest Achieving Group (10)

Figure 7-5. Sequence patterns of 4 groups (clockwise from top left 03, 19, 10, and 22) of students writing collaboratively (from ProM).

An examination of the sequence patterns of all 26 groups reveals that in most groups, one author dominates the writing process. Only four groups display an approximately equal contribution of writing activities from both authors: Group 16, 18, 19, and 25. These groups are above average in terms of their final marks (85, 78, 74, and 6.8, respectively). One of them (Group 19) is depicted in Figure 7-5. All other groups except these four exhibit sequences of collaborative writing predominantly authored by one of the writes. The difference in contributions ranged from *single-author domination,* such as in Groups 03 and 22 (who obtained high marks) to almost equal contribution, such as in Group 10 (who received the lowest mark). In addition to this evaluation of the distribution contribution among authors, the scope of each author's involvement across the various writing activities (i.e. drafting, revising, and editing) is also appraised by applying the next process mining technique.

7.1.2.2.4 Organizational Mining

Organizational Mining provided by ProM (Song & van der Aalst, 2008) is used to ascertain which activities each author performs during the writing process. The writing activities of individual authors in four groups (Group 01, 03, 19, and 22) are depicted in Figure 7-6. As in the previous subsection, author1 and author2 represent different authors for different groups.

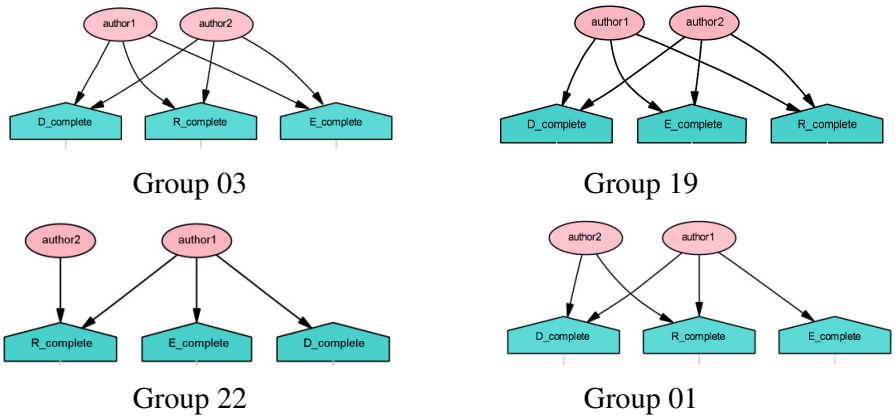


Figure 7-6. Author collaboration based on writing activities.

Use of the Organizational Mining tool contributes further knowledge with regard to who did what during the writing processes. As expected, both authors in the four groups that are characterised by equal contribution (Group 16, 18, 19, and 25 discussed above) carry out all three types of writing activities (for example, Group 19 in Figure 7-6 shows that the two authors contributed to all writing activities). Single-author dominating groups, however, are quite interesting. Although Figure 7-5 above shows one author dominating the writing processes in Groups 03 and 22, it is not clear whether the least dominating authors (author1 in Group 03 and author 2 in Group 22) perform all three categories of writing activities. Figure 7-6 clearly displays that *author1* in Group 03 conducts all three types of writing activities, even though the contributions are fewer than those made by the dominating *author2*, as shown in Figure 7-5. In Group 22, the situation is different. The dominating *author1* performs all three types of writing activities, whereas *author2* only revises the document once in a while.

In most groups, authors carry out the same writing activities, as exemplified in Group 03 and 19, for instance. Of the seven groups where both authors do not have equal writing activities, two have high final marks (Groups 7 and 22) and five have final marks below average (Groups 1, 2, 8, 15, and 24). In these groups, one author performs either only one writing activity (for instance, Group 22) or two writing activities (for example, Group 01) as depicted in Figure 7-6.

7.2 Hidden Markov Models and Heuristic Markov Models

In the previous section, application of the Heuristic Miner algorithm discovers process models based on traces of writing activities which present a high semantic level of text edits. Another type of process model -- the Hidden Markov Model -- which is an extension of Markov Models (Markov processes) depicts a transitional state diagram consisting of a set of states and transition probabilities. Based on the probability of state transition shown in the models, patterns of the writing activities undertaken by the authors during the process are extracted. The theoretical background of Hidden MMs is described in Chapter 3.

This section is organised as follows. First, the approach used in this thesis to discover two models -- the Heuristic Markov Model and the Hidden Markov Model -- of the collaborative writing process is presented. Second, the procedure in constructing these two models is explained in the pre-processing steps. Third, the last subsection illustrates the thesis approach with a case study in extracting the two models of process writing of real documents written by groups of students.

7.2.1 Extracting Heuristic Markov Model and Hidden Markov Model

Using transitional state diagrams as process models, process analysis can be carried out to discover patterns of student writing behaviour. There are two different ways to extract transitional state diagrams for writing processes. For each text edit, one corresponding writing activity is automatically identified using a heuristic. Using these identified writing activities as states, a Markov model is easily extracted which presents process models of writing activities. On the other hand, a writing state, though it cannot be directly observed, can be represented by one or more text edits, which are semantically grouped on the basis of larger behaviour patterns or strategies performed by authors during writing. In this case, Hidden MM (Rabiner, 1989) are good candidates for discovering writing behaviour patterns because they allow identification of author writing behaviour patterns from sequences of text edits made during their writing tasks. Based on a sequence of text edits which are measured and observed, the Hidden MM extracts writing states, which cannot otherwise be directly measured as well as the transitioning probabilities among these states.

Both writing activities and text edits are used as inputs to the Heuristic MM and Hidden MM generating algorithm, called HMM constructor, to create two models of writing processes, which are subsequently compared.

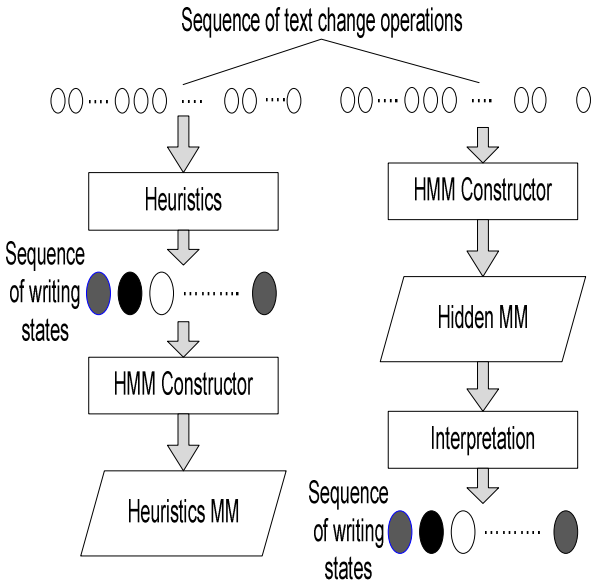


Figure 7-7. HMM model created with semantic heuristic on the left (Heuristic MM) and without the heuristic on the right (Hidden MM).

The first model, which we call a *Heuristic MM*, is depicted on the left of Figure 7-7. A Heuristic MM is a Markov model created from a sequence of writing activities, derived from text edits by applying the semantic heuristic explained in Chapter 6. The full construction consists of the following: A sequence of text edits made in each revision forms the input to our heuristic set. The result is a corresponding sequence of writing activities, which become the states of the Markov model. Using the HMM constructor described in Subsection 7.2.2 below, a Heuristic MM of the collaborative writing process for the corresponding document is discovered from the input sequence of writing activities. Figure 7-7 depicts the process of extracting the Heuristic MM.

The second model, which we call a *Hidden MM*, is depicted on the right hand part of Figure 7-7. Unlike a Heuristic MM, a Hidden MM is built directly from the sequence of text edits. A Hidden MM is a model with writing states that are unobserved. Using the sequence of text edits, the HMM constructor discovers the structure of Hidden MM (i.e. a set of states and the output probability associated with each state) and other parameters (e.g. transition probabilities from one state to others

or itself. The Hidden MM is then analysed to identify and interpret writing states to discover sequences of writing patterns. The process of extracting the Hidden MM is shown in Figure 7-7.

The difference between the two models is that in the first case (Heuristic MM), writing states are derived prior to constructing the model; whereas in the second case (Hidden MM), the model is built first, and the writing states are derived afterwards.

7.2.2 Pre-processing

Pre-processing steps need to be performed in order to use revisions and revision histories for generating the two models, Hidden MM and Heuristic MM. The pre-processing involves two main steps: (i) identifying the text edits and, for the Heuristic MM, the corresponding writing activity that produces each revision; (ii) creating sequences of text edits and, for the Heuristic MM, writing activities. These steps are detailed below.

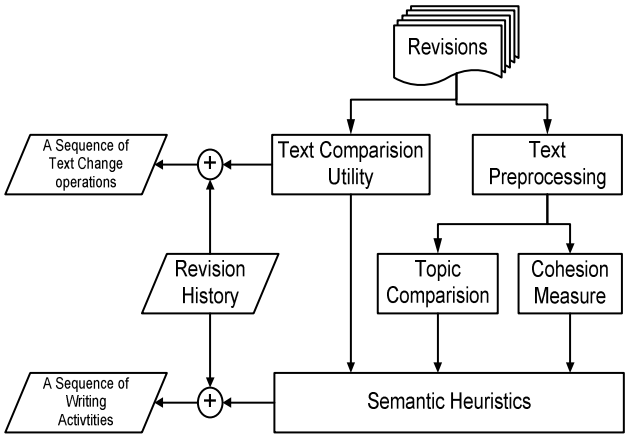


Figure 7-8. Pre-processing steps.

A method outlined in Chapter 6 is used to extract semantics of text changes during the writing process. Illustrated in Figure 7-8 above, this method proceeds as follows:

Two sets of data are accessible for each document; the first one is the revision history, which contains timestamps and author IDs for all document revisions, and the second one contains the text of each revision. A text edit type is identified for every revision by applying the text comparison utility described in Chapter 6 to compare the revision to its former version. The comparison results in a sequence of text edits for that document. In parallel, text mining pre-processing techniques are

used to perform the tokenisation, stemming, and stop word removal for the text in all revision. A topic change and a cohesion measure are calculated as well, using the Lingo algorithm (Osinski et al., 2004) and LSA, respectively. Using the heuristic proposed in Chapter 6, we then associate a writing activity with each revision, obtaining a sequence of writing activities for each document.

In this case study, a "Pause" activity (corresponding to "p" type of text edits) is added to represent an event where authors make no change to the text. This (in)activity indicates a *pause* time in the writing, possibly because authors stop to think or reflect before starting to write again, or the writer conducts further research related to the writing, or any other related reason. The pauses in the writing and the time taken to complete an activity have a potential impact on the interpretation of the process. An activity sequence can include many consecutive long pauses. In this process mining, the accumulated pauses are designated as a delay (wait) time of activities or events.

Inactive events (*Pause* activities and *p* text change operations) represent pauses in the writing process. In the Heuristic MM, these are replaced with the previous writing activity. For example, a sequence composed of one Drafting activity followed by three pauses and one Revising activity becomes four Drafting activities followed by one Revising activity. In the Hidden MM, pause events are replaced with the previous text edit. For example, adding information, followed by one pause and one reorganisation of information, becomes two adding information events followed by one reorganisation of information.

The numbers of these inactive events are also used in calculating stationary probabilities (Jeong et al., 2010) to investigate whether the proportion of time that students spend in each of the writing activities has any importance. The notion of stationary probability is used as the relative proportion of activities that belongs to a certain state. In other words, the stationary probability of a state A is the proportion of occurrences of state A among all the states that occur in a sequence of length n iterations generated by the model; n is normally the average number of activities in the input sequences.

Each generated text change and activity sequence can be used to derive a Hidden MM and Heuristic MM for a document, respectively. The HMM constructor uses the algorithm developed by Li and Biswas (2002) to build Hidden MM and Heuristic

MM. The algorithm has to estimate the number of states as explained in Subsection 3.3.2 for constructing Hidden MM, whereas it uses number of distinct activities in the activity sequence as the number of states for deriving the Heuristic MM.

7.2.3 Case Study

This case study uses the dataset A described in Section 5.4 and used in the process mining case study also. This section illustrates the techniques applied to extract student writing behaviour using both Heuristic MM and Hidden MM and reports the results obtained for two documents, one written by a high performing group and one by a low performing group (in terms of the group's final mark for the assignment). In order to compare patterns extracted from the Heuristic MM and the Hidden MM to those from the process mining algorithm, Heuristic Miner. The two documents are the same documents used in a case study in Subsection 7.1.2.

Table 7-1 lists the text edits used in the heuristic and their description (the complete heuristic is contained in the previous chapter). It is important to note that text edits at both the paragraph and the word levels are used to identify writing activities, thus the deriving Heuristic MM. For the hidden MM, text edits at both the paragraph and the word levels are also taken into consideration. However, all word edits on existing paragraphs -- such as inserting words (C8.1) and deleting words (C8.2) -- are designated as C8.

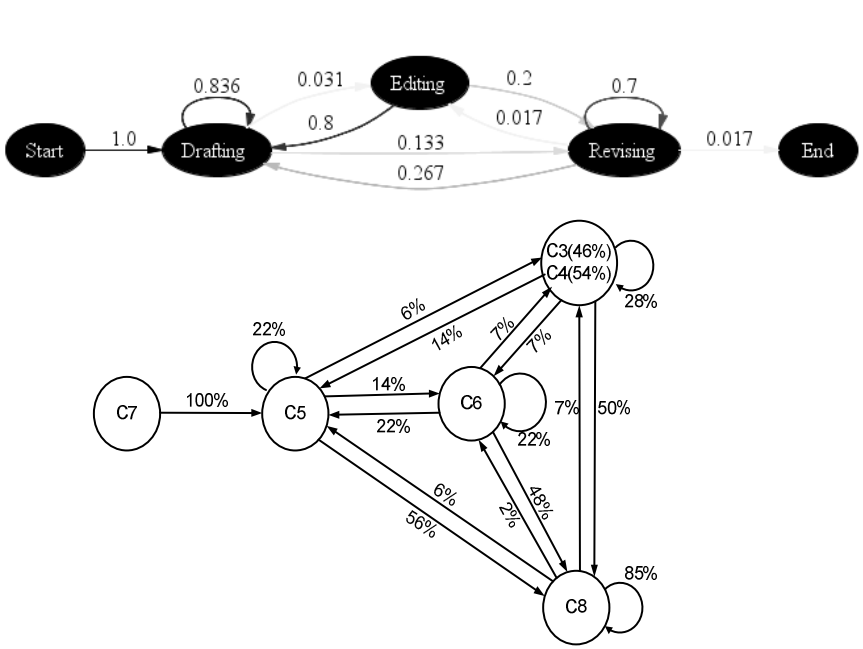
Table 7-1. Text edits and their description

Text Edits	
Code	***Description***
C1	Surface Change
C2	Reorganization of Information
C3	Consolidation of Information
C4	Distribution of Information
C5	Addition of Information
C6	Deletion of Information
C7	Alteration of Form (Macro-Structure change)
C8	Micro-Structure Change
p	No change

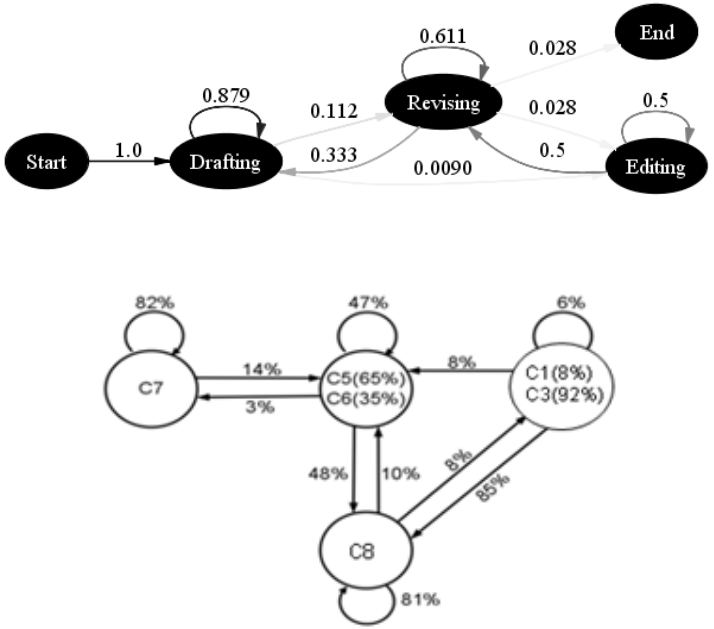
7.2.3.1 Constructing the HMMs

The generated text edit and activity sequences are used to derive two Markov models for each document (i.e. Heuristic MM and Hidden MM).

The writing process models for the documents written by High (H) and Low (L) performing groups created by using the HMM algorithm described in Subsection 7.2.2 are shown in Figure 7-9. In the figure, the models at the top, with black states, represent the Heuristic MM; and the ones at the bottom, with white states, represent the Hidden MM. The Hidden MM models are made up of a set of states, the text change edit patterns (the output probability) associated with each state, and the transition probabilities between states. For example, the model discovers that authors of document H in the C3(46%)C4(54%) state engaged in combining paragraphs 46% of the time and in distributing paragraphs 45% of the time. The transition probability associated with a link between two states indicates the likelihood of the authors transitioning from the current state to the indicated state. For instance, the Hidden MM model of H document predicts that in the C3(46%)C4(54%) state, after either consolidating or distributing text, the likelihood of authors adding new text is 14%; the likelihood of their deleting existing text is 7%; the likelihood of their changing text is 50% and the likelihood of their remaining in the same state (i.e., continuing to consolidate or distribute text) is 28%. Likelihoods of less than 2% are not represented in the figure, which explains why the probability numbers do not add up to exactly 100%. Similar to the Hidden MM, the Heuristic MM consists of a set of states and the transition probabilities. Since each state of the Heuristic MM represents an entire writing activity, there are no output probabilities associated with each state.



High Performing Group, Group 3



Low Performing Group, Group 10

Figure 7-9. MMs of the documents of High and Low Performing groups (Heuristic MM and Hidden MM respectively).

7.2.3.2 Analysis of MMs

These MMs are investigated to learn more about how students write their documents. The Heuristic MMs provide a good overview of the manner in which the authors develop their documents. The models reveal patterns of writing activities of the two groups similar to those highlighted by Heuristic Miner. For the high performing group, Group 03, the transitions between drafting and revising activities are quite low, whereas in the low performing group, Group 10, there is high transition (33%) from revising to drafting. In other words, students of Group 10 tend to start drafting their document again after revising it. Editing activities were more likely followed by drafting (80%) in Group 3, whereas the same activities were commonly followed by revising (50%) in Group 10. In addition, students of both groups tend to perform the same activities for a period of time as there were high percentages of loops for all three activities, except editing activities of Group 03. Editing activities were performed interleaving with drafting and revising activities in Group 3. The patterns described above have also been discovered by Heuristic Miner. Therefore, the Hidden MMs were used to gain deeper understanding in order to distinguish the high from the low performing group based on finer activities such as text change edits.

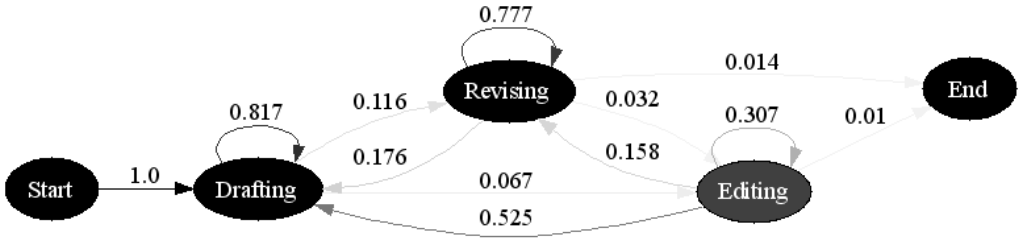
Further analysis of the Hidden MMs furnished more information related to the students' methods of editing and modifying their texts. In Figure 7-9, the model of the high performing group (group H) has five states, whereas the model of the low performing one (group L) has only four states. The alteration of form operation (C7) happens only once in group H. This is interpreted as reflecting that when group H students start drafting after completion of outlining or brainstorming, they begin drafting activities by adding new paragraphs (C5). After that, they are most likely to change existing paragraphs (C8) because the C5 state has a 56% likelihood of transitioning to the C8 state, compared to a 14%, 6% and 22% likelihood of transitioning to C6, C3/C4 states and itself, respectively; so the writers probably continue changing the existing paragraphs to improve the cohesion of the text. This is confirmed by the fact that the C8 state has the highest reiterating probability (85%). After changing the text to their satisfaction, the students are most likely to combine/distribute (C3/C4) the existing paragraphs to make the text more cohesive or to start describing new topics in the text by adding new paragraphs (C5), because the C8 state has a 7% and a 6% probability of transitioning to C3(46%)C4(54%) and

C5 states, respectively. After combining and dividing paragraphs in C3(46%)C4(54%) state, the students are likely to come back and change the existing paragraphs because this state has a 50% probability of transitioning to the C8 state. In addition, there is a strong relationship between C5 and C8 states (56% probability of transitioning) and C6 and C8 (48%). This indicates that the students are most likely to modify text after adding and deleting. These interpreted patterns accorded with common writing activities of the students in group H.

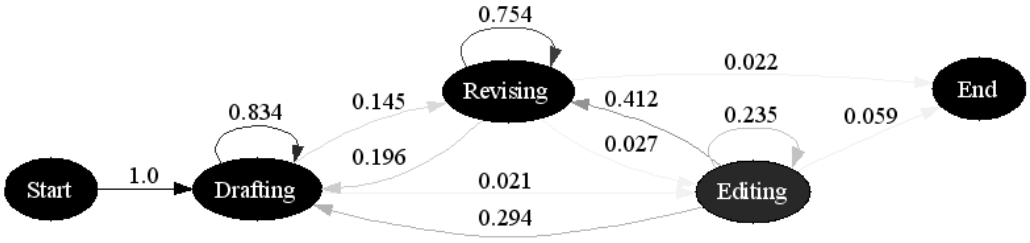
Comparing the Hidden MM of group L and group H reveals some similarities and some differences in the group L students. The state transition behaviours between the two models are quite similar, although the models have different structures, i.e. number of states. With regard to structure difference, the model of group L includes surface change or editing (C1) activities which never occur in the model of group H. There are also stronger differences that distinguish the two groups. First, there are more C7 activities in the writing process of L than in that of H. This suggests that the students of this group alter the whole text completely several times. This occurs particularly when the students change the topics of the text. In addition, there is no obvious transition from the C5(65%)C6(35%) state to the C1(8%)C3(92%), unlike in the model of group H, which has transitions from both C5 and C6 states to C3 state. Importantly, there is a very strong relationship between C1(8%)C3(92%) state and C8 state (85% transitioning probability) in group L. This indicates that after editing and formatting texts, students continue to change the text frequently.

The above analysis compares patterns of writing activities between the highest and the lowest performing groups, Group 3 and Group 10, respectively. After examining both Markov models (MMs) of all individual groups, we found that there are different patterns for different groups. In order to distinguish clearly the better from the weaker groups, aggregated MMs of the top and low groups are discovered and comparatively analysed. Figure 7-10 depicts aggregated Heuristic MMs of the high 8 groups (HS groups) and low 6 groups (LS groups). The HS groups are Group 3, 7, 11, 16, 18, 19, 22 and 23, which obtained final marks higher than 70%. The LS groups are Group 2, 5, 10, 15, and 24, which received final marks below 50%. The final marks are showed in Table 5-1.

From Figure 7-10, the models reveal patterns of writing activities of the aggregated groups similar to those of individual high and low performing groups. Particular, students of the two groups were likely to perform the same writing activities for a period of time because there were high percentages of loops for all three activities. This pattern was also highlighted for all activities of the high and low performing groups, Group 3 and 10, respectively, except for editing activities of Group3. In addition, editing activities were more likely followed by drafting (52%) in HS groups, whereas the same activities were commonly followed by revising (41%) in LS groups. Furthermore, after drafting their documents, both groups tend to conduct revising activities rather than editing activities. This is because the probabilities of the transition from drafting to revising were 12% and 14% for HS and LS groups, respectively, whereas the transition from drafting to editing were only 7% and 2% for HS and LS groups, respectively. Similar to individual high and low performing groups, students of both aggregated groups tend to start drafting their document again after revising it. However, the probability of transition from revising to drafting of the low performing group, Group 10 was high (33%), whereas the probabilities of both aggregated groups were similar and low (17-19%).



High Performing Groups



Low Performing Groups

Figure 7-10. Heuristic MMs of High Performing groups versus Low Performing groups.

7.2.3.3 Analysis of stationary probabilities

Inactive activities (Pause) are incorporated in the calculation of stationary probabilities. For example, if an activity A is followed by five consecutive Pause activities, we would designate this as six occurrences of activity A for this interval. The computed stationary probabilities are summarized in Table 7-2.

Table 7-2. Stationary Probabilities.

Document	C1	C3	C4	C5	C6	C7	C8
H	-	4	5	9	5	0	77
L	1	5	-	10	8	5	71

The table obviously reflects that the models of group H and L are almost identical. There is evidence that both groups spend a great deal of time changing paragraphs. The main difference in terms of the proportion of time between H and L groups is that the L group students spend 5% of their time changing topics, whereas the H group students define their topics early on. The models mentioned in the previous section make the same discovery.

7.3 Distilling Processes to Students and Instructors: A Pilot Study

A pilot study is conducted to assess four types of visualisations: a snapshot of writing processes (shown in Figure 7-11); transitional state diagrams of writing activities (shown in Figure 7-12); topic evolution and topic-based collaboration charts (shown in Figure 7-13); and bar charts depicting the number of revisions per writing activity performed by individual authors (shown in Figure 7-14). There were mock-ups created manually using a synthetic data. The first two types of the visualisations are based on a timeline and process model discussed in the previous sections. The aim of this pilot study is to gain a better understanding of what types of process models and visualisations student authors find helpful for their writing tasks and what kind of information they would like to obtain while engaged in a collaborative writing assignment. The result of this pilot study was used to derive the design of the visualisations proposed in the next chapter.

The four visualisations were first presented to individual students who voluntarily participate in the study. The students were enrolled in a postgraduate course at the Faculty of Education and Social Work, and the assignments in this course involve a substantial amount of writing. After being given the visualisations,

the students were interviewed individually by a course instructor. In the interview, students were asked to describe what information is conveyed to them by each type of visualisation; they are also asked for their suggestions on how to improve the design of the visualisations. In addition, comments from one instructor with regard to these types of visualisations are also included in the study. The following section begins with a description of the four types of mock-up visualisations, followed by a summary of the results drawn from the interview and the presentation of guidelines for designing visualisations of writing processes to use as feedback for students while they are engaged in collaborative writing activities.

7.3.1 Mockup Visualisations

Figure 7-11 is based on a timeline of the Dotted Chart Analysis tool of ProM (ProM, 2013). It depicts writing activities performed over time during the writing process. Each dot represents a writing activity: red, blue, and green for drafting, revising, and formatting, respectively. In this study, an editing activity is called a formatting activity, including surface and formatting changes. Figure 7-11 illustrates four groups with different writing processes. The figure shows that the four groups perform the same number of writing activities (i.e. 22 activities); but a relative comparison among the four groups indicates that the numbers of drafting and revising activities are equal for both Group Alpha and Group Beta, whereas Group Charlie and Group Delta perform more drafting activities than revising activities. The number of editing activities is equal for all four groups. The distribution of writing activities differs among the four groups. In particular, all drafting activities are performed before revising activities for Group Alpha and Group Charlie, whereas revising activities are produced interleaving with drafting activities for the other two groups. This study asks participants to compare how writing activities are performed by the four groups and tests whether they are able to convey the information described above.

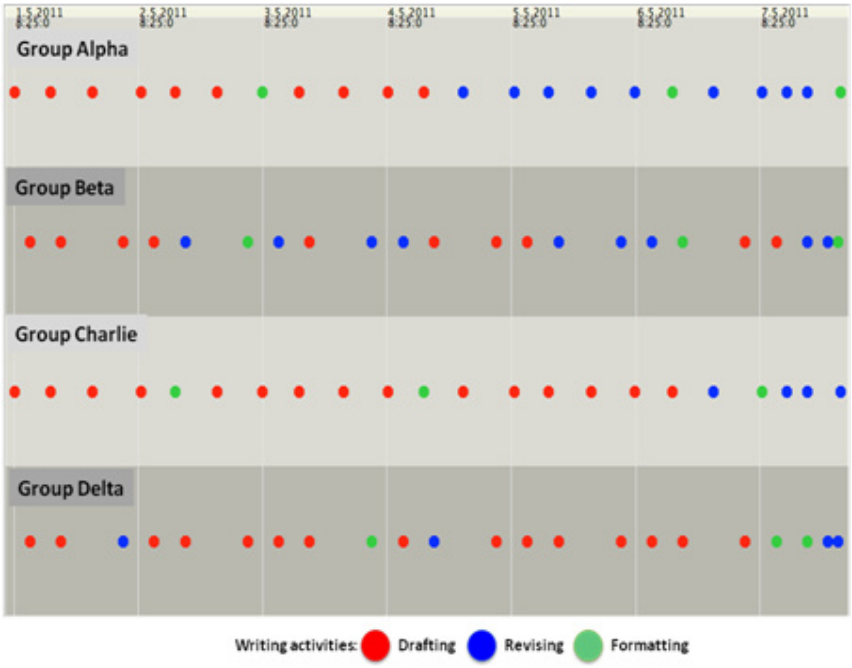


Figure 7-11. Mockup snapshot of writing processes, generated by Dot Chart Analysis plugin of ProM

Figure 7-12 is based on a transition diagram output from Markov models. Each state represents a writing activity. The size of each state is proportional to the total number of words performed for the corresponding writing activity over the period of the writing process. The figure depicts the transitions from individual writing activities to themselves and to other activities. For instance, the figure shows the interleaving of both drafting and revising activities of Group Beta and Group Delta. For Group Alpha and Charlie, revising activities sometimes occurs only after drafting activities. Because of the loops of drafting and revising activities, a drafting activity is often followed by another drafting activity and similarly a revising activity is often preceded by another revising activity. Although transition probabilities of the three writing activities are different for different groups, the amounts of drafting and revising are quite similar for Group Alpha and Group Beta. Group Charlie and Group Delta perform considerably more drafting than revising. The four groups all perform editing activities equally. Participants are tested to see if they are able to discern this information from Figure 7-12.

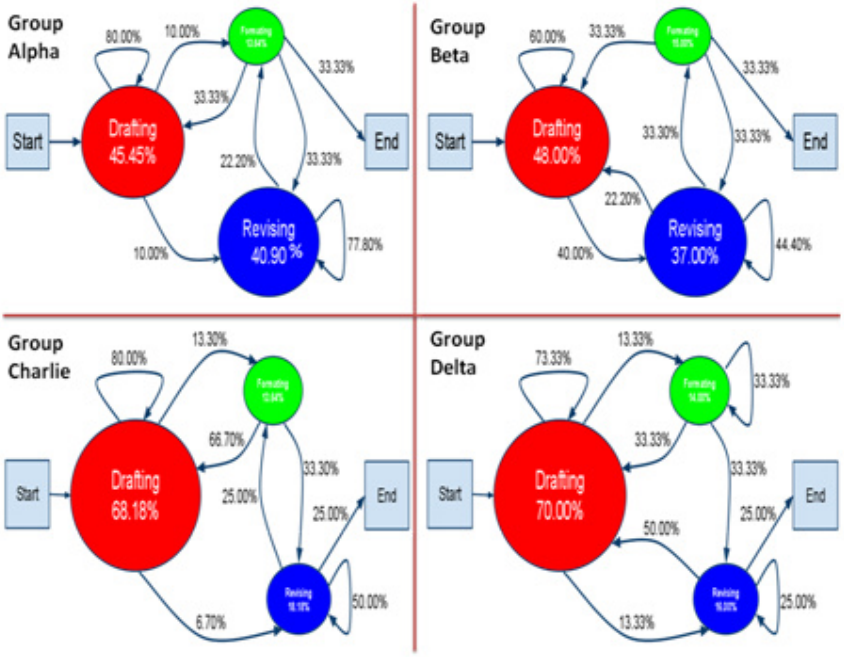


Figure 7-12. Mockup transition diagrams of writing activities based on hidden Markov models.

It is important to note that Figure 7-13 is created for viewing how topics evolve and how they are collaboratively edited. There are three topics and three authors in each group. A dot represents a writing activity through the use of a colour scheme as depicted in the previous two mock-up visualisations. The size of the dot shows the number of words produced by the corresponding writing activity. The topic-based collaboration of individual authors shown in this figure is quite different for each group. Every author in Group Alpha and Group Charlie writes about a different topic, whereas two authors write about the same topic in Group Beta and Group Delta. Participants are interviewed to see if they can understand the figure and gain insight into the topic-based collaboration of different groups.

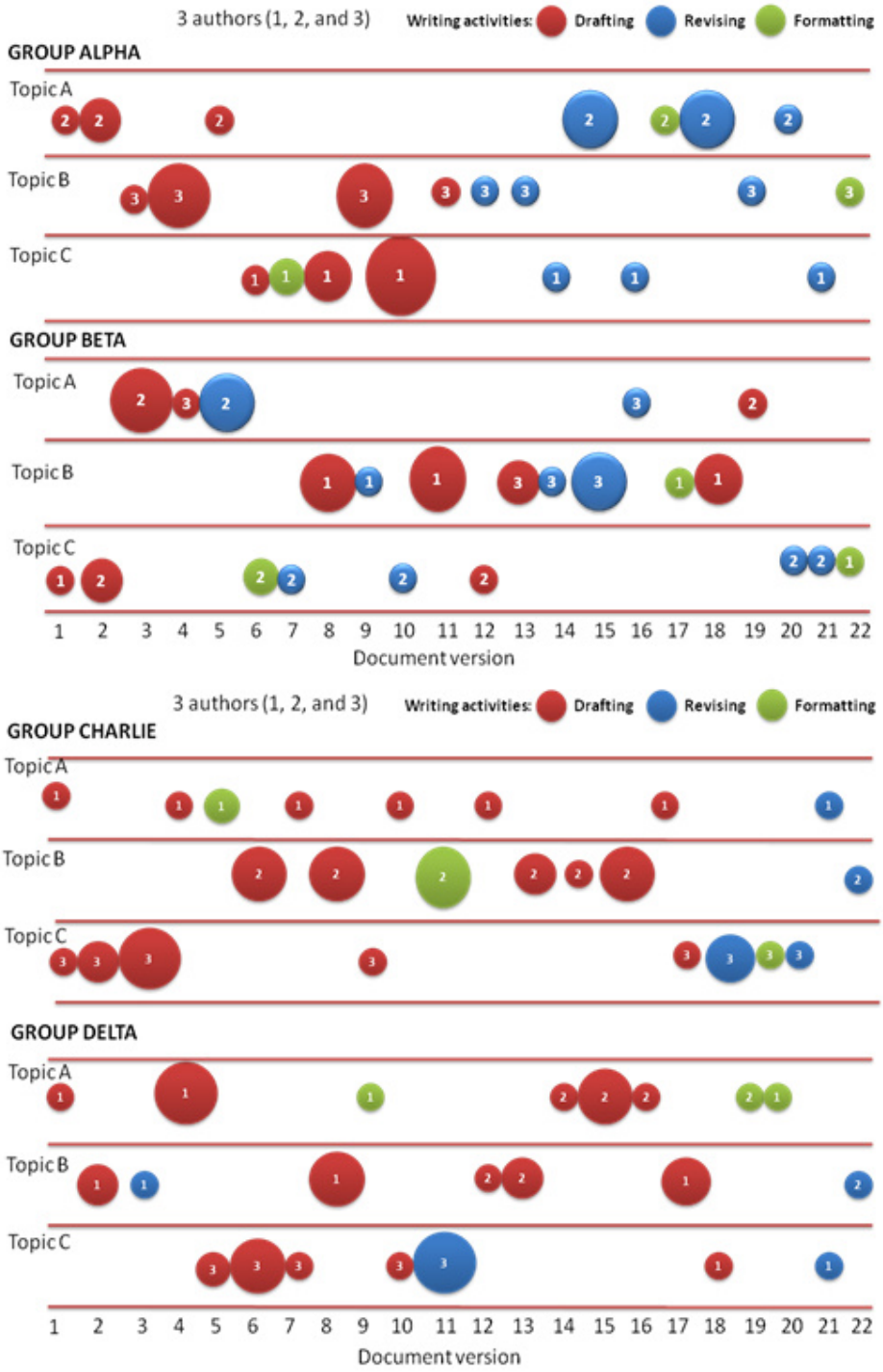


Figure 7-13. Mockup topic evolution and topic-based collaboration

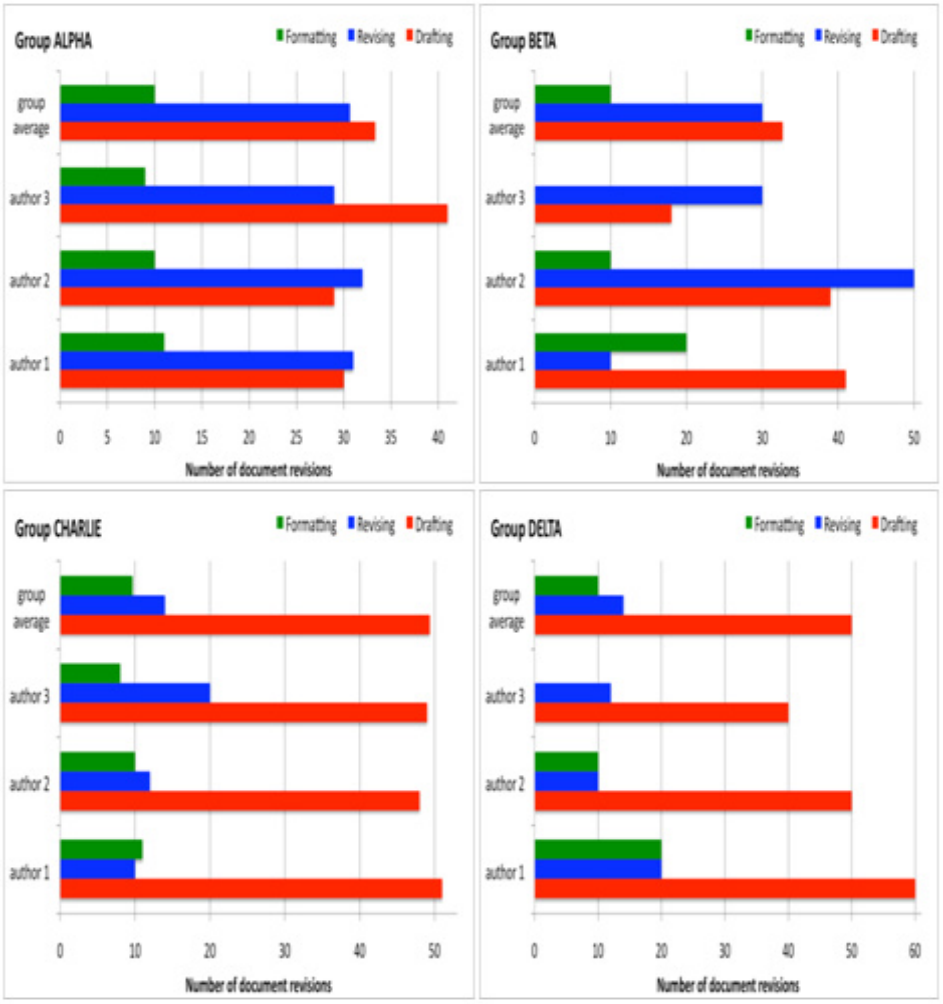


Figure 7-14. Mockup authors' contribution based on writing activities: formatting (i.e. editing) in green bar, revising in blue bar, and drafting in red bar.

Figure 7-14 depicts a mock-up number of revisions categorised by writing activities for the individual authors of four groups. Study participants are asked if they can differentiate the different distribution of writing activities of different groups. The figure shows that Group Alpha and Group Beta perform significantly more revising activities than Group Charlie and Group Delta. In addition, one author of both Group Beta and Group Delta does not carry out any editing activities at all, whereas all members of Group Alpha and Group Data complete all writing activities with different distribution of revisions. The interviews are discussed in the next subsection.

7.3.2 Feedback from Interviews

Eight students participated in the interview. In addition, one course instructor provides comments with regard to the four types of mock-up visualisations. The following sections contain a summary of the interview results.

7.3.2.1 Timeline of Writing Processes

All participants understand and gather information from Figure 7-11 without any problems at all in doing so. This suggests that students can perform process analysis using the timeline.

7.3.2.2 Transitional State Diagrams

Students had the most difficulty in making sense of Figure 7-12. They mostly focus on the size of the circles, which leads them to make inferences about the relative amount of different writing activities. Although this is useful, they can obtain this information from Figure 7-14 as well, so this does not reflect a unique value of this visualisation.

Some students could see differences in end points of the group writing. This is valuable because although this information can also be inferred from Figure 7-11, it appears more salient in Figure 7-12.

Only one student was able to use the arrows to make inferences about dependencies between writing activities. The source of confusion appears to relate to finding out which arrows (or more precisely, which figures in the arrows) to compare. Many students asked why the figures do not add to up 100%.

Because the unique value of Figure 7-12 lies in the information about the dependencies between writing activities, the instructor suggested a way to make this information more salient and intuitive by using different colours for arrows stemming out from each circle. This device might help people to think in terms of, e.g. "what activity most likely follows Drafting".

In addition, the most pedagogically relevant inference from Figure 7-12 involves the dependencies between different writing activities (especially whether there is an interplay between drafting and editing, which suggests the amount of revisiting or

rethinking of ideas). The instructor recommended an investigation into methods of making this more salient.

7.3.2.3 Topic Evolution and Topic-based Collaboration

Figure 7-13 appears to contain too much information. It consists of information about time, person, topic, and amount and types of writing activities. Students generally were able to attend to two or three of these information categories, and only focus on other types of information when prompted or explicitly cued to do so in the interview.

Students differed in what type of information they direct their attention to; but many focused on the size of the circles (which is not the unique information to be gleaned from Figure 7-13), and this direction comes at the expense of attending to the more unique information offered by Figure 7-13, which is the interplay between different writing activities performed by different individuals for each topic. The instructor suggested discarding information about the amount of activity, making the circles the same size.

7.3.2.4 Writing Activity-based Contribution

The students perceived information about relative contributions by different authors/group members fairly easily from looking at Figure 7-14. This suggests that students are typically attuned to discerning whether the group members share equal workloads, etc.

Based on the feedback about the four types of visualisations, the following guidelines can be created for visualising collaborative writing processes:

- A timeline element is preferable for visualising collaborative writing processes.
- Visualisations need to be simple and not contain too much information.
- Topic evolution is an interesting aspect of collaborative writing.
- Author collaboration has to be presented in a simple manner.

These guidelines are used for creating the three types of visualisations explained in the next chapter.

7.4 Summary

This chapter presents a case study conducted to analyse documents written jointly by undergraduate students in an engineering course. The process mining tool ProM is used to analyse the collaborative writing processes. The dotted chart analysis is employed in order to obtain a bird's-eye view of writing processes and compare the start and end time of those processes in order to determine when students actually begin their writing tasks and how the activities spread over time. In addition, process models of individual groups are extracted using the Heuristic Miner algorithm. The models represent the dependency diagrams with frequencies of individual writing activities and dependency probabilities between two activities. The knowledge provided by the models sheds additional light on the mechanisms of the students' writing activities. Process models of student groups with different performances are compared and analysed. In addition, performance sequence analysis is utilised to extract sequences of writing activities and interactions of authors during the writing process; organisational mining uncovers writing activity-based collaboration events.

Based on several layers of semantics (i.e. text edits, and writing activities), a technique is presented to derive the Heuristic and Hidden Markov models of the documents written by groups of students. A case study is conducted to extract the two models and perform process analysis of two student groups who achieve a high and a low performance, respectively. The models represent the writing behaviours of these students. The heuristic MM offers a concise model that gives a good overview of the overall writing process, where each state of the model represents a defined writing activity (such drafting, revising, editing and so on); however, the Hidden MM, by discovering the states automatically, offers a finer-grained analysis by showing the sequences of text operations and transitional probabilities.

A pattern of writing activities and text edits can be extracted for a particular group using process models discovered by process mining techniques and Markov models. However, there are different patterns for different groups. This research would like to provide these patterns and behaviour models of writing activities as feedback, in the form of visualisations, to students and instructors while the students are engaged in writing their documents. As pointed out by Lowry (Lowry et al., 2003), group awareness, participation, and coordination are crucial elements for successful collaborative writing outcomes. For this reason, a pilot study is conducted

to provide mock-up visualisations to instructors and students (i.e. authors) such as a timeline, transition diagram, topic evolution and topic-based collaboration, and student collaboration based on statistics of writing activities. The first two types of visualisations are based on dotted charts, Hidden MM process models. Students are voluntarily interviewed to elicit their perceptions of the usefulness of the visualisations as well as feedback with regard to author awareness and collaboration. The feedback about the mock-up visualisations can be summarized as following: 1) Visualisations need to be simple and not contain too much information; 2) a timeline element is preferable for visualising collaborative writing processes; 3) Topic evolution is an interesting aspect of collaborative writing; 4) Author collaboration has to be presented in a simple manner. The result of this pilot study is used to develop further new types of visualisations which will be described in the next chapter.

CHAPTER 8

VISUALISING COLLABORATIVE WRITING PROCESSES

This chapter proposes three visualisation approaches and their corresponding underlying techniques for analysing the writing processes used in jointly authored documents from different points of view: the nature of text edits that occurred at the paragraph level (revision maps); the emergence and evolution of topics during writing activities (topic evolution charts), and the authorship collaboration information (topic-based collaboration networks).

First, revision maps are created which show a snapshot of the text edits performed by students on the collaboratively written documents. This visualisation depicts document development at the paragraph level over a period of time. Based on these paragraph text edits, topics are then extracted by using several types of probabilistic topic models. Topic evolution charts are used to obtain insight on the development of topics during course of the writing processes. Finally, topic-based collaboration networks are generated to analyse student collaboration with regard to the writing topics. These topic-based collaboration networks present network

diagrams indicating those students who write about the same topics as the document is developed.

This chapter begins by outlining an overview of the approach used in this research. Sections 8.2, 8.3, and 8.4 present revision maps, topic evolution charts and topic-based collaboration networks, in that order. The techniques are then validated with simulated data in Section 8.5, and Section 8.6 illustrates the applicability of the techniques using real world data of documents written by graduate students.

8.1 A Framework for Visualising Collaborative Writing Processes

This chapter extends the framework proposed in Chapter 5 to include visualising and analysing writing processes based on text edits and topic evolutions. Figure 8-1 depicts the architecture of this approach, which consists of a writing environment, i.e. Google Docs as the front end with Google Documents List API (GD API) for retrieving revisions and their information; a text comparison utility; Topic Model; and Author-Topic Model components. Chapter 5 describes GD API and its metadata: revision ID, author ID, and timestamp.

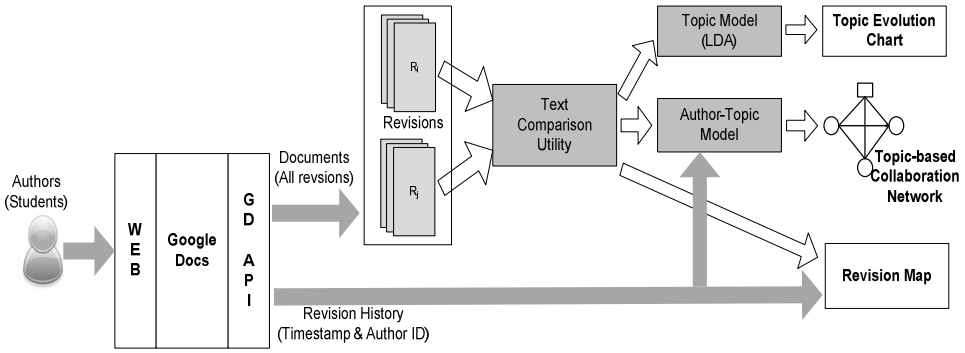


Figure 8-1. Framework of approach producing revision maps, topic evolution charts, and author-topic networks.

Three kinds of visualisations are generated. The first visualisation, the *revision map,* depicts text edits performed on individual paragraphs during the writing process. In order to understand the semantic of these text edits, the second visualisation – the *topic evolution chart* -- illustrates how written topics were created and developed during the writing process. The third visualisation, the *topic-based collaboration network,* delineates a network of collaboration between the authors based on the topics that they share in common. For example, the network indicates instances of any two authors writing about the same topics during their

tasks. The three types of visualisations described above will be elaborated in the next three sections.

As depicted in Figure 8-1, after retrieving the text content of all revisions and all the revision history for a particular document, a *text comparison utility* is employed to identify the text edits in successive revisions and establish a list of added and deleted text. These identified text edits, mapped against the list of author IDs and the timestamps of corresponding revisions, are used in the *revision map*; they are also used as input to both the topic and author-topic modelling algorithms. The topic modelling, especially DiffLDA (Thomas et al., 2011), creates the *topic evolution chart*, while the author-topic modelling (Rosen-Zvi et al., 2010; Rosen-zvi et al., 2003) outputs the *topic-based collaboration network* using author IDs provided in the revision history. The method for generating these three visualisations is described in detail the next three sections.

8.2 Revision Maps

It is necessary to obtain a chronological picture of the events during the course of collaborative writing in order to obtain a better understanding of how students develop their jointly authored document over a period of time. Revision maps summarise text edits made at the paragraph level throughout the writing process. Figure 8-2 depicts the revision map of a real document written by a group of students during the prototype experiment, which will be described in Section 8.6. Each column refers to a revision of the document. Each small rectangle represents a paragraph of the document. Each row shows the evolution of an individual paragraph, as it is created, altered, or deleted over a period of time.

Figure 8-2. Revision map of a real document written by a group of five students: c1, c2, c3, c4, and c5. "ad" is the administrator.

Rectangles are colour-coded to designate the nature and the extent of the edits made to the paragraph: *green* indicates that more words were added than deleted; *red* means more words were deleted than added; and *white* represents no change in the paragraph. The intensity of these colours increases or decreases depending on the extent of their corresponding edits. If the number of words added is the same as the number of words deleted, the rectangle colour is yellow-green. Lastly, the horizontal bar under the row corresponding to author IDs indicates the aggregated edits of individual revisions; and the last vertical column represents the aggregated edits of individual paragraphs across all revisions.

Each paragraph evolution is positioned relative to its position in the current (final) revision. This means that the paragraph evolution rows can move up and down over time, especially when a new paragraph is added. In addition, the paragraph evolutions are grouped into sections based on the structure of the document.

For instance, the revision map shown in Figure 8-2 represents the edits in a document written by a group of students over a period of six days (from 04 to 09/05/2011). The text edits of four paragraphs as indicated in the revision map -- P1, P2, P3 and P4 -- can be described as follows: Many words were added to the first paragraph of Section A (P1) by c1 on 04/05/2011 22:29; it was not edited until 06/05/2011 16:48 (by author c1), when more words were deleted than added to it. Towards the end of the week (on 08/05/2011 21:46), P1 was modified again by c1 when more words were added. The first paragraph of Section B (P2) was inserted on 05/05/2011 13:57 by author c2; it was not modified at all and was deleted altogether from the document on 09/05/2011 02:38 by c2. A new paragraph (P3) was inserted by c2 after P2 was removed. A paragraph can also be split and merged; for example, paragraph P4 was inserted on 05/05/2011 13:57 by c2, changed once by the addition of a few words on /05/2011 16:48 by c1, then split into two paragraphs on 09/05/2011 02:38 by c2.

The next subsection explains how to use the revision map depicted in Figure 8-2 to obtain further insight into how students collaboratively develop their document during the six days of writing.

Using the Revision Maps to Analyse Writing Process

Revision maps help provide answers to the following five questions:

1. Which sections of the document were worked on the most and which were worked on the least? (Location of text edits)
2. When (at what dates) did major edits (i.e. addition and deletion) occur during the writing process? (Time)
3. Did students work sequentially or in parallel? (sequential work consisting of single paragraphs written at different sessions or days; parallel work consisting of many paragraphs written almost simultaneously during the same writing sessions or days).
4. Who made the most or the least edits to the document? (Authorship)
5. How many authors worked on each paragraph and each section? (Collaboration)

Question 1 is easily answered by examining the vertical bar that represents the aggregated edits of individual paragraphs, which indicates that Section A of the document contains more edits than Section B. Answering Question 2 leads to the

interesting observation that most text edits, especially additions, happen at the beginning of the process when students first engage in their writing tasks, while many text deletions occur towards the end of the writing process. There are a number of extensive text edits made exclusively by author c3 during the middle of the process. In answering Question 3, it is interesting to note that the work in Section A is performed sequentially, especially on the first three paragraphs by two students, c1 and c5. In contrast, paragraphs in Section B were created almost at the same time by student c2, although these particular Section B paragraphs are not found in the final revision, but are replaced at the final stage of the writing process.

The revision map also provides information about how students collaborate during the writing process, thus answering Questions 4 and 5. Of the five students, we discern that c4 has the least amount of involvement with the development of the document; that the work of c2 pertains mainly to Section B; and that a significant amount of collaboration is evident from c1 and c5 with regard to their document development.

Although Figure 8-2 indicates very little editing activity from c4 as compared to the work of the other four students, it is nevertheless difficult to conclude from this evidence that c4 contributes the least to the development of the document. There is no way of knowing whether small text editions made by c4 serve to increase the assignment of a topic, thus improving the clarity and the coherence of the text.

This initial analysis derived from revision maps serves to further understand some of the conditions and circumstances related to the creation of the students' jointly authored document at the paragraph level. We will now look at ways to investigate how the students develop their ideas throughout the course of the writing, with particular emphasis on the evolution of the topics over time.

8.3 Topic Evolution Chart

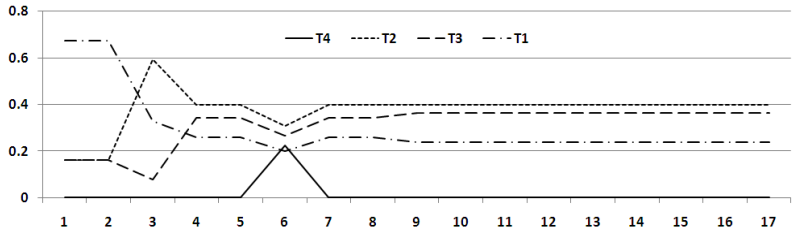


Figure 8-3. A topic evolution chart of four topics: *T1, T2*, *T3,* and *T4*.

Recognising the manner in which topics evolve during text edits helps in achieving a better understanding of how students develop their ideas and concepts while engaged in writing tasks. To shed light on this subject, the topic evolution chart shown in Figure 8-3 represents the development of four topics (*T1, T2*, *T3,* and *T4)* generated from a synthetic dataset, which will be described in Section 8.5. A topic consists of a cluster of words that frequently occur together in a revision, and each document revision is represented by a set of topics. The topic evolution chart depicts changes in the membership of topics throughout the sequence of revisions. For instance, in Figure 8-3, *T1-T3* appears at the start of writing (Revision 1), whereas *T4* emerges in the sixth revision (Revision 6) and disappears later in the writing process (Revision 7). The ratio of importance of the other three topics (*T1-T3*) changes over time. At the beginning of the writing process, the document contains more text related to topic *T1* than to *T2* or *T3 (66% vs 17%.*); however, towards the end of the process, topic *T2* is more dominant in the document than *T1* and *T3*.

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a popular probabilistic topic modelling technique which, at the time of this research, has never been used to extract the evolution of topics during the writing of a document. The closest method used for this purpose is DiffLDA (Thomas et al., 2011), which has been applied for extracting topic evolution in software repositories. In DiffLDA, the GNU *diff* utility is used to identify text edits only at the paragraph level before using LDA. The work in is research builds on the LDA and DiffLDA techniques in order to extract topics and their evolution during the writing process.

In this thesis, a text comparison utility is created to extract text edits at both paragraph and word levels, as noted in Chapter 6. Unlike in DiffLDA, the number of topics and hyper-parameters α and β (of the two Dirichlet distributions: author's topic distribution and topic-specific word distribution) are selected using a trade-off between the model fitting (i.e. perplexity) and the simplicity of model structure (i.e. the smallest number of topics). The number of topics is selected independently for each document. The following subsections furnish an overview of the probabilistic topic models Latent Dirichlet Allocation (LDA) and DiffLDA before describing the method of extending DiffLDA for mining topic evolution of writing processes. Table 8-1 provides the differenfce between this work and DiffLDA proposed in (Thomas, 2011).

Table 8-1, the difference between DiffLDA for software repositories (Thomas, 2011) and DiffLDA for writing processes.

DiffLDA for software repositories	DiffLDA for writing processes
Text differencing at paragraphs (lines) level	Text differencing at both paragraphs and word levels
Predefined number of topics and hyper-parameters	Inferred number of topics and hyper-parameters (using the model fitting and the simplicity of the model structure)

8.3.1 Probabilistic Topic Models

Topic modelling or Latent Dirichlet Allocation (LDA) automatically discovers *topics* within a corpus of text documents (Blei & Lafferty, 2009), in which topics are defined as collections of words that co-occur frequently in the corpus. Because of the nature of language usage, the words that constitute a topic are often semantically related (Thomas et al., 2010b). Each document is represented as a probability distribution over some topics, while each topic is represented as a probability distribution over a number of words. For instance, a topic consisting of the words {bank finance money cash loan} can be described as "the financial industry". In topic modelling, documents can be represented by the topics within them, and the entire unstructured corpus can be structured in terms of this discovered semantic structure. Subsection 3.1.1 provides the theoretical background of LDA including an example of topics and the generative procedure for extracting topics.

Topic evolution models using LDA suffer from the *duplication effect* as explained in Thomas et al. (2010a). These topic evolution models work on an assumption that documents in the corpus are unique across time. This assumption holds for the collection of journals, blog posts, and newspaper articles, which are typically studied in the topic modelling. It is very unlikely that an article published in one year is only slightly updated and republished the next year in the same conference proceedings. Instead, each article (i.e. the specific combination of words within an article) is unique across time. However, the nature of writing process is quite different. Jointly authored documents are usually updated incrementally from one revision to another revision as authors developed the documents. Although sometimes there can be lots of text edits occurring in one revision, there still exists

some overlap of text contents between the revision and the previous one. This particularity has been addressed with DiffLDA, which is described next.

8.3.2 DiffLDA for Mining Writing Processes

In order to address the data duplication effect found in software repositories, Thomas et al. (Thomas, 2011; Thomas et al., 2010a) proposes a simple technique for use in the pre-processing step before applying LDA to the source codes. On top of the normal pre-processing steps, they include the *diff* step to identify text edits between every pair of successive versions of each source code. In particular, for every pair of successive versions, DiffLDA uses the standard GNU *diff* utility to compute the edits (i.e. *add*, *delete* or *change*) at the line levels. According to DiffLDA (Thomas et al., 2011), if an existing line is changed, it is considered to be deleted and then added again. Identified edits (added and deleted lines) are then used as documents, called *delta documents* (Thomas et al., 2011). The corpus then consists of all delta documents in the software repository. This *diff* step effectively removes all duplications, thus preventing the occurrence of the duplication effect when LDA is applied to the corpus.

Nevertheless, the pre-processing step used in DiffLDA could not be applied directly in the context of this thesis. During the writing process, it is common for authors to revise a paragraph, which is a line in plain text, several times merely by changing certain words in the text, so that the number of words in the revised paragraph does not altered in any way. Using the pre-processing step of DiffLDA will generate many *change* edits for particular paragraphs or lines; consequently, the resulting delta documents will contain many duplicated words.

In this thesis, a text comparison utility (TCU) that consists of text differencing algorithms is developed to compute the edits between successive revisions. The text edits at paragraph levels are identified first; in other words, for each revision, it compares individual paragraphs to the corresponding paragraphs in the previous revision, using the GNU *diff* utility. This comparison classifies paragraphs as either added, deleted or changed, depending on whether the text edits from the previous revision that result in the current revision involve the creation of a new paragraph, the removal of a paragraph, or alterations made to a paragraph. TCU then computes text edits at word levels in the paragraphs that were altered, and classifies them as

either added, deleted, or equal (no change) depending on whether addition, removal, or no alterations occurred. The added and deleted words and paragraphs are then used as documents for LDA extraction of topics and topic evolution. The results obtained by using the method developed in this research thus succeed in preventing the duplication effect, as described below.

For each document, the text edits (at paragraph and word levels) of two consecutive revisions are first identified R_j and $R_{j'}$*(j'=j+1)* using the text comparison utility as explained above. For each document revision, two *delta documents,* $\delta_{j'}^{a}$ and $\delta_{j'}^{d}$ are created that capture addition and deletion types of text edits, recalling Thomas et al. (2011). We place all added word and paragraph edits between R_j and $R_{j'}$ into $\delta_{j'}^{a}$ and all deleted paragraph and word edits into $\delta_{j'}^{d}$. The whole of the first revision *(j=1)* is classified as added paragraphs, and is therefore added in its entirety to the delta document, δ_1^a. Using this method, each revision has a maximum of two delta documents, and a revision can have one delta document of either added or deleted paragraphs. LDA is then applied to the entire set of delta documents to produce a set of extracted topics and membership values for each delta document.

Finally, the topic membership values of the revisions is computed at each point in time by using the obtained topics and their membership values for each delta document. The following formula, proposed by Thomas et al. (2011), is applied to compute the corresponding vector of a revision i defined recursively as

$$\theta_i = \frac{\theta_{i-1}|r_{i-1}| + \theta_{\delta_i^a}|\delta_i^a|}{|r_{i-1}| + |\delta_i^a|} - \varphi\left(\frac{\theta_{i-1}|r_{i-1}| - \theta_{\delta_i^a}|\delta_i^d|}{|r_{i-1}| - |\delta_i^d|}\right)$$

Where $|\delta_i^a|$ represents the number of words in δ_i^a, $i-1$ is the index of the previous revision of the document and φ() is the normalising function, Thomas et al. (2011), suggests that this is necessary in a scenario where more words matching a given topic were subtracted in a document than were subtracted in the previous version of that document for that topic.

It is necessary to select the number of topics and set parameters, α and β, of the two Dirichlet distributions -- document's topic distribution and topic-specific word distribution – before applying LDA to the entire set of delta documents. The following section explains the method for selecting hyper-parameters and the number of topics.

8.3.3 Hyper-parameter Selection

LDA and particularly DiffLDA require the setting of parameters, α and β, of the two Dirichlet distributions: author's topic distribution and topic-specific word distribution. There has been relatively little work within the topic modelling community on the appropriate selection method of hyper-parameters (Broniatowski & Christopher, 2012) except for the algorithm proposed by Wallach (2008), which overfit hyper-parameters for the purpose of this analysis; however, this algorithm can slow the convergence of the Markov chain. This thesis uses a strategy to fix α and β depending on the number of topics, *T,* and explores the consequence of every *T*. The techniques proposed by Griffiths and Steyvers (Griffiths & Steyvers, 2004) is then used to set the value of α*=50/(#topics)* and β*=200/(#words)*.

8.3.4 Selection of Number of Topics

After defining the hyper-parameter values as mentioned above, the number of topics (T) is chosen by using perplexity (Griffiths & Steyvers, 2004), which is a standard measure for estimating the performance of a probabilistic model based on its ability to predict the words contained in new unseen documents. The smallest possible *T* to maintain a good model fit was selected as describe below. The number of topics is selected independently for each document; LDA models are fitted to the delta documents for t=1…50 topics, resulting in 50 models for each document. For each of these models, 20 independent samples are generated from one randomly initialised Markov chain after a burn-in of 1000 iterations, guaranteeing the independence of the samples by having a lag of 100 iterations between each one. The smallest value t_0, is taken, so that the 95th percentile of all samples for all larger values of t is greater than the 5th percentile of t0 (Broniatowski & Christopher, 2012). Figure 8-4 depicts the typical trend of the perplexity of DiffLDA model fits. The recommended value of $T=t_0+1$ is selected (Broniatowski & Christopher, 2012) to ensure that the chosen model is not too fit and can be generalised for modelling data.

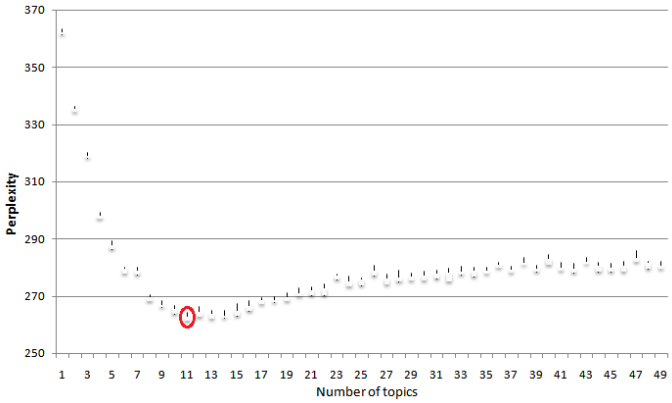


Figure 8-4. Perplexity vs number of topics for a document written by graduate students (from the case study described below). The selected number of topics is equal to 12 as explained above.

After the number of topics, *T*, has been selected, a *T*-topic LDA model is fit to all delta documents. Ten samples are taken from 20 randomly initialised Markov chains, to obatin 200 samples in total. The results of the final samples are used to construct topic evolutions of writing collaboration by showing the change of the distribution of topics over time.

The extracted topics and topic evolutions provide an overview of how topics are created and the way that they evolve. Knowing whether students collaborate and if they often write about the same topics assists both instructors and learners in understanding how the documents are developed. The topic-based collaboration networks created in this thesis with the purpose of further investigating learner collaboration are explained in the next section.

8.4 Topic-based Collaboration Networks

For further analysis, it is useful to visualise how students collaborate around topics, with particular emphasis on ascertaining whether students develop their ideas and concepts independently or whether they work together on the same topics. Figure 8-5 shows a topic-based collaboration network from a group of four students jointly writing a document for the prototype experiment, which will be described in Section 8.6. Each node represents a student author. A square depicts a group coordinator. Circles represent group members. A connection (link) between two nodes indicates that those two students have written about the same topics during their tasks. Figure 8-5 shows that the group coordinator *a1* and group member *a2* have both worked

with all group members to draft, revise, and edit some of the document topics. The group coordinator has a responsibility to assign writing tasks to individual members and to make sure the assigned tasks progress according to plan. Group members *a3* and *a4*, however, have not written about the same topics. In other words, *a3* and *a4* have both worked independently with *a1* and *a2* to develop some topics.

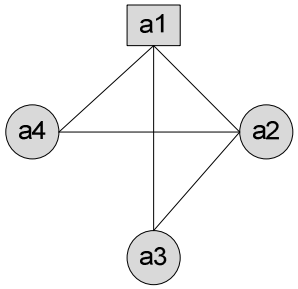


Figure 8-5. A topic-based collaboration network for collaborative writing. The network is inspired by the social network proposed by Broniatowski and Christopher (2012). Nodes represents students: *a1* to *a4*. The square is the group coordinator and circles are group members. A connection between two nodes means that the two corresponding students have written about the same topics.

The contribution of this thesis toward accomplishing the visualisation resides in the creation of a Diff Author-Topic Model (DiffATM), which is an extension of Author-Topic Model (ATM) (Rosen-zvi et al., 2003). As DiffLDA overcomes the duplication effect in LDA, DiffATM is developed to deal with the duplication effect in ATM. In this research, similarly to DiffLDA, DiffATM is applied to text edits identified at the paragraph and word levels in order to extract topics. The application of DiffATM, however, instead of providing a cluster of topics per revision, provides a cluster of topics per author. Based on a number of revisions, a particular author can be represented by a membership of topics written in those revisions. Like DiffLDA for writing processes, DiffATM is developed by selecting the number of topics and hyper-parameters based on the trade-off between the model fitting and the simplicity of model structure. In addition, social networks are applied as proposed by Broniatowski and Christopher (2012) for collaborative writing tasks based on the membership of topics of individual authors.

The subsequent section describes the Diff Author-Topic Model followed by a description of the method used for constructions of topic-based collaboration networks.

8.4.1 Diff Author-Topic Model for writing processes

This thesis develops Diff Author-Topic Model (DiffATM), which in turn uses a variant of LDA known as the author-topic (AT) model (Rosen-zvi et al., 2003) which adds probabilistic pressure to assign each author to a specific topic. Shared topics are therefore more likely to represent common ideas and concepts. The DiffATM model provides an analysis that is guided by the authorship data of the documents (provided by revision histories) and the word co-occurrence data used by DiffLDA. Each author is modelled as a multinomial distribution over a fixed number of topics that is selected empirically as explained below. Each topic is, in turn, modelled as a multinomial distribution over words.

As described in Subsection 8.1, the Text Comparison Utility (TCU) outputs the delta documents (i.e. added and deleted paragraphs) and each revision is produced by one or more authors. The authors of a revision are assigned to the delta documents of that revision. The Author-Topic Model (ATM) is then applied to the entire set of delta documents.

As in DiffLDA, the hyper-parameters defining each Dirichlet prior (α and β) of DiffATM are dependent on the number of topics, which is selected independently for each document using the trade-off between the model fitting and the simplicity of the model structure as described in Subsection 8.3.4. The likelihood of two authors writing the same topic will depend on the hyper-parameters chosen (Broniatowski & Christopher, 2012). In general, larger values of α will lead to more topic overlap for any given corpus, motivating the use of a consistent hyper-parameter selection algorithm across all corpora analysed. All hyper-parameter settings used for the analyses presented in this thesis follow the guidelines derived empirically by Griffiths and Steyvers (2004). In particular, α = 50/(# topics), inducing topics that are mildly smoothed across authors, and β = 200/(# words), inducing topics that are specific to small numbers of words.

Like DiffLDA, the DiffATM model is fit by using a Markov-chain Monte Carlo (MCMC) approach. Information about individual authors is included in the Bayesian inference mechanism, so that each word is assigned to a topic in proportion to the number of words by that author already in that topic, and in proportion to the number of times that specific word appears in that topic. Thus, if two authors use the same word in two different senses, the DiffATM model will account for this polysemy.

Details of the MCMC algorithm derivation are given in the paper by Rosen-Zvi et al. (2003).

After the number of topics, *T*, has been selected, a *T*-topic DiffATM model is fit to all delta documents. Ten samples are taken from 20 randomly initialised Markov chains, such that there are 200 samples in total. The result of the final samples are used to construct topic-based collaboration networks, as described below.

8.4.2 Construction of Networks from Topics

After an ATM has been fit, networks are constructed networks in order to analyse student collaboration, with particular interest in linking together two students who often use the same topics of discourse over the writing period. The same method proposed by Broniatowski and Christoper is used (Broniatowski & Christopher, 2012), in computing the joint probability of each pair of authors writing about the same topic as:

$$P(X_1 \cap X_2) = \sum_{i}^{T} P(Z = z_i|X_1)P(Z = z_i|X_2)$$

A joint probability of two authors which exceeds 1/T (e.g. 0.1 if T=10) is indicated by creating a link between the two nodes; the reason for choosing this condition is explained in (Broniatowski & Christopher, 2012). A square author-author matrix is constructed with entries equal to one for each linked author pair, and entries equal to zero otherwise. This procedure is then repeated several times for each document (Broniatowski & Christopher, 2012) to average across whatever probabilistic noise might exist in the DiffATM fit. Authors who link across multiple DiffATM fits more often than would be expected according to chance are considered to be linked in the network for that document. The author-author matrix is obtained after 200 samplings of DiffLDA. Each author pair with an entry higher than 125 is considered as linked. Five topic-based collaboration networks of four student goups are presented in

Figure 8-6, showing different networks with different numbers of connections, which demonstrates that the dynamic of topic sharing during the writing process differs among groups.

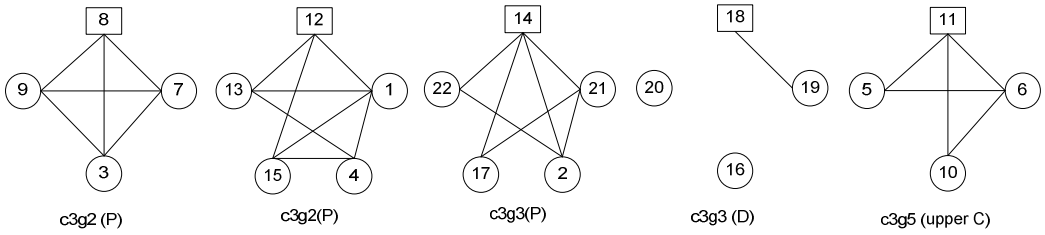


Figure 8-6. Topic-based collaboration networks of four different groups of students writing documents. Squares represent group coordinators. Circles are group members. Links between two nodes indicate that the two corresponding authors have written about the same topics.

8.5 Technical Validation

This section formalises a validation of the accuracy of DiffLDA and DiffATM as used in constructing topic evolution and topic-based collaboration networks. Since there is no public dataset for evaluating the accuracy of topic evolution models, a synthetic dataset is formulated for that purpose. Inspired by Thomas et al. (2011), a simulation of text edits on a document, situated in two simple scenarios that represent several types of text edits, is created in order to evaluate the accuracy of the evolutions discovered by the models. Specifically, the dataset was intended to verify if the text edit events detected by the models correspond with the actual changes that were made during the writing, thus evaluating precision; and if the discovered evolutions contained all the text edits that were actually performed during the writing, thus evaluating recall.

8.5.1 Data Generation

Evaluation of the DiffLDA model for collaborative writing begins with the creation of a document with 17 revisions (R1 – R17) consisting of three paragraphs which are generated from three topic distributions that are equally weighted. Table 8-2 shows the dictionary and topic distribution of the data. After each paragraph is created or first added to the document, it is changed three times (these changed paragraphs are also generated from the three topic distributions as presented in Table 8-2). Table 8-3 illustrates the text edits of these paragraphs. It is important to note that no text edits were performed on some of the revisions. The 17 revisions form a baseline scenario.

Table 8-2, the dictionary and topic distribution of a simulated data

Words	T1	T2	T3
River	0.37		
Stream	0.31		
Bank	0.22	0.28	
Money		0.3	0.07
Loan		0.2	
Debt		0.12	
Factory			0.33
Product			0.25
Labor			0.25
News	0.05	0.05	0.05
Reporter	0.05	0.05	0.05

Table 8-3, event log file presenting text edition events of revisions of a simulated document.

Rev.	P1	P2	P3	P4	P5	P6	C1	C2
R1	a				a		a1	
R2	-				-		a1	
R3	-	a			-		a2	a5
R4	-	-	a		d		a3	
R5	c	-	-				a1	
R6	-	-	-	a			a4	a1
R7	-	-	-	d			a4	a1
R8	-	c	-	-			a5	
R9	-	-	c	-			a3	
R10	c	-	-	-			a1	
R11	-	c	-	-			a5	
R12	-	-	c	-			a3	
R13	c	-	-	-			a1	
R14	-	c	-	-		a	a2	a5
R15	-	-	-	-		-	a2	
R16	-	-	-	-		c	a2	
R17	-	-	c	-		-	a3	

Note: The baseline scenario consists of three paragraphs P1, P2, and P3. The first controlled scenario (C1) is formed by adding and delete P4. The second one (C2) adds and deletes P5 and adds and changes P6. There are four text edition events: no change, adding, changing, and deleting a corresponding paragraph, presented as '-', a, c, and d, respectively. Each revision is produced by no more than two authors. There are five authors: a1 – a5.

Two simulated scenarios are set up as follows:

The first scenario modifies the baseline scenario by adding one paragraph in revision R6, as shown in Table 8-3, and then deleting it in revision R7, thus simulating the addition and deletion types of text edits, using a paragraph generated from a new topic (i.e. the four code names of Ubuntu operating system) totally unrelated to the three topics in the baseline scenario.

The second scenario is created by adding two paragraphs: First, a paragraph from a new topic unrelated to the four topics mentioned above is added in the first revision, R1; it remains (unchanged) in revisions R2 and R3; and is then deleted in revision R4. Second, a paragraph from another unrelated new topic is added in R14 and R15. The first half of the paragraph is added in revision R14, while the second half of the paragraph is added in the final revision, R16. This scenario demonstrates multiple text edits happening simultaneously in the same revisions.

Table 8-3 displays the text edition events. The simulation is designed in such a way as to ensure that there are no more than four paragraphs in any of the revisions. at any given time.

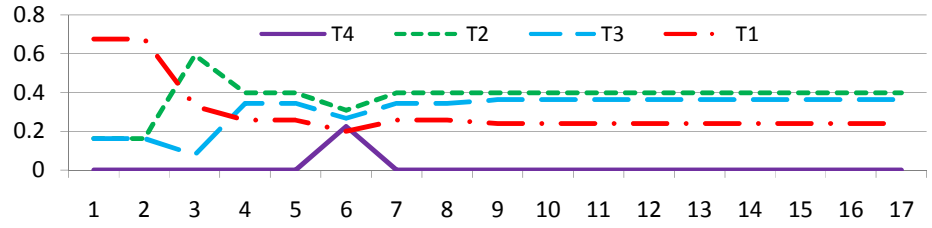
8.5.2 Pre-processing and Study Setup

Pre-processing is performed after the process of identifying text editions and creating delta documents described above. For the analysis reported in this chapter, a word-document matrix and author-document matrix are constructed using *doc2mat* utility from the CLUTO package (Steinbach et al., 2000), which removes all stop-words and stems all words to their roots using the Porter stemming algorithm.

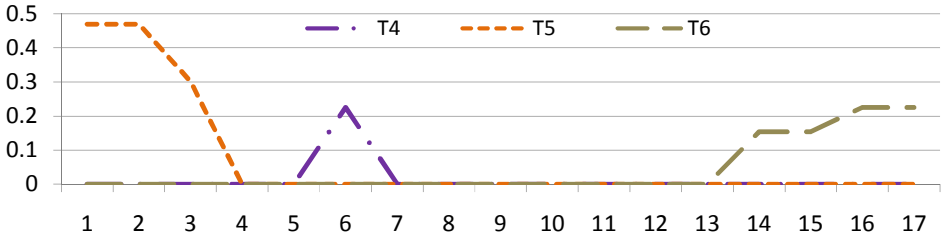
For Scenario 1, the pre-processing results in a total of 417 words (15 of which are unique) in 23 delta documents. There are (M=18.13, STD=0.81) words per revision. Scenario 2 consists of 485 words (23 of which are unique) in 26 delta documents. There are (M=18.65, STD=4.25) words per revision in Scenario 2.

The Topic Modelling Toolbox (Toolbox, 2012) implemented in MATLAB is used for the actual LDA and ATM computation. A total of 500 sampling iterations are performed. Because the amount of simulated data is quite small, no parameter optimisation is performed, thus setting the burning period.

8.5.3 Results



(a) The topic evolution of four topics T1-T4 in Scenario 1.



(b) The topic evolution of T4, T5, and T6 in Scenario 2.

Figure 8-7. Topic evolution for the simulated scenarios.

Scenario 1 consists of a change in Topic 4 when a paragraph is added. Figure 8-7(a) shows that the model detects the topic because the evolution of T4 has a value of 0 in all revisions except in revision R6, where its distribution spikes to slightly more than 20%. Upon checking the corresponding revision, especially the added paragraph, it is discovered that the paragraph has high membership in this topic and low membership in all other topics; and in fact, this is the only paragraph that has a non-zero membership in this topic.

Figure 8-7(b) shows the discovered topic evolutions for Scenario 2. The model indeed captured all three changes of the topic evolutions.

Based on the simulated authors a1-a5, an evaluation of the technique used in constructing the topic-based networks is also conducted. A network diagram is correctly obtained that shows five nodes and two links: the first link connects a2 and a5, both of whom work on P2 either alone or at the same time; and the second link connects a1 and a4, who together add and delete P4.

From the evaluation above, it is concluded that using DiffLDA does indeed discover topic evolutions for writing processes, and constructs topic-based collaboration networks that correctly identify authors who write about the same

topics. The following section illustrates the applicability of the techniques proposed in this research by using real documents in a real learning environment.

8.6 Prototype Experiment

This section presents a prototype experiment conducted in a semester-long graduate course called "Foundation of Learning Science" at the Faculty of Education and Social Work, University of Sydney, 2012, with the aim of deploying the techniques discussed in this chapter within a course in order to illustrate how the three visualisations described above are used, what information they provide, and whether they are useful.

In the pilot study presented in the previous chapter, the visualisations were produced manually by synthetic data based on several process models and used to obtain guideline to develop the visualisations presented in this chapter. In this prototype experiment, the three visualisations were extracted and given to students (authors) while they were writing their documents in order to provide feedback mirroring their writing activities. Therefore, for this purpose, it was required to conduct a separate experiment. As a result, a different dataset was obtained and used in process analysis based on the three visualisations for investigating how individual paragraphs were developed, how topics were evolved, and how authors collaboratively wrote about those topics. By contrast, Dataset A was used for extracting process models (presented in the previous chapter) in offline manner to extract patterns of writing activities that authors performed during writing processes.

The intension of this experiment is to discover how individual groups of students collaboratively performed their writing tasks and developed their ideas during their writing processes. Unlike the case study of extracting process models presented in the previous chapter, we do not intend to use the visualisations to identify the writing processes that produce high and low quality outcomes. Although in this experiment individual groups also obtained final assessment of their written documents, the topics of individual documents were different as described in the experiment setup below. Consequently, their revision maps (especially the structure of written paragraphs) and topic evolutions would be totally different and not appropriate to compare. Although topic-based collaborative networks were extracted and presented side by side in the analysis below, they were used to investigate if authors of

individual groups collaboratively wrote about the same topics. They were not used to discover the types of collaboration that lead to high and low performance.

8.6.1 Experiment Setup

8.6.1.1 Course Setup and Dataset

There are 22 students in the course, which is structured in the following way: Every two weeks, the students are divided into five different groups made up of four or five members. During each fortnight, groups are required to write about a topic (which varies every fortnight) in a jointly authored document of approximately 3000 words. For this study, the writing duration of each fortnight is called a *cycle*. This writing component of the course lasts for 12 weeks, i.e. six cycles; in other words, throughout the semester, every student collaboratively writes six documents. At the end of the semester, there are a total of 30 documents for analysis, all of which are assessed and graded as either Pass (P), Credit (C), Distinction (D), or High Distinction (HD).

During the two weeks of writing about an assigned topic, individual students in each group are assigned reading materials, with six readings per group. Students are encouraged to incorporate ideas and concepts learned in the class lectures and in the reading material into their writing tasks. For every document, students are required to make a plan for their writing tasks and discuss it with the group members during the first week of the cycle.

Each document is comprised of two sections: in the first one, section A, students are required to write about their assigned reading materials. They are asked to describe the main ideas of the articles they read, to provide evidence that they are grappling with these ideas, and to articulate difficult concepts and put them into context. Evidence of the writer's ability to engage in critical thinking is also to be included in this section.

In the second section (section B) students are required to identify relationships between the assigned readings for this cycle and those of the previous cycle, and to specify the “big ideas” contributed by the reading materials.

The visualisations in this experiment are also used as feedback for students during their writing tasks. Revision maps are produced every week for all the groups

(i.e., two revision maps per cycle per group) to enable the students to see the evolution of their work. For each cycle, at the end of the first week, revision maps were created to depict the evolution during the first week. At the end of the cycle, the final revision maps were produced to show the evolution in the past two weeks (during the cycle). Figure 8-8 shows the timeline when the first and final revision maps were created and provided to students during the experiment.

At the beginning of the experiment, the student groups receive an example of the revision map, similar to Figure 8-2, with a description of what the map was about and how it can be used, and a questionnaire is posted online so that students can participate in the study on a voluntary basis (the example, description and questionnaire are shown in Appendix E). Besides being voluntary, participation is also anonymous; students use only their group IDs in recording answers to the questions.

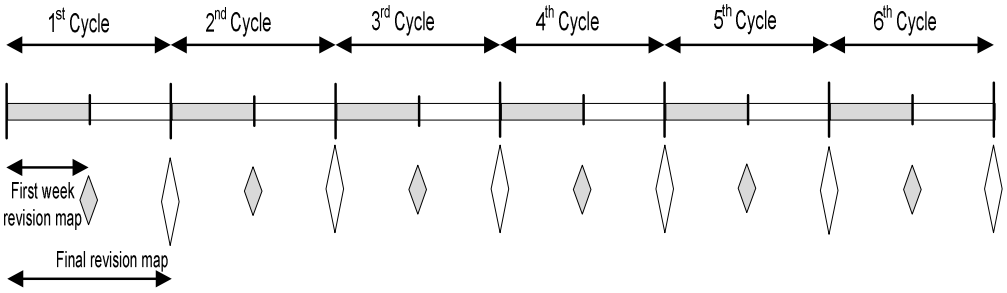


Figure 8-8. Experiment setup of six cycles (fortnights). Two revision maps were provided for each cycle: the first week revision map presented by a grey diamond and the final revision map presented by a white diamond.

8.6.1.2 Questionnaire for Qualitative Evaluation

The aim of this experiment is to help individual student authors recognise how the text content of their documents changes over time by answering 12 questions as shown in Table 8-4. The first five questions are concerned with the usability of the visualisation, and the remaining seven questions seek information about the quality of the visualisation.

Table 8-4. Questionnaire for qualitative evaluation

No.	Question
1	Given a revision map, can student authors specify which parts (sections/paragraphs) of the document have been revised significantly, slightly, and not changed at all?
2	Given a revision map, can student authors indicate at what stage were a large number of words added to or deleted from the document? How long does it takes to produce each part (section/paragraph) of the document?
3	Given a revision map, can student authors specify who made the most and the least changes to the document?
4	Given a revision map, can student authors report how many members of the group work on individual parts (i.e. sections/paragraphs) of the document?
5	Overall, based on a revision map, can students identify text change patterns that occur during the writing: sequential, parallel, or others?
6	Based on a revision map, what do students think about **how** their group wrote the document? e.g. explaining and discussing problems that their group encountered during writing.
7	How accurately does the visualisation represent what actually happened during the writing process, especially with regard to question number 1-5?
8	From the visualisation, do students learn something that they did not know before?
9	From the visualisation, would students do something differently in retrospect?
10	How do student authors use the visualisation during their writing? Do the visualisation prompt them toward any problems (e.g. member contribution or group coordination) thereby alerting them to **do something**, or **plan an action**, regarding their group's writing?
11	What do students want to know about the collaborative writing process that is not represented in the visualisation?
12	Would students find the visualisation useful to be provided with this kind of visualisation for their next group writing assignment? [on a *scale from 1 (strongly disagree) to 7 (strongly agree)*]

It is necessary to refer to the groups' revision maps in order to answer the first five questions. The remaining questions solicit student opinions on the usability and accuracy of the revision maps.

8.6.2 Analysis

Four cycles are selected for our analysis from the six fortnightly cycles of writing: the third, fourth, fifth and sixth. There are five groups of students in each cycle, hence five documents are produced (20 documents in total). After downloading all the revisions of these documents, the text comparison utility is applied to identify the text changes that produce these revisions, which generate delta documents containing the added and deleted paragraphs.

Table 8-5. Numbers of revisions, vocabularies (unique words), delta documents, authors per revision, and final marks of all documents.

Group	#revisions	#delta documents	#vocabularies	#total authors	# inferred topics	Mark
c3g1	49	73	821	4	10	P
c3g2	61	85	1040	5	11	P
c3g3	144	229	1056	5	32	P
c3g4	36	47	640	5	9	D
c3g5	46	67	844	4	11	Upper C
c4g1	86	141	1038	5	18	Upper C
c4g2	46	68	753	4	11	D
c4g3	35	43	727	4	8	P
c4g4	37	47	864	5	9	P
c4g5	46	51	873	4	10	P
c5g1	137	225	1137	5	24	D
c5g2	120	104	1323	5	28	Upper C
c5g3	40	62	953	5	12	Upper C
c5g4	44	66	749	4	10	Upper P
c5g5	42	65	646	3	9	Upper C
c6g1	55	77	871	5	11	D
c6g2	150	255	868	4	18	Lower D
c6g3	26	36	727	4	7	C
c6g4	75	118	966	4	15	C
c6g5	54	75	1040	5	14	C
Mean	66.45	96.7	896.8			
STD	39.33	65.50	173.86			

Table 8-5 summarises the number of revisions, delta documents, vocabularies (unique words), and authors of the 20 documents. The table also shows the final grades. Each group is identified by "*cXgY*", where *X* is the cycle number and *Y* is the group number in that cycle. For instance, *c3g5* is the group number 5 of the third

cycle. This study is concerned mainly with major revisions as defined in Chapter 5. The number of revisions (M=66.45,STD=39.33) varies from 150 for *c6g2* (receiving a grade of D) to 26 for c6g3 (receiving a grade of C). The number of delta documents also varies from 255 for *c6g2* to 36 for *c6g3*. It is important to note that the number of delta documents is not proportional to the number of revisions. For example, *c4g1* produced 141 delta documents within 86 revisions, whereas *c5g2* which contained 120 revisions (i.e. 120 revisions) produced only 104 delta documents.

To further elucidate the document development process, the information that can be obtained from the proposed visualisations is presented below.

8.6.2.1 Revision Maps

After identifying text edits made on all revisions for each document as described in Section 8.1, revision maps are created for each document. Using the revision maps of individual documents, it is possible to discern how individual paragraphs of the two sections (A and B) are created and how they evolve during the process of writing. The five questions presented in Section 8.2 are used to analyse the five documents of the third cycle.

Based on the task description mentioned in Subsection 8.6.1.1, two hypotheses are formed for analysis in this subsection. Firstly, for Section A, it is expected that the individual students, working on their own, develop their own ideas and topics from the assigned readings by writing several paragraphs that explain their ideas and show evidence of their understanding of the material. Secondly, for Section B, students are expected to engage in a significant amount of collaboration to relate the ideas developed from the readings, with paragraphs in this section edited by several group members.

Figure 8-9 shows the revision map of the first week of Group c3g4' writing process. Figure 8-10 shows the final revision map of the two-week period of Group c3g4' writing process. The revision maps of other 4 groups in this cycle are shown in Appendix D. Note that for this experiment, the number of words affected by text edits was shown in the revision maps.

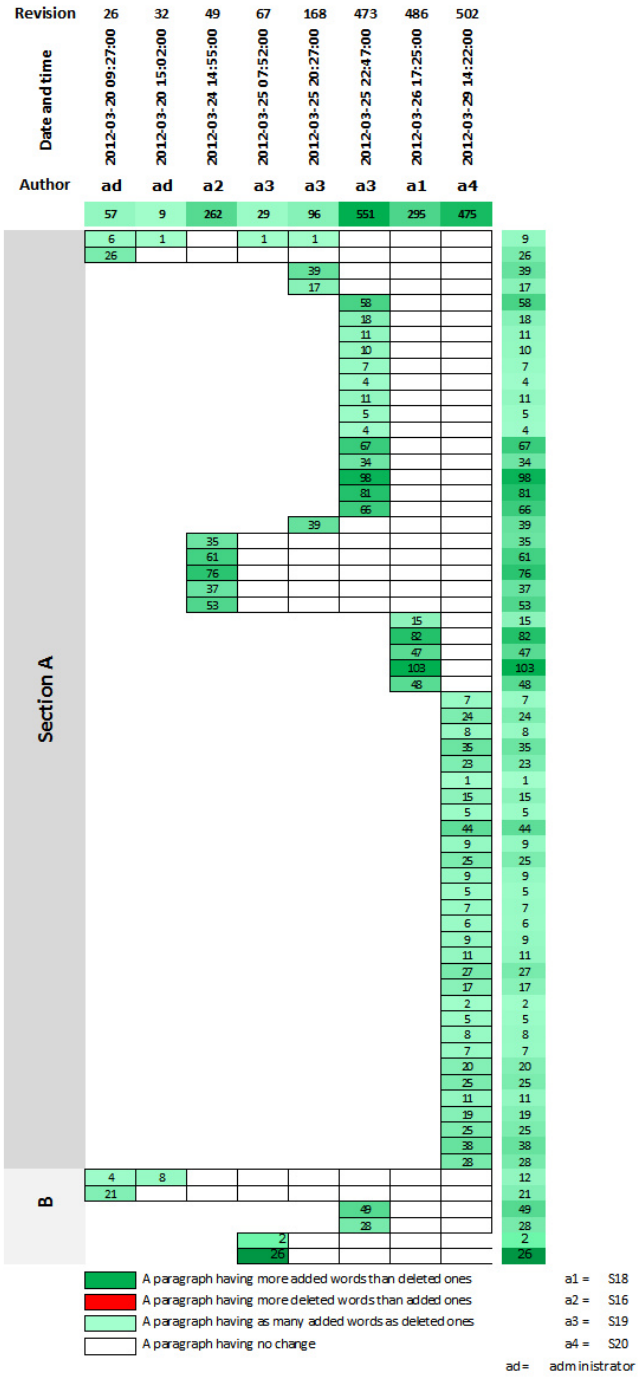


Figure 8-9. The first week revision map of Group c3g4.

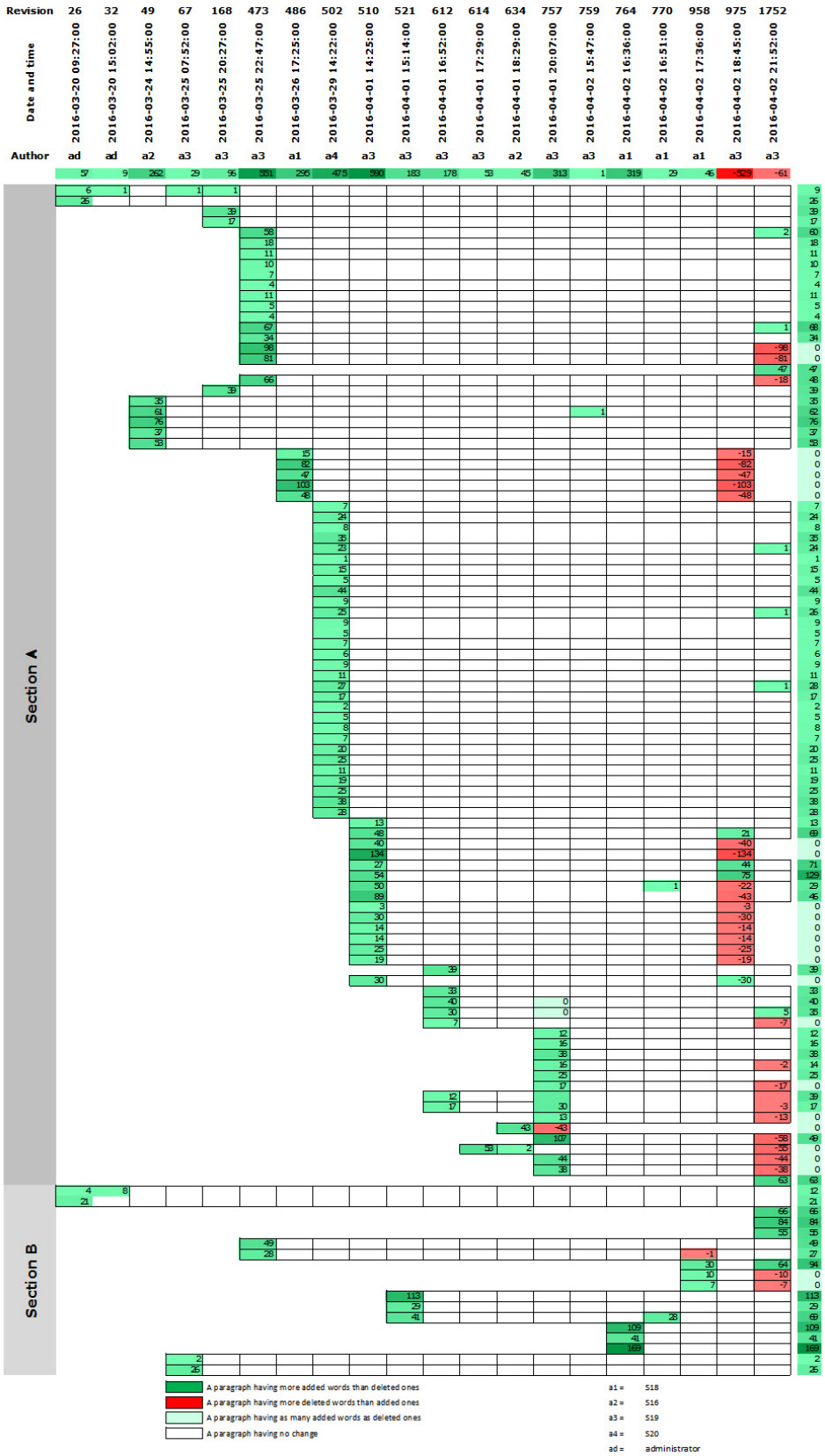


Figure 8-10. The two-week revision map of Group c3g4.

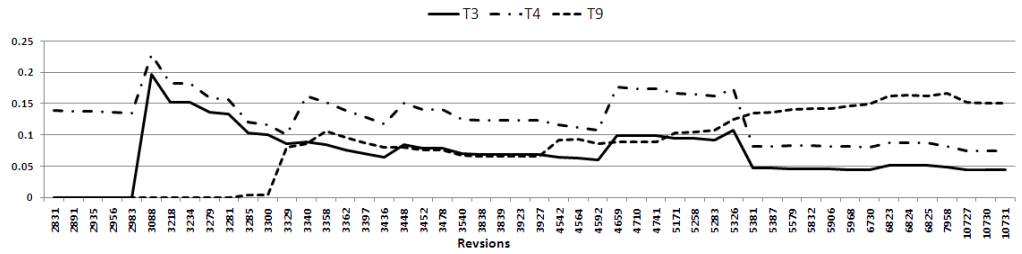
As expected, Section A was created before Section B in all five documents. In fact, many text edits are made in Section A at the beginning of the writing process, and a significant number of text changes are also produced in Section B towards the end of the writing; this suggests that most students spend their time writing in the beginning, then rush to finish their writing toward the end of the process. In all five documents, more than 50% of the paragraphs are created and changed during the same writing sessions or days, revealing that most students prefer to write in one session rather than drafting sequentially over several days.

With regard to the authorship of the edits, in all five documents, most of the paragraphs in Section A are edited and revised by only one student. For Section B, many paragraphs are edited by more than one student. The number of paragraphs in Section B written by several students is more than fifteen for *c3g1*, six for *c3g2*, nine for *c3g3*, five for *c3g4*, and five for *c3g5*. This indicates that, as expected, most students collaborated to write Section B.

8.6.2.2 Topic Evolution Charts

The pre-processing step outlined in Subsection 8.5.2 is first performed and the number of topics for each individual document is chosen. As stated in Section 8.3, unlike in other works (Thomas, 2011), the number of topics, T for each document was determined by fitting the LDA models to their delta documents and selecting the model providing the good perplexity. The number of topics chosen for each document is shown in Table 8-5. After that, the technique described in Section 8.3 is applied to extract topics and create topic evolution charts.

As the creation and development of every topic evolution chart for each document, is examined, either topics in Section B emerge and develop later in the process than topics in Section A, or vice versa. The expectation is that the former scenario is more likely to occur because students begin their writing tasks and idea development while working in Section A based on the assigned reading; and they later work with others to further develop the writing in Section B.



T3	person learn hamilton student creat connect peer specif tool classroom
T4	individu develop phase affect posit support student type content time
T9	metacognit teacher recognitmotiv process appreci learn help goal mean

Figure 8-11. Topic evolution map of three topics T3, T4 and T9 over 50 revisions of Document 2. The table above shows the top 10 words of each topic.

Figure 8-11 shows the topic evolution map of some of the topics of c4g2. There are 11 topics for this document. The topic evolution chart depicts only three topics: T3, T4, and T9. The top ten words used in the three topics are also shown below. Again, the topic evolution map provides an insight into how topics are developed by the students as they write. In particular, T4 is about the instructions for and explanations of the assignment; it appears at the beginning of the document and decreases over time. Unlike T4, T3 is about reading material related to the work of Hamilton on a "theory of personalized learning communities". Students wrote to reflect on this topic, and it spikes up at the third revision. T9 arrives after the two topics already mentioned because it is part of Section B of the document; it is about "teacher's recognition of their learners' cognitive and motivational potential". Although the evolution of topics during the writing is detected, this research also seeks to learn whether students write about the same topics over time. To answer this question, purpose, an analysis based on the topic-based collaboration networks is undertaken.

8.6.2.3 Topic-Based Collaboration Networks

In terms of topic based author collaboration, it is obvious to expect that for each group (each topic-based collaboration network) there is at least one link connecting two nodes, because at least two students collaborate and write about the same topics in Section B, as previously explained. This link, if it exists, may be one that connects a group coordinator (node) to another team member (node) depending on the nature of the text change operations performed by individual group coordinators. If a group coordinator only edits by performing surface changes, there will not be any links

connecting the coordinator with other group members; however, coordinator revisions that elaborate topics developed by other group members create a link between the coordinator and others. This event is not strictly required and is quite difficult to check, since the group coordinator's responsibility is to assign writing tasks to individual members and to make sure the assigned tasks progress according to plan, which means that the coordinator does not necessarily spend time collaborating and writing about the same topics with other group members.

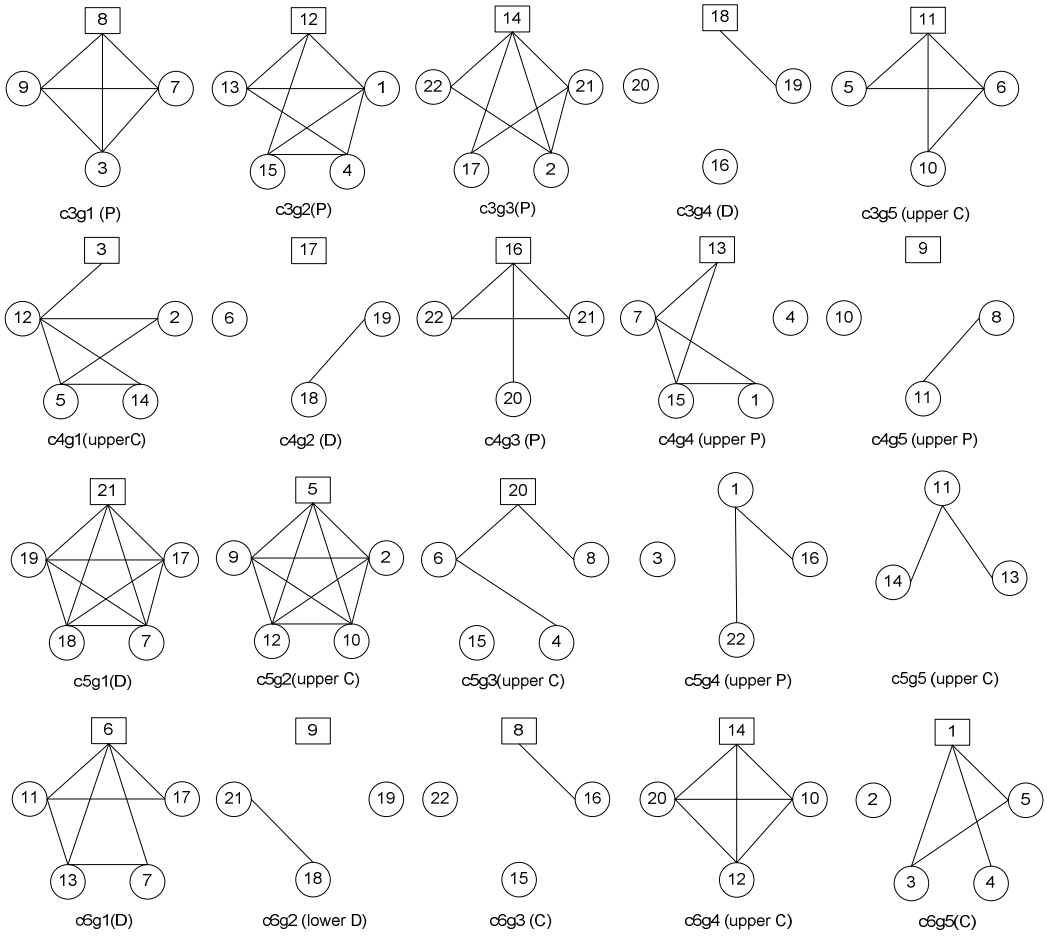


Figure 8-12. Author-topic networks of 20 documents of four cycles. Each row shows a cycle, which is a writing period of two weeks. Squares depict group coordinators and circles are group members. The edge connecting between two nodes represents two corresponding students writing the same topics.

The technique described in Section 8.4 is used to obtain the networks shown in Figure 8-12. Each row represents five groups of students writing collaboratively during a period of two weeks as described in Section 8.6.1. Nodes depict individual students with identification numbers from 1 to 22: squares represent group

coordinators and circles designate group members. There are a total of 22 nodes (students) for each row. Nodes are clustered according to the students' group assignment. For each group, an edge linking between two nodes shows that the two corresponding students have written about the same topics during the writing process.

The networks can be categorised from single-edge graphs (i.e. c3g4, c6g2, and c6g3) to connected graphs with all pairs of nodes linked (i.e. c3g1, c5g1, c5g2, and c6g4).

According to the hypothesis formulated in this research, each network should have at least one edge, because all the students in each group are required to write about the same topics, especially for document section B; and as verified above, all networks do indeed have one or more edges.

The next point to investigate is the appearance of at least one edge for all square nodes. Since group coordinators are expected to collaborate with other group members in order to draft, revise or edit topics, each group should display some edges connecting a square to some circles. Fulfilling this expectation, most networks exhibit at least one edge linking their square to a circle node. The only exception appears in c6g2[3]; for this group, although there is one edge connecting two members who write the same topics, neither of them is the coordinator.

The topic-based networks of third cycle documents is analysed next. This time, all five groups in the cycle display at least one link to group members, indicating that students who coordinated their groups worked with other members on the same topics. Also reflected is the fact that except in Groups c3g2 and c3g4, the group coordinator worked with all the group members on the same topic.

All of the networks except Group 4 exhibit a *strong* connection (i.e. all pairs of nodes are connected). In some groups, notably Group c3g1, all students wrote about the same topics.

The revision history of Group c3g4 demonstrates that although there are four students in the group, only two of them (18 and 19 as shown in Figure 8-9 and Figure 8-10) are involved collaboratively in developing the document, which had 20 revisions in total. Upon checking the revision map shown in Figure 8-10, eleven revisions were edited by 19 and four were wrote by 18, the group coordinator; an

[3] There are two groups: c5g4 and c5g5 that their assigned group coordinators have drop out from the course. Thus, there no particular group coordinators for these groups.

examination of the revision maps finds that 18 and 19 were the only two students working in Section B. They wrote 5 paragraphs together. Therefore, they have a connection in the group coloration network. Unlike 18 and 19, 16 and 20 spent their time writing their own paragraph in Section A and only produce 2 and 1 revisions, respectively.

8.6.3 Qualitative Evaluation

Five students in the course participated in the experiment. Although questionnaire shown in Table 8-4 cannot be statistically evaluated from this small number of participants, the following summary offers some information about the students' perceptions with regard to the usability and the accuracy of the revision maps based on their writing tasks (See the questions in Table 8-4):

- All five students correctly answered the first four Questions (1-4), which implies that given a revision map, students are able to glean information regarding the parts of the document that are revised, when these events occur, and who makes these changes, thus addressing the first research question.
- For Question 5, only one student answers correctly (i.e., by answering "other"). This might be owing to the fact that both sequential and parallel patterns of text edits are present for all groups of students. During a week of work on this assignment, most students wrote their documents very much at the same time. As a result, different parts of documents show different patterns of text edits, both sequential and parallel.
- For Question 6, after a review of their revision maps, two students report the same problem of group coordination -- i.e. team members were unable to schedule a time to work collaboratively, so that students had to write separately on different days and times – as reflected in the map. Similarly, a third student reports that group members initially had difficulty scheduling time to work together, but were later able to work efficiently; according to this author, the revision map shows a corresponding delay in text production until the end of the first week. A fourth student, from a group in which the coordinator leaves during the second week, reports seeing the coordinator's contribution in the beginning of the first week. Another student author reports no problem at all in the group.

- Question 7 asks the students if they find the revision maps to be consistent with their view of what actually happened during their writing processes. All the students respond affirmatively, and describe the revision maps as showing the information that they expected. In one instance, a student reports that one of the team members hardly contributed to the writing and the revision map nicely demonstrates this lack of contribution.
    - Based on the answers to questions 6 and 7, students opine that the revision maps accurately represent the events that took place during the writing process.
- Question 8, which asks the students if they learned something from the revision maps that they did not already know, is answered in a variety of ways. One student expresses surprise at seeing the amount of contribution that she made. Another student relates that the revision map motivates her to keep reminding herself to contribute more than others. And yet another student reports that as she wrote separately in another Google Docs, the revision map did not show her edit history.
- Question 9 inquires if, in retrospect, the students find something that they would have liked to do differently. Most of them agreed that they should have started earlier and completed the summarising of reading materials in the first week, in order to get feedback from their team members.
- Question 10 probes whether the visualisation prompts the students to do something or plan a course of action with regard to their group's work. Three students answer in the affirmative, explaining that they would like to redesign the writing schedule and assign certain times for peer feedback. Two students say "no"; one of them explains that the reason for the negative answer is that regardless of whatever plan was designed, most group members performed their tasks on the weekend before the due date. The other student who answers in the negative relates that some peers did not cooperate or work on their tasks, but merely waited for others to help them.
    - As relevant to questions 8-10, all the information here suggests that student use of revision maps differs considerably among different students.

- In question11, four students agree that they would like to have additional information about their group's writing process that was not represented in the revision map, namely information about other channels of communication such as emails and chat utilities. Interestingly, one of these four students mentions in her answer that the revision map only depicts contributions in terms of text edits performed on revisions, and she would like to see topic-based contributions; on the other hand, another student's answer mentions that the amount of data supplied by the map is quite complete.
- Question 12 solicits an evaluation of the usefulness of the revision maps (by assessing them on a scale from 1 to 7, ranging from 1 as "strongly disagree" to 7 as "strongly agree"). The answers fall between a minimum score of 4 and the maximum score of 7, with an average score of 5.8. Figure 8-13 depicts the scores of five students. In other words, most students believe that the revision maps were quite useful for their group writing tasks.

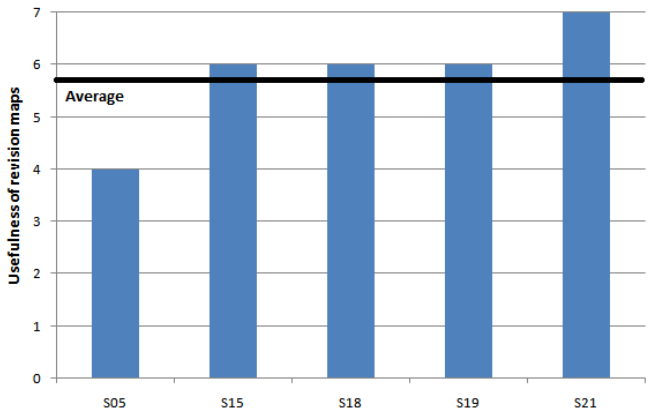


Figure 8-13. The usefulness of revision maps - ranging from 1 as "strongly disagree" to 7 as "strongly agree"

8.7 Summary

This thesis contributes three new types of visualisations (along with their underlying techniques) for analysing the writing processes of jointly authored documents. The first type, the revision map, provides a visual representation of text edits made by students at the paragraph level over a period of time. The second type, the topic evolution chart, displays an image that illustrates how topics unfold as the writing progresses. The third type, the topic-based collaboration network, exhibits the

connections between joint authors who write about the same topics during the process. The proposed techniques used to constructing these visualisations are successfully validated against a synthetic dataset. In addition, this thesis presents a case study using real documents written collaboratively by graduate students that demonstrates the use of the new visualisations in analysing writing processes. The case study inclusion offsets the insufficient amount of information derived from simple statistics and limited access to the final documents by contributing a satisfactory amount of ancillary data that sheds further light on the writing process investigation.

CHAPTER 9

DISCUSSION, FUTURE WORK, AND CONLCUSION

The aim of my research is to create a toolbox, consisting of a set of algorithms and visualisations, that allows the user to better understand and/or improve a collaborative writing process. By applying this toolbox, we can gain insight into the development of collaborative writing as it is taking place, and this insight is then used to give feedback to student authors and/or to education researchers and teachers as the writing tasks are being performed as well as after the document is finished. This toolbox includes the following:

- A method for defining types of text edits that occur; this method is based on the theories of cognitive models of writing processes, the taxonomy of collaborative writing activities, and model for analysing revisions.
- A method for automatically identifying writing activities; this method is based on the text edits that occur during the writing process and other text features such as text structure, number of words, number of sentences, number of paragraphs, cohesion change, and topic overlaps.

- Methods for extracting process models of collaborative writing processes based on text edits and writing activities.
- Methods for visualising a snapshot and creating a chart of paragraph evolution, topic evolution, and topic-based collaboration in writing processes.

Although the techniques described in this research are based on revision histories of Google Docs as event logs, they can be fine-tuned and applied to event logs captured in other writing environments, such as key-stroke logging tools like InputLog (Leijten & Van Waes, 2006) or version controlled Wiki environments like Wikipedia (Wikipedia, 2013).

This thesis research is the first work to systematically propose the coding scheme for types of text edits based on the models for analyzing revisions proposed by Faigley and Witte (1981) and later extended by Boiarsky (1984). In 2010, these types of text edits were introduced and utilised to automatically identify collaborative writing activities (Southavilay et al., 2010). Other researchers, especially Daxenberger and Gurevych (2012), had used similar categories of text edits in their work to automatically classify text change operations performed on Wikipedia articles. Because of the nature of these articles, there were many types of edits: text based edits; Wikipedia policy, such as vandalism and reversion; and surface edits such as those affecting mark-up segments. Nevertheless, the text-based edits in the work of Daxenberger and Gurevych (2012) are similar to text edits used in this thesis

This chapter first validates the work in this research in other domains, then addresses the limitations of the approach used in developing the toolbox. An explanation of implementation then follows, and the thesis concludes with a discussion about future work in this area.

9.1 Validation of this thesis work in other domains

The techniques described in this thesis were applied in other domains, such as in extracting process models of students' model-based inquiry and problem-solving strategies. Relevantly, a description is provided here of another area of my work, which involves applying the Hidden Markov Model (HMM) and Heuristic Miner to discover patterns of student interaction with agent-based computational models such NetLogo models (Wilensky, 2013).

The work of Thompson et al. (2011) chronicles the methodological experiences in capturing and analysing student learning processes and patterns in three different cases. Agent-based models built in NetLogo were used for learning in two of the cases, and a virtual world was used in the third one. First, students interacted in real time for relatively short periods. Second, they interacted both with each other and with interactive software tools that dynamically shaped, and were shaped by, their learning process. The work of (Thompson et al., 2011) builds upon and integrates process analytic approaches of dynamically captured video, as well as computer screen activity and automatic e-learning process analysis techniques.

The first two cases identify areas in which analysis by hand of small amounts of data produces findings of initial interest. My contribution takes place in the third case, and consists of using an automatic pattern discovery technique based on HMM to extract the problem-solving behaviours of students. This work demonstrates that process analyses such as the use of HMM allow education researchers information that helps them to understand how students *learn* in computer-supported collaborative learning (CSCL) environments and what kind of learning processes various combinations of particular collaborative pedagogies and computer supported learning environments can afford.

A process analysis technique described in this thesis was also applied in a design-based research project that investigates the learning of scientific knowledge about climate change through agent-based computational models (Kelly et al., 2012; Markauskaite et al., 2012). This design experiment uses two NetLogo models and problem-based learning materials developed in partnership between this project's researchers and a high school science teacher. In the study, three classes of science students in year nine are divided into two groups, based upon the different levels of structure that are provided during learning activities with the models. Unlike the study mentioned earlier in which screen capture is used and transcribed to event logs, in this study, sequences of student interactions with the NetLogo models are automatically recorded in log files. Based on the sequences, I uses HMM to extract patterns of students' interactions with the models, as well as to identify effective and inefficient behaviours for learning with agent-based computer models. The results indicate that successful learners adopt deeper and more systematic model exploration strategies than less successful learners (Markauskaite et al., 2012).

I has also created a multilevel data pre-processing approach to use in combination with process mining algorithms such as Heuristic Miner (Weijters & Ribeiro, 2010; Weijters et al., 2006) for investigating students' model-based inquiry strategies. A traditional approach in exploring learning processes is to use event logs of students' interactions with computer software as input to process mining algorithms; however, processes of students' interactions with computational models tend to be very flexible, unstructured, and composed from large numbers of fine-grained technical events captured in the logs. As a consequence, the identified patterns from the event-sequences can be hard to interpret and may be too far removed from the intentions of the students. For this reason, it is necessary to employ the heuristic technique described in Chapter 6 to transform sequences of technical events into sequences of more abstract actions and semantic activities. These sequences of actions and activities are then used for discovering patterns of students' interactions with computational models. My approach automatically segmented sequences of events and clustering them into actions, then classifies the discovered actions into higher semantic level activities using a heuristic set. A notion of "bag of events", analogous to the "bag of words" concept in text mining was used to cluster the sequences of events into actions. The pilot study demonstrates the usefulness of multilevel abstraction for extracting and exploring the main characteristics that relate to how learners interact with computational models. The study shows that each abstraction level helps to identify distinct characteristics of students' interaction.

All the material reported above constitutes proof of the successful application of the contributions in this thesis to other domains. Limitations of the techniques used herein are the subject of the following section.

9.2 Limitations

9.2.1 Google Docs API Limitations

Google Document List API is used extensively in this thesis to retrieve revisions and revision histories for documents written by groups of students in the case studies and experiments discussed so far. It is evident, however, that certain technical adjustments need to be made. For example, although several authors can make

changes to the same content at almost the same time, the new version of GD API (i.e. Google Document List API 3.0) only gives one main author for each revision. For this reason, it became necessary to manually record the list of authors for each revision by using the revision history function on the web interface of Google Docs, described in Section 5.2. Although this could be done offline, during the user study as discussed in the previous chapter, it was not possible to produce topic-based author collaboration networks to use as feedback for students in real time as they were performing their writing tasks. In addition, there are several text edits performed on each revision. The authorship information of each edit can not be obtained automatically.

9.2.2 Coding and Heuristic Limitations

There is a limitation in the level of text edit coding. As pointed out in the first chapter, the context of this research includes jointly authored documents collaboratively written by groups of students in the form of several writing sessions. Any incorrect information or claim written on the document had to be resolved among group members by online or face-to-face discussion. Unlike the writing in the form of Wiki, the information of what information was incorrect and when it was corrected was not posted and/or recorded in our study and experiment for further analysis. Therefore, the coding of text edits did not include deleting erroneous claim posted by other group members, clarifying, providing illustration/examples, inserting statements to denote the limitation of a given claim, etc. In addition, this research did not examine how action sequences affected the quality (e.g. relevance, accuracy, veracity) of the revised content/idea/claim at the sentence level. Instead, the quality of the whole document was considered based on the assessment of its final version.

Overall, the heuristic achieves higher accuracy than the baseline, as described in Chapter 6. Among the five types of writing activities, however, there is a problem with detecting editing activities. This is because the heuristic only considers surface edits as editing activities, so that other editing activities (such as grammatical corrections) are not detected. In order to detect edit types pertaining to grammatical corrections and spelling -- thus improving the accuracy of the heuristic -- natural language process techniques can be employed and will be described later in the section that outlines future work.

In addition, for the validation of the heuristic as described in Section 6.3, manual tagging was performed by one rater. In future studies where activities are greater in number and are more difficult to distinguish, this type of manual tagging should be conducted by two human raters to see what percentage of their tags are in agreement. If agreement is poor, validating the heuristic's tags with the human tags is questionable.

9.2.3 Hidden Markov Model Limitations

The algorithm for constructing the Hidden Markov Model (HMM) suffers from the problem of local maxima. This thesis follows the work developed by Jeong et al. (2010) and executes the algorithm one hundred times with random initialisations (by sampling the initial parameter values from uniform distributions). All of these executions converge to the same configuration. A better solution is needed to execute this algorithm in real time and provide feedback to students as they write.

Each HMM also has a space complexity problem in storing all parameters (i.e. a transition matrix, emission matrix, and initial matrices) during the training process to construct the HMM models. The number of *free* parameters of an HMM can be calculated as $N + N(N-1) + N(M-1)$, where M is the number of observations (i.e. the number of text edits) and N is the number of hidden states, which is also estimated during the training process. Because the training process has to be executed a hundred times to estimate the parameters, including the number of hidden states, it requires a large amount of storage space for these parameters.

The transitional state diagrams obtained from Hidden MMs are difficult to interpret intuitively, as described by students participating in the pilot study reported in Chapter 7. Hidden MMs do not show the necessary number of occurrences of writing activities and states for obtaining a general statistical overview of those activities and states. In order to depict these transitional state diagrams to students or authors, the models must depict extra information, such as the size of states or writing activities, which represents the number words performed in those states or writing activities.

9.2.4 Heuristic Mining Limitations

Heuristic Miner (HM) can handle noisy and incomplete event logs (i.e. process instances) and process instances (i.e. sequences of writing activities and states) of any length. In addition, the output of Heuristic Miner – dependency diagrams or heuristic nets -- provides not only the dependency values between activities and states but also the number of their occurrences, along with highlighting strong dependencies, thus making it easier to interpret and extract patterns of writing processes, comparing to Hidden Markov Models.

The algorithm used in HM, however, suffers from limitations. As discussed in the previous section, this research uses the default setting of the three threshold parameters of HM -- the *dependency threshold*, the *positive observation threshold*, and the *relative to best threshold* -- in order to obtain fully connected dependency graphs that represent writing process models. Most event logs, however, consist of many different kinds of activities, some of which occur infrequently (especially brainstorming and outlining); and since the algorithm had to discover the causal dependencies of all events, the discovered models included a high number of connections with low dependency values. In addition, HM does not guarantee that the obtained models can replay all cases in the event logs (van der Aalst, 2011), which results in some discovered models that are typically underfitting. An underfitting model over-generalises the items seen in the log and allows for more behaviours, which may not occur at all in the log. For this reason, when interpreting and comparing models, the fitness is computed (i.e. the proportion of activities in the log that can be explained by the process model) and only dominant dependencies are used between activities to extract emerging patterns. These identified emerging patterns represent a portion of the large number of possible behaviours. The technique of measuring fitness between log and model is commonly used in process mining research (van der Aalst, 2011; Weijters & Ribeiro, 2010). Although the fitness indicates how much of the observed behaviour in the event log is captured by the process model, it does not indicate how much of the behaviour not observed in the event log can be recognised by using the process model. In other words, there should be a measure to indicate to what degree the process model permits extra allowed behaviour.

9.2.5 Visualisation Limitations

Because off-the-shelf process models (representations) obtained from process mining algorithms like Hidden MMs are intuitively difficult to interpret as discussed above, this thesis has developed three types of writing process visualisation. Each type of the visualisation has its own uniqueness, in which they depict different types of information and compromise each other to provide feedback to authors in order to better understand writing processes. They can not be used individually. Therefore, this thesis did not compare one another and evaluate each of them against process models like MM diagrams in order to make explicit advantages and disadvantages of using each individual type of visualisation.

However, the user study that was conducted to provide the three visualisations to authors during their writing tasks (described in previous chapters) suffers from certain technical limitations. As discussed in subsection 9.2.1, the unavailability of authorship information records affected the experiment on constructing topic-based collaboration networks in real time. In addition, it was necessary to infer the number of topics in order to extract topics and topic evolution charts. As explained in the prior chapter, Diff Latent Dirichlet Allocation (Diff LDA) and Diff Author-Topic Models (Diff ATM) use delta documents created by identifying the text edits (added and deleted words and paragraphs) of all revisions; but in this case, the number of topics for each document was computed at the end of the writing process by using all revisions of the document. The two visualisations were therefore only shown to the instructor of the course at the end of the writing.

In topic evolution charts, each topic consists of several key words. It is important to note that each key word can belong to more than one topic, because a topic is a mix-membership of key words; consequently, analysts who are not experts in the writing field found it quite difficult to interpret topics during the analysis of topic evolution charts. Open research still exists with regard to analysing the word distribution of topics and the accurate method for comparing these distributions. It is possible to find the similarity of two topics using certain kinds of measures such as DS convergence; but it is not clear if this measure actually works, because a corpus of interest may have thousands of terms, and each term has its own membership for particular topics (contributes to topic differently).

For topic-based collaboration networks, the links between two nodes can be made directional links. The connection indicates that the two authors corresponding to the linked nodes have written about the same topics. We can identify these topics and analyse them by using the topic evolution charts. Based on the emergence of the topics, we can see who first created the topics, thus leading the collaboration. The directional link is created using this information, represented as an arrow-link that starts at the node of the initiator of the common topics and goes in the direction of the linked authors to connect the corresponding collaborators.

9.3 Implementation of the Toolbox

WriteProc, the framework for retrieving revisions and revision histories from Google Docs (as explained in Chapter 4) is developed in Java (1.6) using Google Document List API (V3). The content of revisions and revision histories are stored in a relational database. The content texts of revisions are indexed using Apache Lucene (Lucene, 2013). A graph database, Neo4J, is used in this thesis, in which each paragraph in a revision is represented by a node and linked to its corresponding paragraph node in the previous revision. Each node has several attributes, including a text edit performed on this paragraph (node), the number of words affected by the text edit, and a relative location of this paragraph (node) on the document. Using the graph database, text edits performed on particular paragraphs are extracted by traversing a chain that links all revisions of the paragraphs. These sequences of edits are then used in visualising revision maps, as delineated in previous sections of this thesis.

All algorithms -- except Heuristic Miner, which is implemented in ProM (ProM, 2013), and topic modelling algorithms: DiffLDA and DiffATM, which are used for extracting topic evolution charts and topic-based collaboration networks -- are created in Java. The current DiffATM is developed by using topic model toolbox implemented in MATLAB for this thesis; however, it can be transferred into Java codes quite easily by using MALLET library (McCallum, 2002). Therefore, the framework and algorithms are developed to extract process models and provide their process representations as visualisations (i.e the three types previously explained) in real time. The code is provided as an open source.

9.4 Future Work

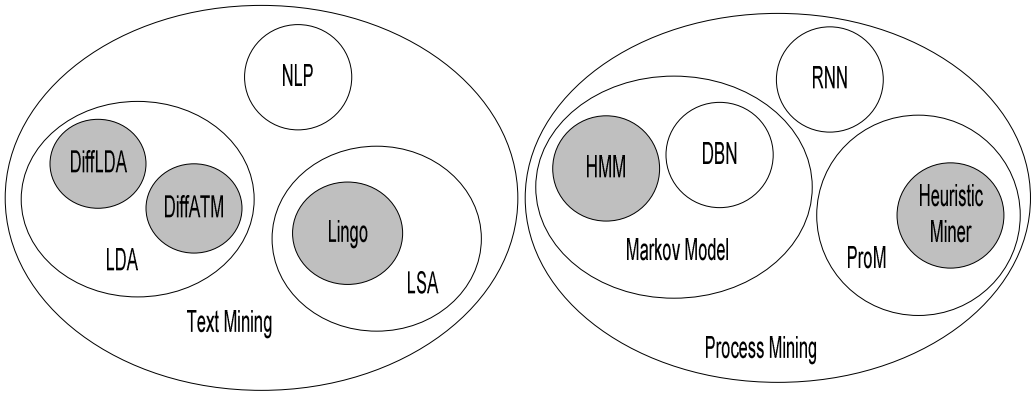


Figure 9-1. Summary of algorithms adapted and created in the toolbox, and algorithms which can be used for improving the toolbox in the future.

Abbreviation: Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), Diff Latent Semantic Analysis (DiffLSA), Diff Author-Topic Model (DiffATM), Natural Language Processing (NLP), Hidden Markov Model (HMM), Dynamic Bayesian Network (DBN), and an open source process mining framework, (ProM).

In this thesis, in order to build a toolbox for automatically extracting process models of writing processes and providing visualisations that illustrate aspects of collaborative writing, several algorithms are created and adapted from two main fields: text mining and process mining.

Figure 9-1 summarises all the algorithms in the toolbox. The oval depicts an algorithm developed in the two main fields. The grey ovals show the algorithms used and created in this thesis. Text mining algorithms are used extensively in the heuristic to automatically identify writing activities. Latent Semantic Analysis (LSA) (Landauer & Dumais, 1997; Landauer et al., 2007) is used to compute the text cohesion of each revision and to detect cohesion changes during the writing process. The LSA-based document clustering algorithm Lingo (Osinski & Weiss, 2005) is used for extracting topics and calculating topic overlap. A probabilistic graphical modelling technique like Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is also employed. DiffLDA (Thomas, 2011; Thomas et al., 2010a) is adapted to extract topics for topic evolution charts and Diff Author-Topic Model (DiffATM) is created to construct topic-based collaboration networks. From the field of process mining, two techniques are used for extracting writing process models. The first one is based on the process mining framework ProM (ProM, 2013), like Heuristic Miner (Weijters & Ribeiro, 2010; Weijters et al., 2006). The other one is based on Markov models,

like Hidden Markov Model (HMM) (Rabiner, 1989). Obviously, several techniques exist that can be integrated into the toolbox as shown in Figure 9-1. For instance, Dynamic Bayesian Network can be used to model collaborative writing processes and extract patterns of collaborative writing activities. Natural Language Processing (NLP) techniques can be used to improve the heuristic for automatically identifying collaborative writing activities. The following subsections are concerned only with techniques that can be used to improve the toolbox in the future.

9.4.1 Improving the Heuristic with Natural Language Processing

In this thesis, the technique used in the heuristic for automatically identifying collaborative writing activities is based purely on text mining methods. In the future, the technique from this thesis can incorporate those used in natural language processing (NLP). Recently, Bronner and Monz (2012) proposed a method for automatically distinguishing between factual and fluency edits performed on Wikipedia articles. Factual edits alter meaning, whereas fluency edits improve style or readability. The Bronner and Monz approach was based on supervised machine learning using language model probabilities, string similarity measured over different representations of user edits, comparison of part-of-speech tags and named entities, and a set of adaptive features extracted from large amounts of unlabelled user edits. Although their method requires a huge amount of labelled data which can be acquired from Wikipedia, it achieves high classification accuracy. Other techniques of NLP that may be helpful to improve the accuracy and the effectiveness of automatically identifying writing activities include work in recognising text entailments, identifying paraphrases, and simplifying sentences. For instance, if a sentence in the current revision can be identified as a paraphrase of the same sentence in the previous revision, the text edit that transforms the sentence can be designated as a revising activity; but when applying these techniques to collaborative writing processes, a problem arises in ensuring that the sentence in the current revision is the same sentence in the previous revision. Nevertheless, natural language processing technique appears a promising avenue for automatically identifying writing activities.

The aforementioned natural language processing techniques for classifying text edits, especially those of (Bronner & Monz, 2012), are based on several features of

the writing process, chosen from hundreds of possible features. In Wikipedia text edit classification, researchers use not only text-based features, similar to those in this work, but also other types of features such as surface, vandalism, and revert. Surface edits consist of edits affecting mark-up segments. Vandalism edits include edits deliberately compromising Wikipedia's integrity and revert edits representing edits restoring a previous stage of a page. An open research question still exists that asks what features should be included in the classification of text edits and how the features should be weighted. Recently, neural network technique has reappeared, using neural networks to learn features from a set of inputs and labelled outputs. Interestingly, Sutskever et al. (2011) uses recurrent neural networks to generate text, character by character, given an initial set of words or phrases. Although the technique requires a great deal of resources for computation, in the future, when powerful computers are easily available and accessible, it will be interesting to see if neural network technique can be incorporated to improve the automatic detection of collaborative writing activities.

9.4.2 Improving Topic Extraction

In order to extract topics for computing topic overlap, the heuristic prefers the document clustering algorithm Lingo to topic modelling or Latent Dirichlet Allocation. The reason for this preference is that unlike topic modelling, which outputs a topic as a related group of words (thus creating the interpretation difficulty previously noted), Lingo first finds the label for topics before performing the clustering task. In addition, Lingo uses Latent Semantic Analysis, which is also used to compute cohesion. Therefore, both topic overlap and cohesion changes can be computed in one operation every time a new revision is produced. Recently, Liu et al. (2009) proposed a new technique for measuring the cohesion of classes in software repositories, based on the analysis of latent topics embedded in comments and identifiers in source codes. This proposed approach, named Maximal Weighted Entropy, utilises the topic modelling technique and information entropy measures to quantitatively evaluate the cohesion of classes in software. Interestingly, based on this technique and the topic evolution, the topics and cohesion measure of a revision can be extracted by applying LDA only once. The only drawback of the approach is the time factor, as the inference of the number of topics and the topic membership

can take a significant amount of time to accomplish (see Subsection 9.2.5). Again, at a point in the future when more advanced technology exists, topic modelling is likely to be a viable method for extracting topics and cohesion because of the availability of powerful computer resources and the improvement of inference techniques that will speed up the computation.

9.4.3 Creating Interactive Visualisations

All the visualisation types proposed in this thesis are created as prototypes for proof of concept. The revision map provided to students as they were engaged in writing their documents was intended to serve the purpose of the user study (as described in Chapter 8). Obviously, more interactive types of visualisations can be developed as well. For example, revision maps can be created by using a Javascript library of D3.js (Bostock, 2012). This library can manipulate documents based on data. D3 can bring data to life using HTML, SVG and CSS. The characteristic D3 emphasis on web standards offers the full capabilities of modern browsers without ties to a proprietary framework, combining powerful visualisation components and a data-driven approach to DOM manipulation.

As another example, revision maps can be created to have a split-attention effect: to understand data, the teacher or researcher can juxtapose two windows. One window shows the map, and the other window displays the text. We can then go back and forth between the two windows to connect data with the contents of a paragraph. With a bit of creativity, the visualisation could be integrated into the text itself, by playing with multiple parameters such as the colour of the text, the colour of the background and some type of bar chart placed vertically in the margins, making the data more useful.

9.5 Conclusion

Collaboration and particularly collaborative writing is an increasingly essential skill needed for the workplace and for use in education. Until recently, most of the focus of research in this area has been placed on the final writing product, rather than on the writing process. Investigations into the development of ideas and concepts as they unfold during the course of the collaborative process can be used to improve not only the quality of the final documents created jointly, but more importantly, the

writing skills of the authors. The process of writing consists of steps of writing activities. These steps of writing activities can be considered as sequence patterns comprising both time events and the semantics of changes made during those steps. In order to obtain insight into the manner in which students undertake collaborative writing tasks, two techniques can be combined: process mining, which focuses on extracting process-related knowledge from event logs recorded by an information system; and semantic analysis, which focuses on extracting knowledge about what the student wrote (or edited). This thesis presents the development of a data mining toolbox consisting of both process mining and text mining algorithms, as well as visualisations for extracting writing process models and analysing collaborative writing processes.

The work of this thesis constitutes a big step toward accomplishing the automatic extraction of process models and visualisations with the purpose of gaining a better understanding about how students work and create their documents collaboratively. The ultimate aim of the efforts in this research is to support the collaborative writing process by providing these process models and visualisations as feedback to groups of students who are working together on a document. This feedback then enables individual students to become aware of the group's writing activities, so that all authors can work more efficiently and effectively. This same feedback also provides support for teachers, allowing them to monitor groups more skilfully by supplying them with a tool for detecting problems early in the writing process.

APPENDIX A EXAMPLES OF REVISION HISTORIES AND TEXT EDITS

A.1 An Example of Revision Histories

Figure A-1 shows an example of revisions histories. Each row consists of revision ID, timestamp, and author IDs of a revision. Note that a revision may have one or more authors associated with it, for an example, see Figure A-2. Google API (version 3.0) only provides the first author ID in a revision history record. The remaining author IDs were manually identified by using the web-based interface of revision history, as described in Chapter 5.

Revision ID	Timestamp	Author IDs
2	5/20/2012 7:24	admin
101	5/26/2012 7:44	S05
132	5/27/2012 0:43	S05
144	5/27/2012 13:11	S05;S04;
147	5/27/2012 22:31	S05;S03;
196	5/28/2012 2:56	S01;S05;
269	5/28/2012 3:45	S01
388	5/28/2012 4:57	S01
428	5/28/2012 6:30	S01;S03;
524	5/28/2012 6:45	S01
531	5/28/2012 8:39	S01;S03;
612	5/28/2012 11:39	S01;S03;
1672	5/29/2012 4:50	S01
1674	5/29/2012 7:21	S01;S05;
1675	5/29/2012 9:31	S01;S05;
1791	5/29/2012 10:29	S01;S03;
1793	5/30/2012 3:43	S02;S01;
2578	5/31/2012 4:47	S01
2637	5/31/2012 7:07	S02;S01;
2732	5/31/2012 7:45	S02
2780	5/31/2012 9:55	S01;S05;
2782	5/31/2012 10:40	S01;S05;
2806	5/31/2012 10:53	S01
2881	5/31/2012 11:08	S01;S03;
3026	5/31/2012 11:21	S01;S03;
3082	5/31/2012 11:29	S01
3094	6/3/2012 10:47	S01;S04;
3142	6/4/2012 7:01	S01;S05;
3143	6/4/2012 8:50	S01;S05;S03
3203	6/4/2012 8:58	S01;S05;
3251	6/5/2012 7:42	S01;S05;
3444	6/5/2012 11:10	S01;S05;S03
….		

Figure A-1. An example of revision histories.

A.2 An example of Multiple Text Edits

Figure A-2 show two consecutive real revisions of a document with three text edits performed on the left revision, resulting in the right revision.

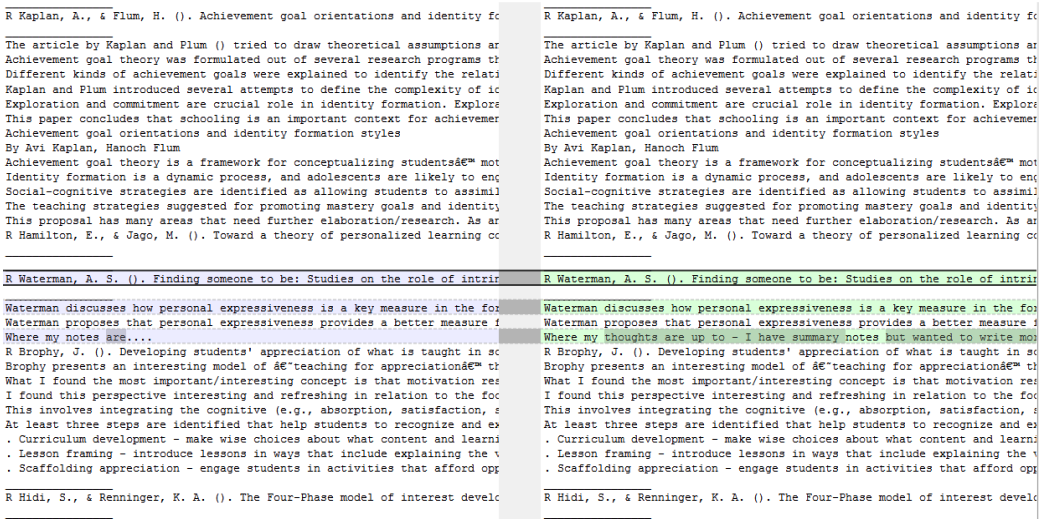


Figure A-2. Two consecutive revisions showing three text edits.

APPENDIX B TEXT DIFFERENCING PROCEDURE

The text comparison utility described in Chapter 6 uses a text differencing algorithm which is based on two levels of text edits: paragraph and word. At the *paragraph* level, the algorithm detects six types of text edits: inserting (C5), deleting (C6), moving (C2), changing (C8), merging (C3), and splitting (C4) paragraphs. At the *word* level, the text differencing algorithm identifies five types of text edits: inserting (C8.1), deleting (C8.2), moving (C8.4), replacing (C8.5), and appending (C8.3) words.

In order to identify text edits, paragraph differencing was first performed to detect the types of edits that transform paragraphs from old revisions to those in the current one. This operation successfully distinguished which paragraphs had been inserted, deleted, moved and changed. After that, word differencing was performed on all changed paragraphs in order to detect all word edits as well as which paragraphs have been merged and distributed. These two differencing algorithms are explained below.

B.1 Paragraph Differencing:

The paragraph level differencing is based on Longest Common Subsequence (LCS)(Hunt & McIlroy, 1976), a text differencing method implemented with the standard Unix *diff* utility. The texts of an old and a new revision are used as input to produce a difference statement in terms of the insertion, deletion and replacement events. The algorithm provides a diff record of triplets (*Opt,* $Para_{old}$*,* $Para_{new}$):

- *Opt* is either an insertion, or deletion, or replacement.
- $Para_{old}$ consists of a starting position (*OStart*) and an ending position (*OEnd*) of original paragraphs in the old revision (if *Opt* is an insertion, then *OEnd* is 0).
- $Para_{new}$ consists of a starting position (*NStart*) and an ending position (*NEnd*) of changed paragraphs in the new revision (if *Opt* is a deletion, then *NEnd* is 0).

Figure B-1 depicts an example of the evolution of paragraphs during the document writing process. There are 7 revisions; each revision has a diff record associated with it. Each rectangle represents a paragraph. Each circle shows a text

edit. The diff utility can identify three main types of text change operations: inserting (a), deleting (d) and changing (c) paragraphs, shown in green, blue, and red circles, respectively. The green and blue circles are C5 and C6 text edits, which are detected by the paragraph differencing algorithm. In order to select other types of paragraph edits (i.e. merging and splitting paragraphs) and word edits, a word differencing algorithm is used.

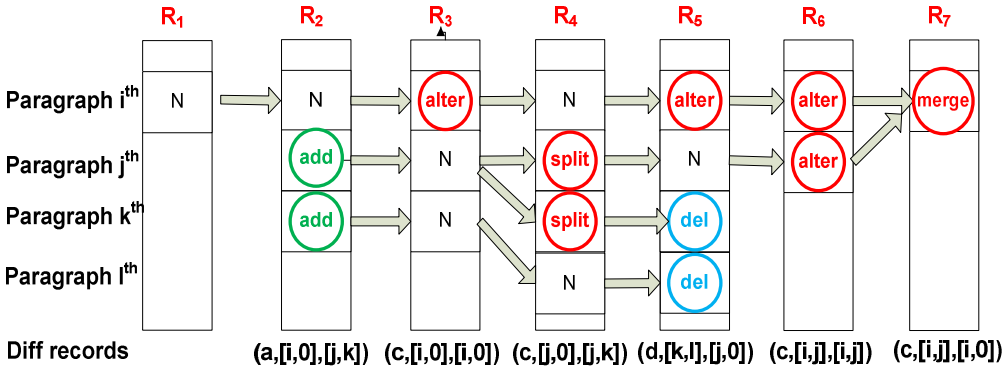


Figure B-1. An example of text edits performed on 7 revisions. A revision has a diff record associated with it. A rectangle represents a paragraph. Each circle shows a text edit. Red and blue edits are detected by paragraph differencing, whereas red ones are identified by a word differencing algorithm.

However, this kind of diff record does not report text movements (C2) explicitly, i.e. a portion of text that now located up or down from its previous location. Paragraph movement is detected by checking whether a formerly deleted paragraph is now included elsewhere within the new revision. A paragraph split is detected when any replaced paragraph in the new revision is formed from one paragraph in the old revision (i.e. *OEnd* is 0 and *NEnd* is not 0) and the words in the new paragraph match the words of the potential original paragraph in the old revision. If the match value is higher than a predefined threshold, the paragraph edition is designated as a distribution; otherwise, it is considered to be replacement and insertion of new paragraphs. Similarly, paragraph merges between old and new revisions are discovered when replaced paragraphs in the new version are formed from several paragraphs in the old one (i.e. *OEnd* is not 0 and *NEnd* is 0) and the words in the new revision match with those of the potential original paragraphs in the old revision. The matching mechanism is performed by the word level differencing algorithm described below. For all other reported replacements, word level differencing is performed by comparing the text of each replaced paragraph in the

new revision to the text of its corresponding original paragraph in the old revision, as described next.

B.2 Word Differencing:

Word differencing uses Myers' algorithm (Myers, 1986), which inputs two blocks of plain text-- old and new paragraphs -- and efficiently compares them to disclose words that are equal, inserted, and deleted. Word differencing was specifically created to produce a word diff list consisting of a sequence of words that are equal to, inserted to, and deleted from the existing paragraphs. Figure B-2 depicts one example of word edits performed on an existing paragraph and the corresponding word diff list.

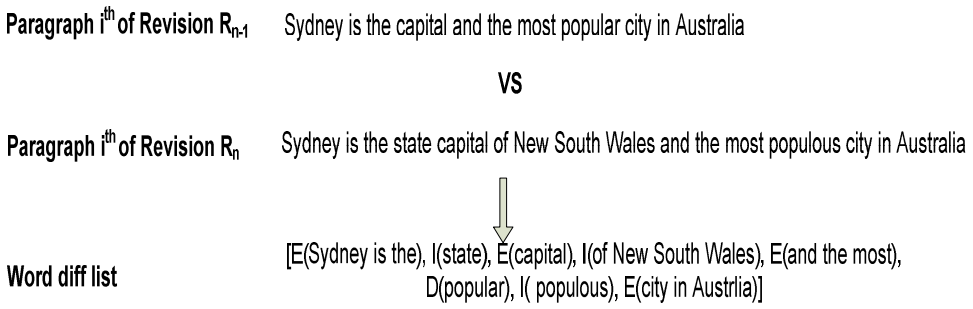


Figure B-2. An example of word edits performed on an existing paragraph.

Myer's algorithm was used in this thesis for word differencing to identify text edits at the word level within replaced paragraphs (i.e. changing existing paragraphs), and computing word matching rates to detect whether paragraphs are merged or split. The matching rate is inspired by the matching rate proposed by (Fong & Biuk-Aghai, 2010).

If the following is denoted:

lo_i as the number of words in the i^{th} paragraph of the old version,

ln_j as the number of words in the j^{th} paragraph of the new version,

$lc_{i,,j}$ as the number of common words between the above two paragraphs,

$m_{i,j}$ as the word matching rate between the above two paragraphs,

then the matching rate can be computed by the formula below:

$$m_{i,j} = \frac{2 \times lc_{i,j}}{lo_i + ln_j}$$

lo_i, and ln_j are easily obtained. Mayer's algorithm is used to compute $lc_{i,j}$. Since the number of common words will never exceed the two numbers of words in both

paragraphs, the upper boundary of the matching rate is 100%, which occurs when two paragraphs are identical. The lower boundary is 0%, which occurs when two paragraphs have no common words.

Using the above formula, text distribution and consolidation can be achieved by comparing the matching rate to a certain threshold. In order to detect text distribution of the i^{th} paragraph in the old revision to the paragraphs from the j^{th} to the $j+k^{th}$ of the new revision, the following computation applies:

$$m_{i,j\ to\ j+k} = \frac{2 \times (lc_{i,j} + lc_{i,j+1} + \ldots + lc_{i,j+k})}{lo_i + ln_j + ln_{j+1} + \ldots + ln_{j+k}}$$

If $m_{i,j\ to\ j+k} > 40\%$, as the same threshold used in Fong and Biuk-Aghai (Fong & Biuk-Aghai, 2010), the utility detects the existence of a text distribution of the i^{th} paragraph in the old revision to the paragraphs from j^{th} to $j+k^{th}$ of the new revision. Otherwise, the utility infers a replacement of the i^{th} paragraph in the old revision to become the j^{th} in the new revision and an insertion of $j+1^{th}$ to $j+k^{th}$ paragraphs into the new revision.

Similar computation and comparison are performed for detecting the consolidation of paragraphs from the i^{th} to the $i+k^{th}$ of the old revision to the j^{th} paragraph of the new revision. The matching rate for this case can be calculated by:

$$m_{i\ to\ i+k,j} = \frac{2 \times (lc_{i,j} + lc_{i+1,j} + \ldots + lc_{i+k,j})}{lo_i + lo_{i+1} + \ldots + lo_{i+k} + ln_j}$$

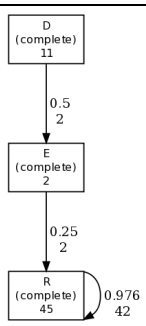
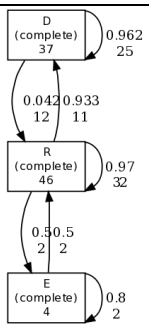
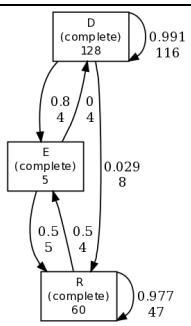
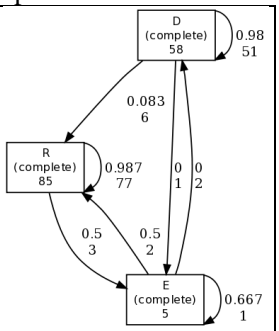
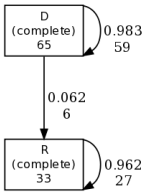
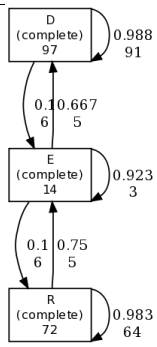
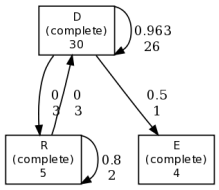
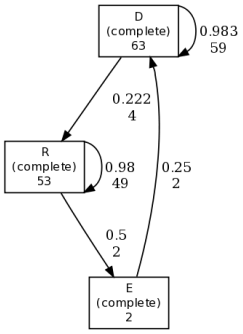
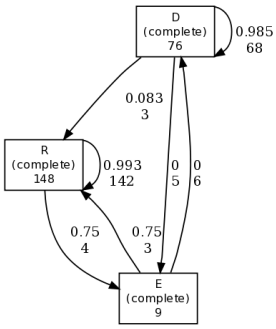
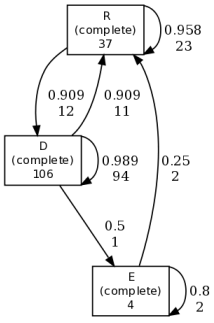
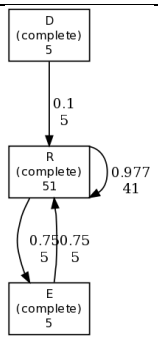
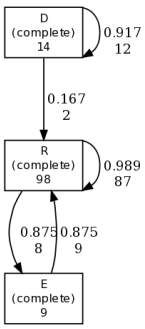
If $m_{i\ to\ i+k,j} > 40\%$, as the same threshold used in Fong and Biuk-Aghai (Fong & Biuk-Aghai, 2010), an existing text consolidation of paragraphs from the i^{th} to the $i+k^{th}$ of the old revision to the j^{th} paragraph of the new revision is detected; otherwise, a process of replacement of the i^{th} to the $i+k^{th}$ paragraphs (i.e. deletion of those paragraphs and insertion of the j^{th} paragraph) is inferred.

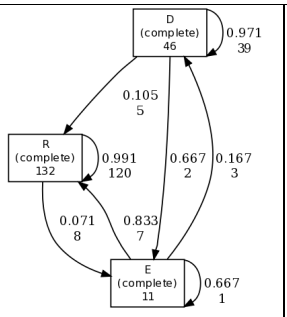
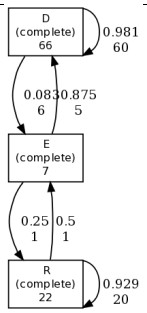
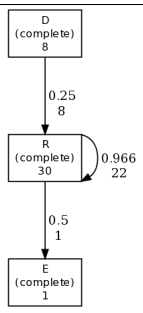
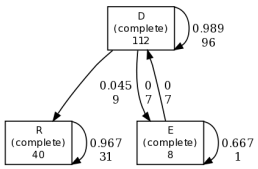
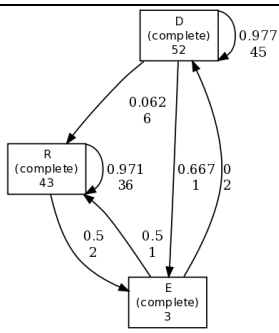
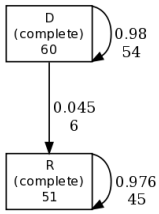
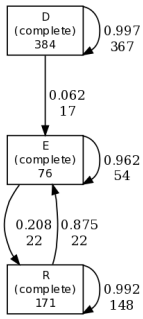
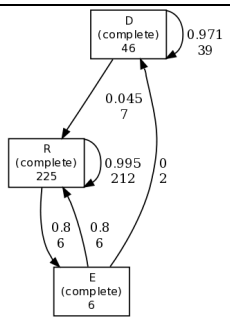
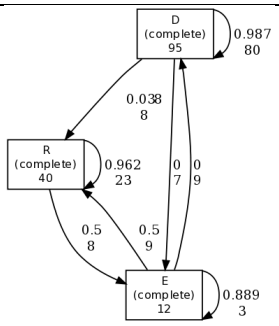
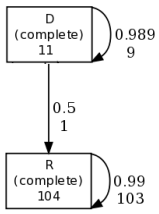
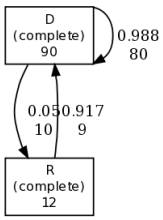
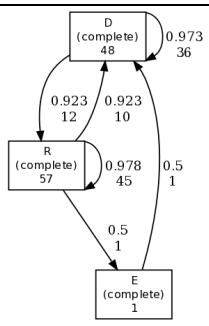
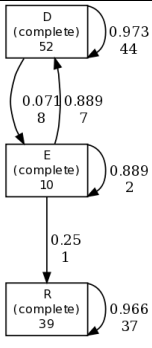
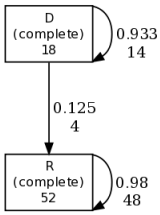
To detect the word edits performed on existing paragraphs, the text comparison utility first counts the number of inserted, deleted, and equal words between the old and new paragraphs and then checks the positions of those words using the word diff lists (Figure B-2 shows word edits and the corresponding word diff list). For instance, if the new paragraphs reflect words added to previously equal paragraphs, the text edit is identified as “appending words”. If the new paragraphs consists of previously equal paragraphs to which some words were inserted followed by equal words toward the end, the text edit is identified as “inserting words”. If some words

in the old paragraphs were deleted and those same words inserted in the new paragraphs at different locations, the text edit is identified as “moving words”.

APPENDIX C DEPENDENCY DIAGRAMS OF DATASET A

Figure C-1. Process models, as dependency diagrams of documents written by 26 groups of two students. The final marks (out of 100) of individual groups are shown in parenthesis. The fitness of each model is the decimal number below the group number.

<table>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Group 01(64)
0.8833</td>
<td>Group 02 (54)
0.764</td>
<td>Group 03 (91)
77.20</td>
<td>Group 04 (68)
0.9</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Group 05 (58)
0.87</td>
<td>Group 06 (65)
0.8</td>
<td>Group 07 (91)
0.78</td>
<td>Group 08 (63)
0.925</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Group 09 (61)
0.8468</td>
<td>Group 10 (38)
0.93288</td>
<td>Group 11 (75)
0.7778</td>
<td>Group 12 (68)
0.7805</td>
</tr>
</table>

			
Group 13 (68) 0.806	Group 14 (56) 0.7113	Group 15 (39) 0.878	Group 16 (85) 0.642
			
Group 17 (68) 0.88	Group 18 (78) 0.814	Group 19 (74) 0.8073	Group 20 (65) 0.9032
			
Group 21 (61) 0.7383	Group22 (76) 0.9909	Group 23 (90) 0.7596	Group 24 (45) 0.7963
			
Group 25 (68) 0.7087	Group 26 (65) 0.875		

APPENDIX D FOUR REVISION MAPS OF A PROTOTYPE EXPERIMENT

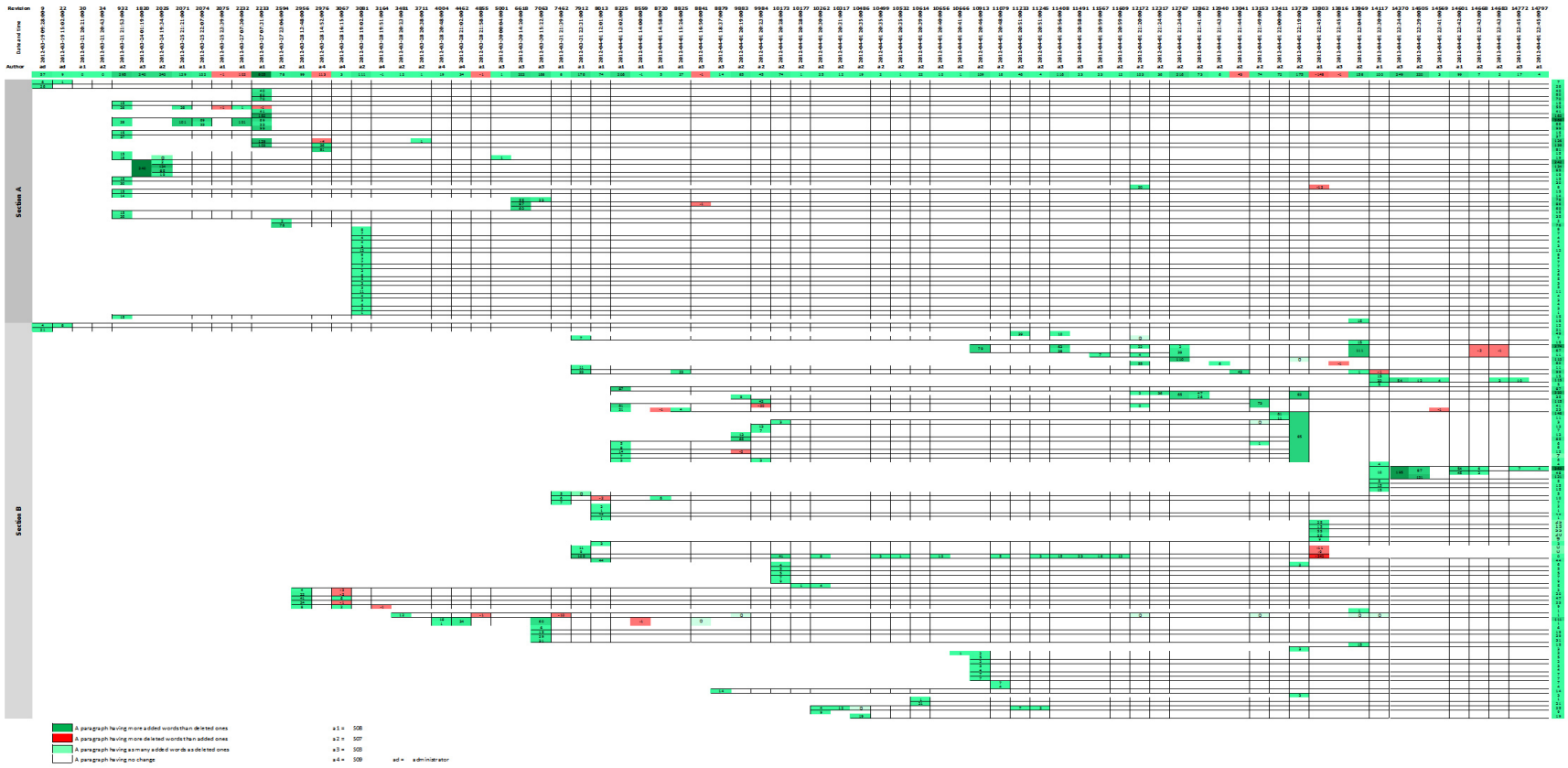


Figure D-1. Revision Map of Group c3g1.

Figure D-2. Revision Map of Group c3g2.

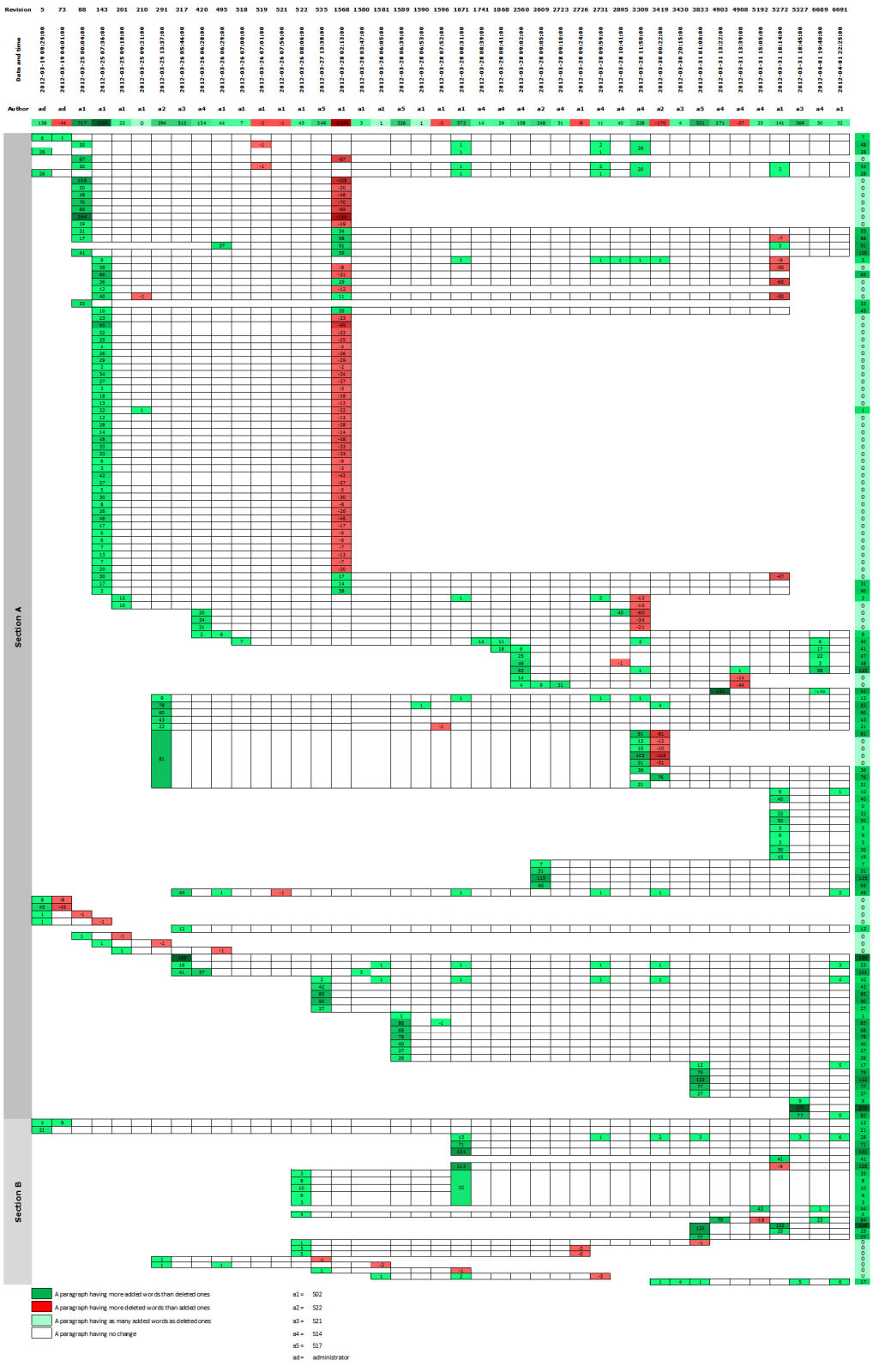


Figure D-3. Revision Map of Group c3g3.

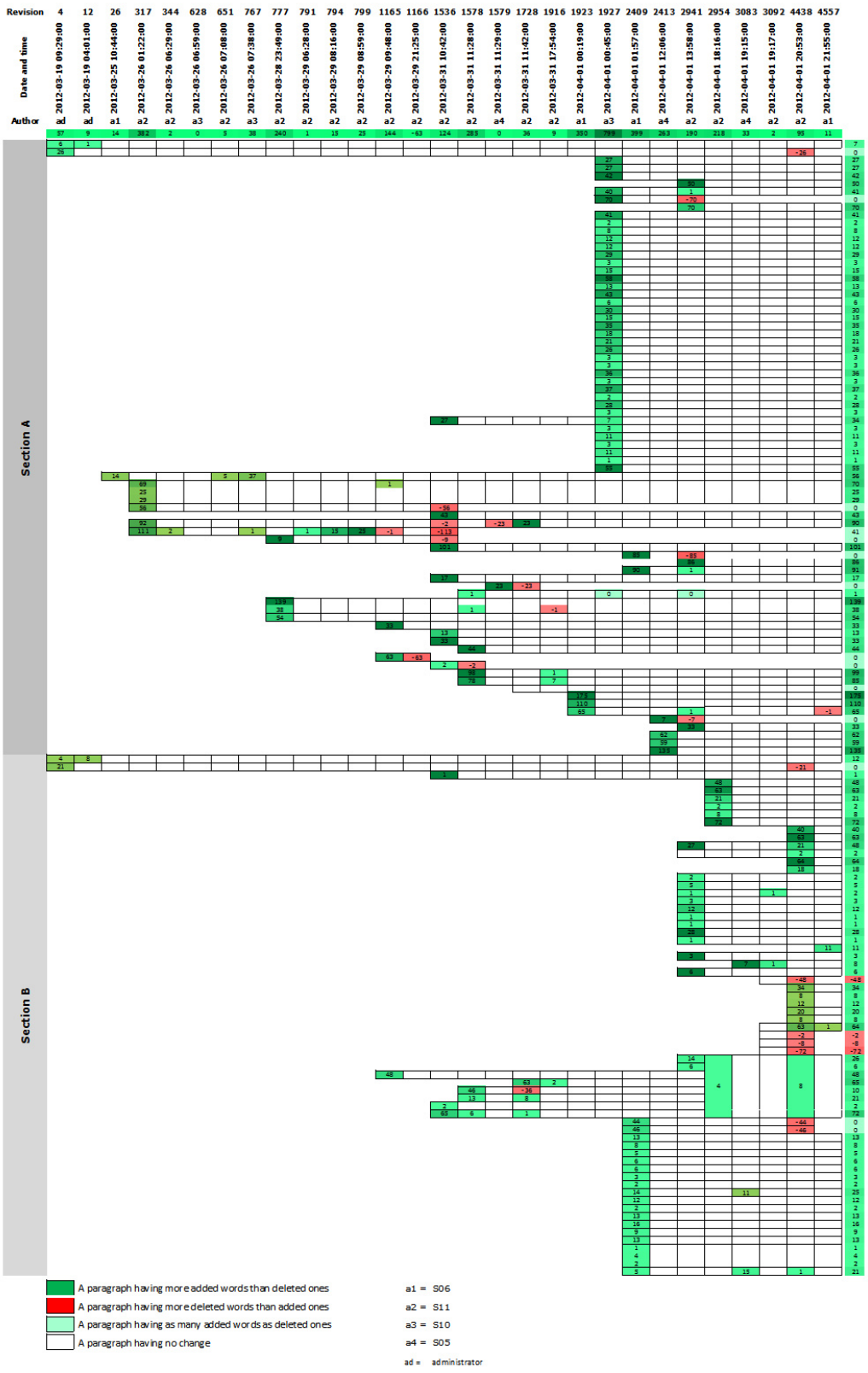
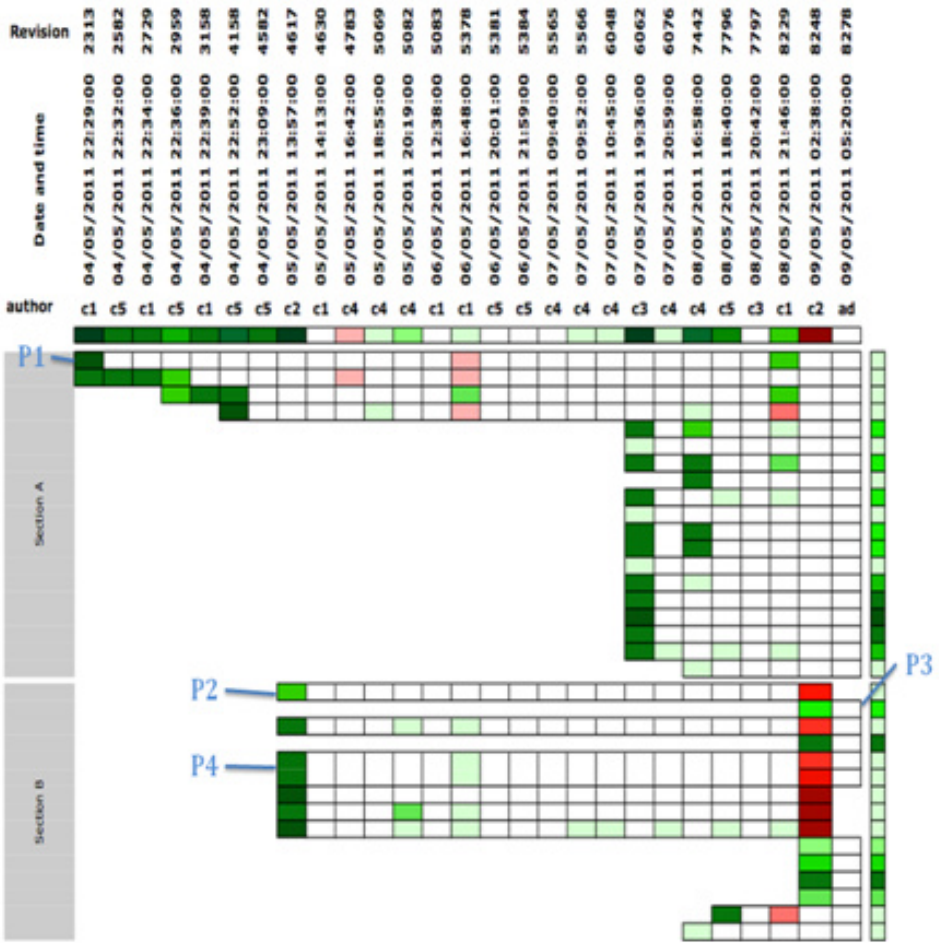


Figure D-4. Revision Map of Group c3g5.

APPENDIX E A SURVEY FOR REVISION MAPS

This appendix shows an example of the revision maps that were provided to students for use during their collaborative writing assignment, and the survey questions given to the students at the end of the assignment, as part of the case study incorporated in this thesis using real documents jointly-created over real time to evaluate the usefulness of revision maps with regard to writing tasks.

E.1 An Example of Revision Maps:



This visualisation represents the changes in a document written by a group of students over a period of six days (from 04 to 09/05/2011).

In this example, the text changes of four paragraphs (P1, P2, P3 and P4, as indicated in the revision map) are described as follows: The first paragraph of Section A (P1) is added on 04/05/2011 22:29 with a large number of words; this paragraph is not edited until 06/05/2011 16:48, when more words are deleted than added to it. Toward

the end of the week (on 08/05/2011 21:46), P1 is modified again when more words are added.

The first paragraph of Section B (P2) is inserted on 05/05/2011 13:57. After that, it is never modified at all and is then suddenly deleted from the document on 09/05/2011 02:38. A new paragraph (P3) is inserted after the removal of P2.

A paragraph can also be split and merged. For example, P4 is inserted on 05/05/2011 13:57, then changed when a few are words added on 06/05/2011 16:48; after that, it is split into two paragraphs on 09/05/2011 02:38.

E.2 Survey questions for revision maps

Your name:

Email address:

Date of completing this survey:

REVISION MAP

How to read the visualisation:

This visualisation represents the changes in your group's document over the past 9 days (from 01 to 09/03/2012).

Each small rectangle depicts a paragraph of your document. Each column refers to a revision of your document. Each row shows the evolution of an individual paragraph over time, as it is created, altered, or deleted during the writing process.

These rectangles are colour-coded to depict the nature and the extent of the changes made to the paragraph: *green* means more words were added than deleted; *red* means more words were deleted than added; and white represents no change in the paragraph. The intensity of these colours approximately denotes the extent of those changes. If there are as many added words as deleted ones, the rectangle is colour is very light green.

Lastly, the horizontal bar under the author ID row shows the aggregated changes of individual revisions; and the last vertical column represents the aggregated changes of individual paragraphs across all revisions during the 9 days.

Based on this visualisation, please answer the following questions.

1. According to this visualisation, **which part** (i.e. Section A, Section B, or Section C) of your document:

- has been changed **a lot**?
- has been changed only **a little**?
- has **not changed** at all?

2. According to this visualisation, **when** (at what dates) were **a lot** of words:

- **added** to the document? and in which part (i.e. Section A, Section B, or Section C) of your document?
- **deleted** from the document? and in which part (i.e. Section A, Section B, or Section C) of your document?

3. According to this visualisation, **who** (which author(s): a1, a2, a3, or a4, or all authors contributing equally) made:

- the **most** changes to the document?
- the **least** changes to the document?

4. According to this visualisation, **how many authors** worked on:

a) Part A:
b) Part B:
c) Part C:

5. According to this visualisation, do the following **patterns of text changes** apply to your group's writing?

a) **Sequential**: single paragraphs were written at different writing sessions or days.

NO YES , for Part(s) …

b) **Parallel**: many paragraphs were written almost in parallel, i.e. at the same writing session or day.

NO YES , for Part(s) …

c) Other: (please describe)

6. What does this visualisation tell you about **how** your group wrote the document? Please explain and discuss problems that your group encountered during writing.

7. Is this visualisation consistent with what you think actually happened during your writing?

- Please describe:

8. From the visualisation, do you learn something that you did not know before?

- Please discuss:

9. From the visualisation, would you do something differently in retrospect?

10. Does this visualisation prompt you to **do something**, or **plan an action**, regarding your group's writing?

YES NO

Please comment:

11. Is there other information about your group's writing process that you would like to get that is not represented in this visualisation?

YES NO

Please comment:

12. Would you find it useful to be provided with this kind of visualisation for your next group writing assignment? [on a *scale from 1 (strongly disagree) to 7 (strongly agree)*].

Please comment:

BIBLIOGRAPHY

Alpaydin, E. (2010). *Introduction to Machine Learning*: The MIT Press.

Andrews, N. O., & Fox, E. A. (2007). *Recent Developments in Document Clustering*. Computer Science, Virginia Tech.

Bereiter, C., & Scardamalia, M. (1987). *The Psychology of Written Composition*: Mahwah, NJ: Lawrence Erlbaum Plublishers.

Bilmes, J. A. (2006). What HMMs Can Do. *IEICE Transactions on Information and Systems., E89-D*(3), 869-891.

Blei, D. M., & Lafferty, J. D. (2009). Topic models *Text Mining: Theory and Applications*. London: Taylor and Francis.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research, 3*, 993-1022. doi: 10.1162/jmlr.2003.3.4-5.993

Boiarsky, C. (1984). Model for Analyzing Revision. *Journal of Advanced Composition, 5*, 65-78.

Bostock, M. (2012). D3.js: Data-Driven Documents.

Bozkaya, M., Gabriel, J., & van der Werf, J. M. (2009). *Process Diagnostics: A Method based on Process Mining*. Paper presented at the International Conference on Information, Process, and Knowledge Management.

Broniatowski, D. A., & Christopher, L. M. (2012). Studying Group Behaviours: A Tutorial on Text and Network Analysis Methods. *IEEE Signal Processing Magazine*, 22-32.

Bronner, A., & Monz, C. (2012). *User Edits Classification Using Document Revision Histories.* Paper presented at the the 13th Conference of the European Chapter of the Association for Computational Linguistics.

Buffet, S., & Geng, L. (2010). Using Classification Methods to Label Tasks in Process Mining. *Software Process: Improvement and Practice, 22*(6-7), 497-517

Calvo, R. A., O'Rourke, S. T., Jones, J., Yacef, K., & Reimann, P. (2011). Collaborative writing support tools on the cloud. *IEEE Transactions on Learning Technologies, 4*(1), 88-97.

Caporossi, G., & Leblay, C. (2011). Online Writing Data Representation : A Graph Theory Approach. *Search*, 80-89.

Carlson, P. A., & Berry, F. C. (2008). Using Computer-Mediated Peer Review in an Engineering Design Course. *IEEE Transactions on Professional Communication, 51*(3), 264-279.

Cho, K., & Schunn, C. D. (2007). Scaffolded writing and rewriting in the discipline: A web-based reciprocal peer review system. *Computers & Education, 48*(3), 409-426.

Crossley, S. A., & Mcnamara, D. S. (2007). Cohesion , Coherence , and Expert Evaluations of Writing Proficiency. *Corpus*, 984-989.

Daxenberger, J., & Gurevych, I. (2012). A Corpus-Based Study of Edit Categories in Featured and Non-Featured Wikipedia Articles *the 24th International Conference on Computational Linguistics (COLING 2012).*

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of The American Society for Information Science, 41*(6), 391-407.

Dong, Z. (2002). *Towards Web Information Clustering.* Doctoral Dissertation, Southeast University.

Dumais, S. T. (1991). Improving the retrieval of information from external sources. *Behavior Research Methods, 23*(2), 229-236.

Ede, L. S., & Lunsford, A. A. (1992). *Singular Text/Plural Authors: Perspectives on Collaborative Writing*: Southern Illinois University Press.

Erickson, T., Smith, D. N., Kellogg, W. A., Laff, M., Richards, J. T., & Bradner, E. (1999). *Socially Translucent Systems: Social Proxies, Persistent Conversation, and the Design of "Babble".* Paper presented at the the SIGCHI conference on Human factors in computing systems.

EtherPad. (2013). http://etherpad.org.

Faigley, L., & Witte, S. (1981). Analyzing Revision. *College Composition and Communication, 32*(4), 400-414.

Fellbaum, C. (2005). WordNet and wordnets. In Keith et al. (Ed.), *Encyclopedia of Language and Linguistics* (pp. 665-670). Oxford: Elsevier.

Ferschke, O., Daxenberger, J., & Gurevych, I. (2013). A Survey of NLP Methods and Resources for Analyzing the Collaborative Writing Process in Wikipedia, *The People's Web Meets NLP: Collaboratively Constructed Language Resources,* p. (to appear).

Flower, L., & Hayes, J. (1981). A Cognitive Process Theory of Writing. *College Composition and Communication, 32*(4), 365-387.

Fong, P. K., & Biuk-Aghai, R. P. (2010). *What Did They Do? Deriving High-Level Edit Histories in Wikis.* Paper presented at the the 6th International Symposium on Wikis and Open Collaboration, Gdansk, Poland.

Galbraith, D. (1999). Writing as a knowledge-constituting process. In M. Torrance & D. Galbraith (Eds.), *Knowing What to Write* (pp. 139-160). Amsterdam: Amsterdam Unversity Press.

Galbraith, D. (2009). Writing about what we know: Generating ideas in writing. In R. Beard, D. Myhill, J. Riley & M. Nystrand (Eds.), *The SAGE Handbook of Writing Development* (pp. 48-64): SAGE Publications.

Goldberg, A., Russell, M., & Cook, A. (2003). The effect of computers on student writing: A meta-analysis of studies from 1992 to 2002. *Journal of Technology, Learning, and Assessment, 2*.

Google Docs. (2013). http://docs.google.com,

Google Docs White Paper. (2010a). *What's different about the new Google Docs.* Retrieved from http://googledocs.blogspot.com.au/2010/09/whats-different-about-new-google-docs_23.html

Google Docs White Paper. (2010b). *What's different about the new Google Docs (1).* Retrieved from http://googledocs.blogspot.com.au/2010/09/whats-different-about-new-google-docs.html

Google Docs White Paper. (2010c). *What's different about the new Google Docs (2).* Retrieved from http://googledocs.blogspot.com.au/2010/09/whats-different-about-new-google-docs_22.html

Google Documents List API. (2012). Version 3.0. https://developers.google.com/google-apps/documents-list/.

Graesser, A. C., McNamara, D. S., Louwerse, M. M., & Cai, Z. (2004). Coh-Metrix: Analysis of text on cohesion and language. *Behavior Research Methods Instruments and Computers, 36*, 193-202.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America, 101 Suppl* 5228-5235. doi: 10.1073/pnas.0307752101

Gunther, C. W. (2009). XES Standard Definition, http://www.xes-standard.org/

Haley, D. T., Thomas, P., Roeck, A. D., & Petre, M. (2005). *A research taxonomy for latent semantic analysis-based educational applications*. Technical Report. Department of Computing, Faculty of Mathematics and Computing, The Open University.

Hall, D., Jurafsky, D., & Manning, C. (2008). *Studying the History of Ideas Using Topic Models.* Paper presented at the the Conference on Empirical Methods in Natural Language Processing.

Han, J., Wang, C., & Jiang, D. (2011). *Probabilistic Quality Assessment Based on Article's Revision History.* Paper presented at the the 22nd International Conference Database and Expert Systems Applications, Toulouse, France.

Hasan Dalip, D., André Gon\ccalves, M., Cristo, M., & Calado, P. a. (2009). *Automatic quality assessment of content created collaboratively by web communities: a case study of wikipedia.* Paper presented at the the 9th ACM/IEEE-CS joint conference on Digital libraries, Austin, TX, USA.

Hayes, J. R., & Flower, L. S. (1980). Identifying the organization of writing process. In L. W. Gregg & E. R. Steinberg (Eds.), *Cognitive Process in Writing* (pp. 3-30). Hillsdale, NJ: Lawrence Erbaum Associates.

Heeter, P., & Jeong, A. (2012). *The sequential analysis of individual versus collaborative writing processes in Wikis.* Paper presented at the the 2012 American Educational Research Association conference, Vancouver, B.C.

Hindle, A., Godfrey, M. W., & Holt, R. C. (2009). *What's hot and what's not: Windowed devloper topic analysis.* Paper presented at the The 25th International Conference on Software Maintenance.

Hunt, J. W., & McIlroy, M. D. (1976). *An Algorithm for Differential File Comparison*. Computing Science Technical Report. Bell Laboratories 41.

Jeong, H., & Biswas, G. (2008). *Mining Student Behavior Models in Learning-by-Teaching Environments.* Paper presented at the Educational Data Mining 2008: 1st International Conference on Educational Data Mining, Proceedings.

Jeong, H., Biswas, G., Johnson, J., & Howard, L. (2010). *Analysis of Productive Learning Behaviors in a Structured Inquiry Cycle Using Hidden Markov Models.* Paper presented at the International Conference on Educational Data Mining, Pittsburgh, PA, USA.

Juang, B.-H., & Rabiner, L. R. (1990). The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 38*, 1639-1641. doi: 10.1109/29.60082

Kay, J., Maisonneuve, N., Yacef, K., & Reimann, P. (2006). *The big five and visualisations of team work activity.* Paper presented at the the 8th International ConferenceIntelligent Tutoring Systems.

Kay, J., Maisonneuve, N., Yacef, K., & Zaïane, O. (2006). *Mining patterns of events in students' teamwork data.* Paper presented at the Educational Data Mining Workshop, held in conjunction with Intelligent Tutoring Systems (ITS).

Kelly, N., Jacobson, M., Markauskaite, L., & Southavilay, V. (2012). *Agent-Based Computer Models for Learning About Climate Change and Process Analysis Techniques.* Paper presented at the 10th International Conference of the Learning Sciences, Sydney, Australia.

Kim, S., & Lebanon, G. (2010). *Local Space-Time Smoothing for Version Controlled Documents.* Paper presented at the the 23rd International Conference on Computational Linguistics, Beijing, China.

Kushmerick, N., & Lau, T. A. (2006). *Automaticed Email Activity Management: An Unsupervised Learning Approach.* Paper presented at the the 2005 International Conference on Intelligent User Interfaces, San Diego, California.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: the latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review, 104*, 211-240.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An Introduction to Latent Semantic Analysis. *Discourse Processes*(25), 259-284.

Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (Eds.). (2007). *LSA: A road to meaning*. Mahwah, NJ: Lawrence Erlbaum Association.

Leijten, M., & Van Waes, L. (2006). Inputlog: New Perspectives on the Logging of On-Line Writing Processes in a Windows Environment. In G. Rijlaarsdam, K. P. H. Sullivan & E. Lindgren (Eds.), *Studies in Writing* (pp. 73-93): Elsevier.

Li, C., & Biswas, G. (2002). A bayesian approach for learning hidden markov models from data. *Special issue on Markov Chain and Hidden Markov Models, Scientific Programming, 10*, 201-219.

Linstead, E., Lopes, C., & Baldi, P. (2008). *An application of latent Dirichlet allocation to analyzing software.* Paper presented at the Proceeding of the 7th International Conference on Machine Learning and Application.

Liu, M., & Calvo, R. A. (2011). *Question Taxonomy and Implications for Automatic Question Generation.* Paper presented at the Artificial Intelligence in Education, Auckland, New Zealand.

Liu, Y., Poshyvanyk, D., Ferenc, R., Gyimothy, T., & Chrisochoides, N. (2009). *Modeling class cohesion as mixtures of latent topics.* Paper presented at the IEEE International Conference on Software Maintenance.

Lowry, P. B., Curtis, A., & Lowry, M. R. (2003). Building a Taxonomy and Nomenclature of Collaborative Writing to Improve Interdisciplinary Research and Practice. *Journal of Business Communication, 41*, 66-99.

Lowry, P. B., & Nunamaker, J. F. (2003). Using Internet-based, distributed collaborative writing tools to improve coordination and group awareness in writing teams. *Professional Communication, IEEE Transactions on, 46*, 277-297.

Lucene. (2013). Apache Lucene, http://lucene.apache.org

Macedo, A. L., Reategui, E., Lorenzatti, A., & Behar, P. (2009). *Using Text-Mining to Support the Evaluation of Texts Produced Collaboratively.* Paper presented at the Education and Technology for a Better World.

Markauskaite, L., Jacobson, M., Southavilay, V., & Kelly, N. (2012, April). *Using Process Analysis Techniques to Understand Students' Learning Strategies with Computer Models.* Paper presented at the American Educational Research Association (AERA) Annual Meeting, Vancouver, Canada.

Markov, Z., & Larose, D. T. (2007). *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*: Wiley.

McCallum, A. K. (2002). MALLET: A Machine Learning for Language Toolkit, http://mallet.cs.umass.edu

McNamara, D., Louwerse, M., McCarthy, P., & Graesser, A. (2010). Coh-Metrix: Capturing Linguistic Features of Cohesion. *Discourse Processes, 47*, 292-330. doi: 10.1080/01638530902959943

McNamara, D. S., Crossley, S. a., & McCarthy, P. M. (2009). Linguistic Features of Writing Quality. *Written Communication, 27*, 57-86. doi: 10.1177/0741088309351547

McNamara, D. S., Kintsch, E., Butler-Songer, N., & Kintsch, W. (1996). Are good texts always better? Interactions of text coherence, background knowledge, and levels of underdanstanding in learning from text. *Cognition and Instruction, 14*, 1-43.

Medeiros, A. K. A. D., Weijters, A. J. M. M., & van der Aalst, W. M. P. (2007). Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery, 14*(2), 245-304.

Mei, Q., & Zhai, C. X. (2005). *Discovering evolutionary them patterns from text: an exploration of temporal text mining.* Paper presented at the The 11th International Conference on Knowledge Discovery in Data Mining.

Myers, E. (1986). O(ND) Difference Algorithm and Its Variations. *Algorithmica, 1*(2), 251–266.

Novak, J. D., & Gowin, D. B. (1984). *Learning How To Learn*. Cambridge: Cambridge University Press.

O'Reilly, T., & McNamara, D. S. (2007). Reversing the reverse cohesion effect: Good texts can be better for strategic, high-knowledge readers. *Discourse Processes, 43*, 121-152.

O'Rourke, S., Calvo, R. A., & McNamara, D. (2011). Visualizing Topic Flow in Students' Essays. *Journal of Educational Technology and Society, 14*(3), 4-15.

Olson, D. L., & Delen, D. (2008). *Advanced Data Mining Technique*: Springer.

Osinski, S., Stefanowski, J., & Weiss, D. (2004). *Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition.* Paper presented at the Advances in Soft Computing, Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM´04 Conference, Zakopane, Poland.

Osinski, S., & Weiss, D. (2004). *Conceptual Clustering Using Lingo Algorithm : Evaluation on Open Directory Project Data.* Paper presented at the Advances in Soft Computing, Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM´04 Conference, Zakopane, Poland.

Osinski, S., & Weiss, D. (2005). A Concept-Driven Algorithm for Clustering Search Results. *IEEE Intelligent Systems, 20*, 48-54.

Ozuru, Y., Briner, S., Best, R., & McNamara, D. (2010). Contributions of Self-Explanation to Comprehension of High- and Low-Cohesion Texts. *Discourse Processes, 47*, 641-667. doi: 10.1080/01638531003628809

Palincsar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction, 1*(2), 117-175.

Pechenizkiy, M., Trcka, N., Vasilyeva, E., Aalst, W. M. P. v. d., & Bra, P. D. (2009). *Process Mining Online Assessment Data.* Paper presented at the the Second International Conference on Educational Data Mining, Cordoba, Spain.

Pechenizkiy, M., Trcka, N., Vasilyeva, E., Aalst, W. M. P. v. d., & De Bra, P. (2009). *Process Mining Online Assessment Data*. Paper presented at the Educational Data Mining.

Perrin, D., & Wildi, M. (2010). Statistical modeling of writing processes. In C. Bazerman, R. Krut, K. Lunsford, S. McLeod, S. Null, P. Rogers & A. Stansell (Eds.), *Traditions of Writing Research* (pp. 378-393): Routledge.

ProM. (2010). Version 5.2, http://prom.win.tue.nl/tools/prom/,

ProM. (2013). Version 6.2, http://www.promtools.org/prom6/,

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*, 257-286. doi: 10.1109/5.18626

Romero, A. C., & Ventura, S. (Eds.). (2006). *Data mining in e-learning*. Southampton: WITpress.

Rosen-Zvi, M., Chemudugunta, C., Griffiths, T., Smyth, P., & Steyvers, M. (2010). Learning author-topic models from text corpora. *ACM Transactions on Information Systems, 28*, 1-38. doi: 10.1145/1658377.1658381

Rosen-zvi, M., Griffiths, T., Steyvers, M., & Smyth, P. (2003, July 7-11). *The Author-Topic Model for Authors and Documents.* Paper presented at the the 20th Conference on Uncertainty in Artificial Intelligence, Banff, Canada.

Rozinat, A., de Jong, I. S. M., Gunther, C. W., & van der Aalst, W. M. P. (2007). *Process Mining of Test Processes: A Case Study*. BETA Working Paper Series, WP 220. Eindhoven University of Technology. Eindhoven.

Salton, G., & McGill, M. (1983). *Introduction to Modern Information Retrieval*: McGraw-Hill.

Scardamalia, M., & Bereiter, C. (1996). Engaging Students in a Knowledge Society. *Educational Leadership, 54*(3).

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics, 6*(2), 461-464.

Shermis, M. D., & Burstein, J. (2003). *Automated Essay Scoring: A Cross-disciplinary Perspective* (Vol. 16): MIT Press.

Song, M., & van der Aalst, W. M. P. (2007). *Supporting Process Mining by Showing Events at a Glance.* Paper presented at the the Seventh Annual Workshop on Information Technologies and Systems.

Song, M., & van der Aalst, W. M. P. (2008). Towards comprehensive support for organizational mining. *Decision Support Systems, 46*(1), 300-317.

Southavilay, V., Yacef, K., & Calvo, R. A. (2009). *WriteProc: A Framework for Exploring Collaborative Writing Processes.* Paper presented at the Australasian Document Computing Symposium, Sydney, Australia.

Southavilay, V., Yacef, K., & Calvo, R. A. (2010). *Process Mining to Support Students' Collaborative Writing.* Paper presented at the the third International Conference on Educational Data Mining, Pittsburgh, PA, USA.

Southavilay, V., Yacef, K., & Calvo, R. A. (2010b). *Analysis of Collaborative Writing Processes Using Hidden Markov Models and Semantic Heuristics.* Paper presented at the Proceedings of the third International Workshop on Semantic Aspect of Data Mining, Sydney, Australia.

Steinbach, M., Karypis, G., & Kumar, V. (2000). *A Comparison of Document Clustering Techniques.* Paper presented at the Proceedings of the International KDD Workshop on Text Mining 2000.

Sutskever, I., Martens, J., & Hinton, G. (2011, June). *Generating Text with Recurrent Neural Networks* Paper presented at the the 28th International Conference on Machine Learning (ICML-11), Bellevue, Washington, USA.

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical Dirichlet Processes. *Journal of the American Statistical Association, 10*, 1566-1581.

Thomas, C., & Sheth, A. P. (2007). Semantic Convergence of Wikipedia Articles *Web Intelligence* (pp. 600-606).

Thomas, S. W. (2011, May 21-28). *Mining Software Repositories Using Topic Models.* Paper presented at the the 33rd International Conference on Software Engineering, Waikiki, Honolulu, HI, USA.

Thomas, S. W., Adams, B., Hassan, A. E., & Blostein, D. (2010a). *DiffLDA : Topic Evolution in Software Projects*. Technical Report 2010-574. School of Computting, Queen's University.

Thomas, S. W., Adams, B., Hassan, A. E., & Blostein, D. (2010b). *Validating the Use of Topic Models for Software Evolution.* Paper presented at the EEE International Working Conference on Source Code Analysis and Manipulation (SCAM), imisoara, Romania.

Thomas, S. W., Adams, B., Hassan, A. E., & Blostein, D. (2011, May 21-28). *Modeling the Evolution of Topics in Source Code Histories.* Paper presented at the the 8th IEEE working conf on mining software repositories, Honolulu, HI, USA.

Thompson, K., Kennedy-Clark, S., Markauskaite, L., & Southavilay, V. (2011). *Capturing and analysing the processes and patterns of learning in collaborative learning environments.* Paper presented at the The Ninth International Conference on Computer-Supported Collaborative Learning, Hong Kong, July.

Tillema, M., Bergh, H., Rijlaarsdam, G., & Sanders, T. (2011). Relating self reports of writing behaviour and online task execution using a temporal model. *Metacognition and Learning*. doi: 10.1007/s11409-011-9072-x

Toolbox, T. M. (2012). http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm,

Trčka, N., Pechenizkiy, M., & van der Aalst, W. (2010). Process Mining from Educational Data. In C. Romero, S. Ventura, M. Pechenizkiy & R. S. J. d. Baker (Eds.), *Handbook of Educational Data Mining*. Boca Raton: CRC Press.

Upton, K., & Kay, J. (2009). *Narcissus: interactive activity mirror for small groups.* Paper presented at the User Modeling, Adaptation and Personalisation, Trento, Italy.

van der Aalst, W. M. P. (2011). *Process Mining: Discovery, Conformance and Enhancement of Business Processes*: Springer.

van der Aalst, W. M. P., Weijters, A. J. M. M., & Maruster, L. (2004). Workflow Mining: Discovering Process Models From Event Logs. *IEEE Transaction on Knowledge and Data Engineering, 16*(9), 1128-1142.

Varelas, G., Voutsakis, E., & Raftopoulou, P. (2005). *Semantic Similarity Methods in Wordnet and Their Application to Information Retrieval on the Web*. Paper presented at the ACM International Workshop on Web Information and Data Management.

Villalon, J., & Calvo, R. A. (2009). *Single Document Semantic Spaces*. Paper presented at the The Australasian Data Mining conference.

Villalon, J., & Calvo, R. A. (2011). Concept maps as cognitive visualizations of writing assignments. *Journal of Educational Technology and Society, 14*(3), 16-27.

Villalón, J. J., Kearney, P., Calvo, R. A., & Reimann, P. (2008). *Glosser: Enhanced Feedback for Student Writing Tasks.* Paper presented at the International Conference on Advanced Learning Technologies, Sydney, Australia.

Viterbi, A. (2006). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor., 13*(2), 260-269.

Wallach, H. M. (2008). *Structured Topic Models for Language.* Doctoral Dissertation, University of Cambridge.

Weijters, A. J. M. M., & Ribeiro, J. T. S. (2010). *Flexible Heuristics Miner (FHM).* BETA Working Paper Series, WP 334. Eindhoven University of Technology. Eindhoven.

Weijters, A. J. M. M., & van der Aalst, W. M. P. (2003). Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering., 10*(2), 151-162.

Weijters, A. J. M. M., van der Aalst, W. M. P., & Medeiros, A. K. A. D. (2006). Process Mining with the HeuristicsMiner Algorithm. *Technology, 166*(WP 166), 1-34.

Weston, J. L., Crossley, S. A., McCarthy, P. M., & McNamara, D. S. (2011). *Number of Words versus Number of Ideas: Finding a Better Predictor of Writing Quality.* Paper presented at the Twenty-Fourth International FLAIRS Conference.

Wikipedia. (2013). http://www.wikipedia.org,

Wilensky, U. (2013). NetLogo. Evanston, IL: Center for Connected Learning and Computer-Based Modeling. Northwestern University, http://ccl.northwestern.edu/netlogo/

Xiaoli, F., Chaitanya, K., Valentina, G., & Margaret, B. (2010). Mining problem-solving strategies from HCI data. *ACM Transactions on Computer-Human Interaction, 17*(1), 1-22. doi: http://doi.acm.org/10.1145/1721831.1721834

Yih, W., & Meek, C. (2007). *Improving Similarity Measures for Short Segments of Text.* Paper presented at the The 22nd National Conference on Artificial Intelligence.

Zeng, H., Alhossaini, M. A., Ding, L., Fikes, R., & McGuinness, D. L. (2006). *Computing trust from revision history.* Paper presented at the the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services, Markham, Ontario, Canada.