ABSTRACT

JONES, PAUL. Self-Tracking Instrumentation Agents and Analytics to Enable Knowledge Creation and Collaborative Intelligence from Analytical Workflows. (Under the direction of Dr. Nagiza F. Samatova.)

The ‘self-tracking’ (or ‘quantified-self’) movement is enabling people to gain increased self-awareness of aspects of their daily lives, such as health and fitness, and is enabling analytics developers to create useful services based on the collected data. This is achieved through a combination of sensors, algorithms and visualizations. More recently, it has become apparent that the self-tracking movement also has the potential to help knowledge workers gain self-awareness of their processes for analysis of complex information - individually and/or collaboratively - and to assist them with knowledge creation, collaborative intelligence, and other tasks.

This research is motivated by a study of how students and intelligence analysts work with their computers to find and manage information, the difficulties that they encounter, and the opportunities that aren't being fully explored as a result. We observe that often these groups work in teams to analyze information, and are often forced to regularly switch between multiple tasks. In such environments, the principle difficulties we discovered are in keeping track of information artifacts and sources during knowledge creation tasks, and especially doing so collaboratively. These can prevent knowledge workers from clearly reasoning about analytic conclusions, and from backing up those conclusions with detailed provenance.

We propose a novel method for mathematically modeling analytical workflows that links heterogeneous information sources with the tasks that they correspond to, and that combines information from collaborating users. Inspired by recent ideas in the deep learning and graph analytic communities, we propose a new abstraction of the idea of embedding entities in low dimensional spaces, which allows us to co-represent users, documents and tasks in a common vector space, and hence to automatically infer links between these different types of entities. While other applications of this method are possible, we focus on the challenge of inferring associations between documents (including files and URLs) and the individual and shared tasks to which they correspond - we refer to this as ‘task-centric document curation’. Our method allows us to easily leverage a combination of content-based features and task-related features, as well as other characteristics of individual and team-based workflows, such as temporal patterns. We also show how our algorithm can operate in close to real-time (in order to facilitate timely user feedback) and can work well with few training examples. To demonstrate our method in practice, we introduce a ‘recent-work dashboard’ exemplar application that displays the inferred associations, as well as gathering feedback from users.

We evaluate our approach using multiple studies with groups of real-world knowledge workers. Results from empirical evaluation of several algorithm variants against three analytic workflow datasets (collected from students and intelligence analysts) are presented, along with an exploration
and evaluation of ideas to further improve the classification accuracy that can be achieved from the vector space embeddings. One such method that proved successful is to incorporate coarse-grained temporal characteristics of tasks. We show how to leverage relationships between different entity types to increase classification accuracy by up to 20% over simpler baselines, and with as little as 10% labelled data from users.

To enable the development of these analytics, we first needed to capture data from longitudinal studies of analytic workflow, and to facilitate iterative prototype development. To achieve these aims, we address two further research challenges: the first of these is the challenge of creating self-tracking ‘instrumentation’ (or measurement) agents capable of capturing necessary features for task-centric document curation algorithms on a continuous basis. By learning from previous self-tracking prototypes that did not prove suitable for continuous use, we propose novel agents that are entirely passive (and hence non-disruptive to users), and we employ careful feature selection in order to create state-of-the-art desktop instrumentation agents. The second challenge is the creation of active ‘journaling’ interfaces to facilitate user provision of task labels for individual and collaborative tasks. This is tackled using a hybrid ‘task-tree’ interface consisting of a mixture of elements based on a fixed taxonomy and an adaptable ‘folksonomy’, combined with carefully controlled user prompting.

Finally, we present an alternative method for better enabling knowledge creation tasks by helping knowledge workers keep track of finer-grained information artifacts - this time in the context of ‘claims’ they are making in written reports, and the associated ‘evidence’ they might be using to justify their claims. We use a large corpus of screenshots collected from one of our instrumentation agents during a controlled analysis study to demonstrate a novel algorithm to automatically associate these two types of information artifacts - we refer to this as ‘automatic provenance generation’.

Whilst this dissertation focusses on the challenges involved in gathering instrumentation and journaling data, and on the creation of instrumentation-enabled analytics (such as task-centric document curation), we also discuss the wider implications of such self-tracking, workplace-monitoring and smart digital assistant technologies. In particular, these technologies necessitate a careful balance of human factor considerations (such as privacy intrusion) with the future potential of such technologies to save knowledge worker’s time and to improve the quality of their work.
Self-Tracking Instrumentation Agents and Analytics to Enable Knowledge Creation and Collaborative Intelligence from Analytical Workflows

by
Paul Jones

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2017

APPROVED BY:

_________________________  _______________________
Dr. Kathleen M. Vogel         Dr. Rada Y. Chirkova

_________________________  _______________________
Dr. Dennis R. Bahler          Dr. Nagiza F. Samatova
Chair of Advisory Committee
DEDICATION

To my mother, who never ceases to inspire me with her unfailing ability to just get on with it.
BIOGRAPHY

Paul Jones graduated from Cambridge University, England, in June 1999 with a Master's in Engineering, specializing in Electrical and Information Sciences. He gained a First Class degree with Distinction, and was awarded the Mike Lynch prize in Engineering, as well as a lifelong scholarship of Christ's College, Cambridge. Since then he has been an employee of Her Majesty's Government, and has taken on a variety of technical and leadership roles. These included a seven-year collaboration with IBM Research, in which he worked on a groundbreaking streaming analysis platform [Bal10], and helped to establish general design principles and architectural underpinnings for a new breed of complex stream processing applications [Tur10]. He became a Chartered Engineer in 2003 and has been an active member of the IET, IEEE and ACM ever since, working to promote careers in science and engineering to young people through community events. Paul joined the Laboratory for Analytic Sciences (LAS) in Raleigh, North Carolina, and commenced the PhD program in Computer Science at NC State University in January 2014. During his studies, he developed a passion for data-driven methods for understanding human sensemaking processes, and for advancing human-machine collaborative intelligence. He passed his Written Qualifier Examination in December 2015, and his Oral Preliminary Examination in January 2017. In his spare time, Paul is a keen aviator and holds both European and American pilot’s licenses. He is also co-founder of the 'Wolf Wings' aviation club at NC State University.
ACKNOWLEDGEMENTS

There are many people I would like to thank for helping with this work, and for making my time at the LAS and NCSU so productive and enjoyable. First and foremost, I would like to thank my tireless advisor, Dr Nagiza Samatova, for her expert guidance, ideas and highly motivating feedback over the years. It has been a great privilege to have been a part of her research group. Next, I would like to extend my deepest thanks to my doctoral committee members - Dr Kathleen Vogel, Dr Rada Chirkova, and Dr Dennis Bahler - for their valuable time and feedback. This work is the result of several highly fruitful collaborations - firstly, between the LAS and the College of Humanities and Social Sciences (CHASS) at NCSU, which was made possible by the expert direction and boundless energy of Dr Joann Keyton and her research assistants, Grant Harned, Heewon Shin and Hannah Jaffee. The data gathered from Dr Keyton’s first ‘collaborative interactive study’ in 2014 was used as the basis for the work in Chapter 4, and is continuing to push the bounds of both computer science and social science research. Secondly, I would like to acknowledge the collaboration between LAS and the Renaissance Computing Institute, RENCI, especially Steve Cox, Michael Matthews and Sidharth Thakur, who were responsible for much of the design and implementation of the Instrumentation platform and Journaling prototypes described in Chapters 3 and 5. Thirdly, I extend my sincere thanks and appreciation to Dr Kathleen Vogel, Chris Kampe, Gwendolynne Reid and Eli Typhina, without whom we could not have gathered the data for Study B, nor could we have gathered such detailed and insightful feedback from participants and LAS analysts. Next, I would like to thank all the staff at the LAS who kindly ‘instrumented’ themselves in order to provide the data for Study C (with a special shout-out to Sean Streck and John Slankas for their invaluable support), as well as all the students who helped with the implementation of various aspects of the prototypes and evaluations described here, including Stewart Cartmell, Shivani Sharma, Preetham Srinath, Sajal Garg, Manish Singh, Sreekanth Ramakrishnan, Rajat Shah and Arvind Lakshmanan. A special mention is also due to Jonathan Downing, whose infectious enthusiasm for the macOS platform helped enormously when creating the macOSInstru-menter. It has also been a great pleasure working with the highly talented graduate students in the data analytics research group at NCSU, especially Changsung Moon, Dakota Medd, Claude Hsu and Steve Harenberg, who provided helpful guidance for much of this work and for the associated publications, as well as all the students in the group who instrumented their laptops to provide data for Study A. Finally, many thanks to Adam Amos-Binks for helpful chats throughout my time as a grad student, and to James Grzymkowski for his comprehensive proof-read of this dissertation.

This material is based upon work supported in whole or in part with funding from the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.
# TABLE OF CONTENTS

LIST OF TABLES ........................................................................................................ viii

LIST OF FIGURES ..................................................................................................... ix

## Chapter 1  INTRODUCTION .................................................................................. 1
1.1 Motivation ........................................................................................................... 1
1.2 Terminology ........................................................................................................ 4
1.3 Research Questions and Approaches ................................................................. 5
  1.3.1 Task-Centric Document Curation (Chapter 2) ................................................ 5
  1.3.2 Passive Instrumentation (Chapter 3) ................................................................ 8
  1.3.3 Analysis of Screenshots (Chapter 4) ............................................................. 8
  1.3.4 Active Journaling Interfaces (Chapter 5) ....................................................... 9
1.4 Publications ........................................................................................................ 10
  1.4.1 Published .................................................................................................... 10
  1.4.2 Accepted .................................................................................................... 11
  1.4.3 Under Submission ...................................................................................... 11
1.5 Human Subject Participation ............................................................................. 12

## Chapter 2  TASK-CENTRIC DOCUMENT CURATION ............................................. 13
2.1 Problem Statement ............................................................................................ 13
  2.1.1 Formulation .............................................................................................. 14
2.2 Related Work .................................................................................................... 16
  2.2.1 Task Switch Detection and Task Prediction ................................................ 16
  2.2.2 Document Content Analysis ..................................................................... 20
  2.2.3 Task Sequence Prediction ....................................................................... 22
  2.2.4 Heterogeneous Network Fusion ............................................................... 23
  2.2.5 Classifier Combination ............................................................................ 25
2.3 Major Research Contributions ......................................................................... 26
2.4 User Behavior Characterization ........................................................................ 27
  2.4.1 Datasets .................................................................................................... 27
  2.4.2 Observations ............................................................................................. 28
2.5 Algorithmic Approach ....................................................................................... 30
  2.5.1 Simplest Baseline ..................................................................................... 31
  2.5.2 Classification by Document Similarity ....................................................... 32
  2.5.3 Classification by Task Sequence Prediction ............................................. 34
  2.5.4 Classification from Heterogeneous Network Fusion .................................. 35
  2.5.5 Enhancing the Node Embeddings ............................................................. 39
  2.5.6 Combination of Classifiers .................................................................... 42
  2.5.7 Approaches Chosen for Evaluation .......................................................... 42
2.6 Empirical Evaluation ......................................................................................... 43
  2.6.1 Workflow datasets ................................................................................... 43
  2.6.2 Parameter Optimization and Calibration .................................................. 43
<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>ACTIVE JOURNALING INTERFACES</th>
<th>95</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Motivation</td>
<td>95</td>
</tr>
<tr>
<td>5.2</td>
<td>Related Work</td>
<td>96</td>
</tr>
<tr>
<td>5.3</td>
<td>Major Contributions</td>
<td>98</td>
</tr>
<tr>
<td>5.4</td>
<td>Creating a Taxonomy of Tasks</td>
<td>98</td>
</tr>
<tr>
<td>5.5</td>
<td>Labeling Information Artifacts</td>
<td>99</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Requesting labels during initial document use</td>
<td>99</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Requesting labels through the dashboard</td>
<td>101</td>
</tr>
<tr>
<td>5.6</td>
<td>Benefits to Users and Managers</td>
<td>102</td>
</tr>
<tr>
<td>5.7</td>
<td>Empirical Evaluation</td>
<td>103</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Approach - Iterative Evaluation Studies</td>
<td>103</td>
</tr>
<tr>
<td>5.7.2</td>
<td>Computer Science Class</td>
<td>105</td>
</tr>
<tr>
<td>5.7.3</td>
<td>Political Science Class</td>
<td>107</td>
</tr>
<tr>
<td>5.7.4</td>
<td>Intelligence Analysts</td>
<td>108</td>
</tr>
<tr>
<td>5.8</td>
<td>Discussion and Future Work</td>
<td>109</td>
</tr>
<tr>
<td>5.9</td>
<td>Human Factor Considerations</td>
<td>110</td>
</tr>
</tbody>
</table>

| Chapter 6   | CONCLUSIONS AND FUTURE DIRECTIONS                                                              | 114|

BIBLIOGRAPHY                                                                                     | 117|
### LIST OF TABLES

| Table 1.1 | Common terms used in this dissertation. | 4 |
| Table 2.1 | Comparison of main features of related task switch detection and task prediction research. | 19 |
| Table 2.2 | Basic characterization of Journaling datasets. | 28 |
| Table 2.3 | Results from classification using document vectors alone. | 34 |
| Table 2.4 | Accuracy of next task prediction using sequence prediction alone. | 35 |
| Table 2.5 | Characteristics of our 3 workflow datasets for algorithm evaluation. | 44 |
| Table 2.6 | Percentage changes in accuracy from baseline. | 46 |
| Table 2.7 | Analysis of standard deviations in cumulative accuracy over 10 runs on each dataset with varying probability of user feedback. | 54 |
| Table 3.1 | Core Section of Event Protocol. | 68 |
| Table 3.2 | Events captured by `macOSInstrumenter` during a single run of the experiment, based on the activity of three people over a 90-minute period. | 72 |
| Table 3.3 | Benchmarking Characterization of Instrumentation Platform. | 74 |
| Table 3.4 | Aspects of Sensemaking we could attempt to measure using our platform. | 77 |
| Table 4.1 | Summary of key metrics from the screenshot corpus [KJ17]. | 88 |
| Table 4.2 | Confusion matrix for active window detection. | 88 |
| Table 4.3 | Summary of results from Stateless and Stateful algorithms. | 89 |
| Table 4.4 | Likert scoring for quality of claim-evidence associations. | 90 |
| Table 4.5 | Fleiss’ Kappa Statistics for Stateful Jaccard ratings. | 91 |
| Table 4.6 | Examples of high quality claim-evidence associations extracted by our algorithms, which were given a score of 5 (Excellent) on our Likert scale by human raters. | 92 |
| Table 5.1 | Comparison of Journaling tools in industry and academia. | 97 |
| Table 5.2 | Levels Defined in Task-Tree Hierarchy. | 99 |
| Table 5.3 | Participation and Engagement Metrics obtained from the 3 studies. | 106 |
LIST OF FIGURES

Figure 1.1 Survey Results from Knowledge Workers .............................................. 3
Figure 1.2 Recent-work dashboard exemplar application ........................................ 7

Figure 2.1 Sequence of User Document Accesses .................................................... 15
Figure 2.2 Observed task switching behavior for users in Study A .......................... 29
Figure 2.3 Task switching behavior observed from studies .................................... 30
Figure 2.4 2D embedding of similarity between TF-IDF document vectors .............. 33
Figure 2.5 Cumulative accuracy of streaming sequence prediction algorithms with varying probability of feedback ................................................................. 36
Figure 2.6 Multi-Layer Graph relating Users, Tasks and Documents ....................... 38
Figure 2.7 Automatic construction of edges in Task layer ...................................... 40
Figure 2.8 Micro and Macro-F1 scores for batch-mode evaluation ......................... 45
Figure 2.9 t-SNE plot showing embeddings of User||Task nodes ............................ 49
Figure 2.10 t-SNE plot showing embeddings of concatenated node combinations ...... 50
Figure 2.11 Top-n accuracies on the three datasets ................................................. 51
Figure 2.12 F1 scores for node2vec evaluation of multi-layer graph algorithm ........ 52
Figure 2.13 Cumulative accuracy in streaming mode with varying probability of feedback ................................................................. 57

Figure 3.1 Our model of an Instrumentation Hierarchy ............................................ 59
Figure 3.2 Instrumentation Platform Overview and Architecture ............................ 66
Figure 3.3 Example of a Personnel Dashboard ....................................................... 69
Figure 3.4 Data Collection/Aggregation Service Architecture .................................. 69

Figure 4.1 Example of a screenshot captured during the experiment .................... 84
Figure 4.2 Likert Scores for Evaluating Quality of Claim-Evidence Associations ........ 91
Figure 4.3 Human vs Jaccard Similarity Scores ..................................................... 92
Figure 4.4 Examples of Some Problem Images ...................................................... 93

Figure 5.1 Journaling User Interfaces ................................................................. 100
Figure 5.2 Example Feedback button added to Dashboard .................................... 102
Figure 5.3 Icicle Plot and Task-Tree Visualization constructed from data from Study A 104
Figure 5.4 Survey Results on Journaling Interfaces .............................................. 107
Figure 5.5 Collaboration patterns over time in Study B ....................................... 109
1.1 Motivation

Knowledge workers - including students, intelligence analysts and others - are being exposed to more information than ever before, as well as increasingly having to work in multi-tasking [Dra05] and collaborative [DC15a] environments. The overarching goal of the work described in this dissertation is to support these workers by generating continuous feeds of data on what they are doing (using Instrumentation Agents on their computer desktops), and by creating novel analytics to run on this data that support them when carrying out information-rich tasks, such as knowledge creation [VK00] and collaborative sensemaking [Moo12]. We observe that the recent ‘self-tracking’ movement [GD16] has so far had the greatest impact in health, fitness and life-style domains, as well as in some specialist workplaces, such as trucking [Lev15] (although in this context the tracking is seldom voluntary). However, self-tracking technologies are yet to become widespread in the field of knowledge work, with the notable exception of ‘productivity’ apps that are designed to simply keep track of the time spent on particular activities [Res14][Des14]. We think the potential space of applications for self-tracking of knowledge workers is much wider than this.

Before commencing this research, we wanted to understand how knowledge workers typically worked with their computers to find information, and the frustrations that they encountered. We conducted a survey of grad students and intelligence analysts working in partnership with our university, and asked them about their approaches to storing, organizing and searching for documents
on their work computers, and what kinds of interfaces and algorithms they might find helpful.

A summary of responses to some of the questions we asked is provided in Figure 1.1. From this, we identified that these populations tend to spend time creating organized folder hierarchies on their computers. However, on downloading new documents, many people initially store them in a default location (such as their Desktop or Downloads folder) and, although most intended to sort them later, they often did not in practice. Also, they generally tended to access previously seen documents (that were related to past work) more than new ones. Next, we asked a freeform question about different ways to organize documents - suggestions included sorting by date, topic, keyword, project and task. In a subsequent question, 86% agreed that a system that could help organize by task (i.e. that could perform task-centric document curation) would be useful. We followed up the survey with some discussions with participants and discovered that the time spent organizing and keeping track of information artifacts was detracting from the time that knowledge workers wanted to be spending clearly reasoning about analytic conclusions, and on backing up those conclusions with a detailed record of where the information was derived from (i.e. recording the provenance of the information).

For these reasons, we decided to focus on algorithms and interfaces for organizing previously accessed documents (rather than for helping to search for new ones) and to do this in a task-centric manner. This is the focus of chapter 2. After a detailed literature review of previous algorithmic approaches, we realized that an opportunity existed to model analytic workflows in a new way, building on recent developments in the deep learning and graph analytic communities. Our research resulted in a multi-layer graph model for capturing analytic workflows that allows us to bring together a lot of different ‘clues’ as to the current task that a user is working on. We then map the different types of nodes in the graph (representing users, tasks and documents) to a common vector space in order to create an effective, robust and scalable algorithm to automatically associate documents (and potentially other entities) with individual or collaborative tasks. The challenge of creating a suitable interface for displaying the curated documents is not the primary focus of this dissertation but we considered an exemplar application, known as the ‘recent-work dashboard’, which is initially described in section 1.3.1.

In order to train robust task-centric document curation algorithms, it was first necessary to capture our own labelled datasets of user’s workflows. We wanted to do this over the course of several months work in order to capture sufficient volume and variety of tasks. Unfortunately, but understandably due to privacy and other concerns, these datasets are difficult to obtain, or do not exist, in the open research community. To achieve our aim, we began by researching how to enable users to continuously self-track their own workflows via passive (i.e. non-disruptive) desktop ‘instrumentation’, both in individual and team settings - this is the focus of chapter 3. We wanted to learn lessons from previous studies [Dra05][Cow05] that were only able to capture data from real users for relatively short periods of time (and often only by providing incentives). Next, before any work could
begin on supervised machine learning algorithms for organizing documents by task, it was necessary to ask participants to label their documents with the (possibly shared) tasks and goals to which they corresponded. The research challenge of creating active ‘journaling’ interfaces suitable for continuous use is the subject of chapter 5, which includes empirical evaluation studies with three groups of students and intelligence analysts. As well as providing labelled data for our algorithms, these studies also show how workflow logs can provide insights into collaborative sensemaking processes.

Finally, we demonstrate an alternative method for instrumentation of user desktops, based on capturing screenshots at regular intervals, and on events that indicate user engagement, such as mouse clicks. We show how this type of data can provide an opportunity to capture information artifacts at a lower granularity than entire documents. In chapter 4, we show how it is possible to use screenshots to extract sentences corresponding to ‘claims’ being made in written reports, and to automatically associate these with ‘evidence’ derived from web browsing, thus providing a means for automatic gathering of provenance information for some types of knowledge creation tasks.

We hope that this early work in self-tracking for knowledge workers will lead to more effective recommender and knowledge-creation systems in the future, and to other analytics that will help computers provide pro-active and timely assistance to knowledge workers. To further stimulate this, we provide a brief analysis in section 5.9 of some of the softer, ‘human factors’, involved in self-tracking technologies. This analysis suggests ideas to improve our prototypes, as well as potentially promising directions for future research work in this nascent field.

**Figure 1.1** Results from some 5-point Likert scale questions in a survey of 36 knowledge workers consisting of students and analysts. The width of each colored bar indicates the percentage of responses on a frequency scale from ‘Never/Occasionally’ (Red), through ‘Sometimes’ (Gray) to ‘Often/Always’ (Blue). Image reproduced from [Jon17a] with permission. ©ACM 2017.


## 1.2 Terminology

For the sake of consistency and clarity, a list of definitions of terms used throughout this dissertation is provided in Table 1.1. Some of these terms are well established in the research community; others have specific meanings in the context of the work described here.

### Table 1.1 Common terms used in this dissertation.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition in our context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytical Workflow</td>
<td>The process by which people (possibly in collaboration with others), apply their knowledge, tools, and organizational resources to perform information analysis tasks. Sometimes also referred to as <em>Tradecraft</em>.</td>
</tr>
<tr>
<td>Collaborative Intelligence</td>
<td>The use of teams to solve hard knowledge-related problems [Hac11].</td>
</tr>
<tr>
<td>Enterprise Intelligence</td>
<td>Information that helps provide overall situational awareness and a decision support structure for organizations.</td>
</tr>
<tr>
<td>Information Analysis</td>
<td>The science of evaluating information content, and refining data to develop an accurate understanding of its important features.</td>
</tr>
<tr>
<td>Instrumentation</td>
<td>A systematic approach for measuring human activities during the process of information analysis.</td>
</tr>
<tr>
<td>Instrumentation Agents</td>
<td>Devices or tools (software in our case) for measuring human activities while making sense of information on a computer.</td>
</tr>
<tr>
<td>Journaling</td>
<td>The activity of keeping a record of work-related actions, thoughts and ideas; specifically in the context of tasks and goals.</td>
</tr>
<tr>
<td>Knowledge Creation</td>
<td>The formation of new ideas in a human mind through interaction of explicit and tacit knowledge [Cor99].</td>
</tr>
<tr>
<td>Knowledge Worker</td>
<td>Someone who works with information or data. In our research, we include intelligence analysts, research professionals and college students.</td>
</tr>
<tr>
<td>Microblogging</td>
<td>The activity or practice of making short, frequent posts to a web log. Well known examples are <em>Twitter</em> and <em>Yammer</em>.</td>
</tr>
<tr>
<td>Process Mining</td>
<td>Techniques and tools for discovering process, control, data, organizational, and social structures from event logs [Don05]. This is also known in the industry as Automated Business Process Discovery (ABPD).</td>
</tr>
<tr>
<td>Recommender System</td>
<td>A system that generates suggestions for its users relating to items, actions or products that might be relevant to them [MS10].</td>
</tr>
<tr>
<td>Sensemaking</td>
<td>The process by which people give meaning to experiences and observations, specifically in the context of understanding (often hidden) patterns in data [Moo12].</td>
</tr>
<tr>
<td>Smart Digital Assistant</td>
<td>A new generation of technologies that attempt to provide personalized, real-time and relevant advice to computer or mobile users.</td>
</tr>
</tbody>
</table>
1.3 Research Questions and Approaches

Here we provide a summary of the key research questions addressed in each chapter of this dissertation, along with a high-level summary of the general approaches taken. The full context for each question should become clear from reading the ‘Motivation’ and ‘Related Work’ sub-sections of each chapter to which they relate.

1.3.1 Task-Centric Document Curation (Chapter 2)

This chapter addresses the challenge of building a machine learning model and classifier to automatically associate documents with the tasks to which they correspond. More formally:

*Can an efficient, accurate and scalable streaming algorithm be created to predict task associations for each information artifact that is accessed by knowledge workers in a collaborative and multi-tasking environment?*

Tackling this research question necessitated a unique algorithmic approach. Specifically, we required an algorithm capable of taking into account many different types of ‘clues’ to a document-task association, including from content analysis, collaboration patterns, task similarity and temporal patterns. Furthermore, the algorithm must be capable of providing probabilistic associations, it must handle (and learn from) multiple users, and it must be adaptable, scalable and suitable for streaming deployment. Each of these aspects is discussed in section 2.5. Our eventual approach built on much past research and led to the creation of a novel network-fusion algorithm that generates representations of users, tasks and documents in a common vector space.

In order to provide a reference application for our algorithm, we developed a novel dashboard that automatically organizes a user’s recently-accessed documents according to the tasks to which they relate. The dashboard is both enabled by document curation algorithms, and also provides a means to evaluate the algorithms with real-world users.

A screenshot of the ‘recent-work dashboard’ prototype is shown in Figure 1.2. Documents are listed in reverse-time order in blocks labelled by the task they are related to (note that the same document can appear in any number of task blocks). The dashboard relies on underlying ‘instrumentation’ to detect that documents have been accessed, and also on ‘journaling’ interfaces for gathering ground-truth task labels at the time of access, when these are needed.

Users can re-open all artifacts associated with a task simply by clicking the ‘Open All’ button to the right of the task name. This was reported to be a considerable time-saver when multi-tasking, although we have not yet quantified the time saved empirically. As noted in recent research [Liu14], providing visual feedback on task suspension can help people resume tasks more efficiently. We in-
corporated this observation into the dashboard by highlighting the currently suspended task, and displaying the time since the last activity. This acts as a subtle cue for the user to resume their work.

As well as assisting in information recall, the dashboard also allows documents and tasks to be re-labelled by users, thus improving the quality of future document/task associations. The dashboard also aims to promote collaboration by displaying a list of other users working on the same tasks (the ‘initial icons’ on the right hand side). In due course, we will also use the dashboard interface to provide recommendations and results from analytics running over the documents (e.g. on-demand topic classification and summaries).

In summary, the dashboard acts as an exemplar application for the task-centric document curation algorithm presented in this chapter, and motivates the need for the instrumentation and journaling components that underpin it. While we demonstrate these additional components in the context of the dashboard, they could equally be used to drive many other types of interface, including those that knowledge workers already use, such as operating system file/folder navigation interfaces, project management or other productivity tools, and smart digital assistants.
Figure 1.2 Example of part of a user’s ‘recent-work dashboard’ in a Chrome web browser. Information artifacts are organized according to the associated tasks, with most recently accessed first. The ‘Open’ buttons allow easy opening of multiple documents, either in a file explorer, or in the native application for each document type. The ‘Tag’ button allows users to re-label documents with a new task (or to delete labels) as needed. The ‘Summary’ button runs an algorithm to summarize the contents of documents. The currently active task is always shown at the top of the dashboard and highlighted. Collaborating users are denoted by colored icons on the right hand side - their public documents can be viewed by clicking these icons, which adds an additional colored block to the table. In this example, the collaborating users are ‘M’ and ‘S’. Users can mark documents (or entire task blocks) as private to prevent documents being shared. In this example, the ‘Training_DeepLearning’ task has been marked private by default, and only one document from this block has been shared; by contrast, ‘MyTeam_Admin’ has been marked public by default and all documents in this block are shared with collaborating users. Image reproduced from [Jon16b] with permission. ©IEEE 2016.
1.3.2 Passive Instrumentation (Chapter 3)

This chapter addresses the challenges of gathering continuous data streams of user activities related to their analytic workflow. In order to enable algorithms that can continually learn about user tasks, it is necessary to create instrumentation agents that run silently in the background on knowledge worker's computers, automatically gathering all the required information without disrupting users. We address the following research questions:

*Can passive instrumentation agents continuously capture all features necessary for state-of-the-art task-centric document curation in a collaborative multi-tasking environment?*

*What other aspects of sensemaking tasks, such as knowledge creation and collaborative intelligence, might it be possible to capture using passive instrumentation agents?*

Whilst these questions are our core focus, we also have to address some prerequisite issues in this chapter, such as how to store and analyze all the live streams of data from our instrumentation agents. We consider some architectural and design issues for doing this and conduct a brief performance evaluation on our chosen architecture. In the course of our research on instrumentation, we also struck upon several other applications of instrumentation data related to user workflows, such as next-action prediction, and analysis of web search behavior. These and other use cases are also briefly described.

Our general approach to this was to create a single instrumentation agent for the macOS operating system that was able to provide a holistic (but high-level) view of user activity, as well as a platform-independent agent for Chrome web browsers. Using this combination, alongside a state-of-the-art analysis platform, we were able to obtain sufficiently granular data to support our use-cases, and our algorithm development.

1.3.3 Analysis of Screenshots (Chapter 4)

As an orthogonal approach to the event-based (or ‘log-based’) instrumentation described in Chapter 3, in Chapter 4 we explore the potential of instrumentation from desktop screenshots. We explore the hypotheses that, from analysis of screenshots, it may be possible not only to replicate some forms of passive event-based instrumentation, but also to go a step further and extract finer-grained information relating to a user’s sensemaking process. In particular, we focus on automated extraction of analyst ‘claims’ in written reports, and on automated extraction of any associated ‘evidence’ from web pages that they might be using to justify these claims. In this chapter we address the following two research questions:
Can some or all of the features extracted by passive instrumentation agents be extracted (or at least approximated) from analysis of desktop screenshots?

Is it possible to accurately and reliably associate analyst ‘claims’ in written reports with associated ‘evidence’ from web pages, at least in a controlled setting?

These questions are tackled using a large corpus of screenshot data obtained from a series of controlled experiments conducted by the Laboratory for Analytic Sciences and the College of Humanities and Social Sciences (CHASS). Various methods are explored for identifying and extracting relevant claims and evidence, and for associating them using a variety of text similarity methods. These methods are verified through a human evaluation with expert analysts.

This work represents early inroads into this novel approach to instrumentation, but promising results are obtained and possible future directions are discussed.

1.3.4 Active Journaling Interfaces (Chapter 5)

Finally, we tackle the question of how to systematically obtain ‘ground-truth’ information on tasks that knowledge workers are engaged in, and how to do so in a way that users find acceptable (or even beneficial) on an ongoing basis.

We encapsulate this phase of the work in the following research question:

Can active journaling interfaces be created to capture task labels for individual and collaborative user activities, and to provide continuous benefits to users that successfully offset any disruption?

In answering this question, we consider the case where we want to capture labels at the time the activity is taking place, and also how to allow corrections (and new labels) to be supplied at a later date or time. We also tackle the question of how to define a taxonomy of tasks, and how to allow this to be easily modified so as not to become constraining.

Our eventual approach involves a simple but effective ‘task-tree’ visualization that is automatically triggered at appropriate times when ground-truth labels are needed. We evaluate our interfaces with groups of students and analysts using a design-based research approach. At the same time, we use these evaluation trials to capture unique labelled document corpora for development of our document curation algorithms.

In this chapter, we also briefly consider the ‘human factors’ of instrumentation and journaling, including the inevitable trade-off between privacy invasion and potential benefits. We conclude by presenting user feedback on potential future directions for this technology.
1.4 Publications

This section lists publications associated with or enabled by the work described in this dissertation. In the list below, * indicates lead author papers, some of the content from which makes up parts of this dissertation. Text content and figures are reproduced with permission from the IEEE and ACM, and references to the original published works are included below.

1.4.1 Published

1. *Title: Towards Automatic Linkage of Knowledge Worker’s Claims with Associated Evidence from Screenshots [Jon17b]
Authors: Paul Jones, Dakota Medd, Sreekanth Ramakrishnan, Rajat Shah, Joann Keyton, Nagiza Samatova

2. *Title: A Network-Fusion Guided Dashboard Interface for Task-Centric Document Curation [Jon17a]
Publication: 22nd Annual ACM International Conference on Intelligent User Interfaces (IUI) 2017.
Authors: Paul Jones, Shivani Sharma, Changsung Moon, Nagiza Samatova.

3. *Title: Journaling Interfaces to Support Knowledge Workers in their Collaborative Tasks and Goals [Jon16b]
Authors: Paul Jones, Sidharth Thakur, Steve Cox, Michael Matthews, Sean Streck, Chris Kampe, Preetham Srinath, Nagiza Samatova.

4. *Title: A Versatile Platform for Instrumentation of Knowledge Worker’s Computers to Improve Information Analysis [Jon16a]
Authors: Paul Jones, Sidharth Thakur, Steve Cox, Michael Matthews.

5. Title: A Hybrid CNN-RNN Alignment Model for Phrase-Aware Sentence Classification [Hsu17]
Authors: Shiou Tian Hsu, Changsung Moon, Paul Jones, Nagiza Samatova.
6. Title: *Online Prediction of User Actions through an Ensemble Vote from Vector Representation and Frequency Analysis Models* [Moo16]
   Publication: SIAM International Conference on Data Mining (SDM) 2016.
   Authors: Changsung Moon, Dakota Medd, **Paul Jones**, Steve Harenberg, William Oxbury, Nagiza Samatova.

7. Title: *Enabling Self-Awareness for Knowledge Workers Through Visualization of Instrumentation Data* [Tha15]
   Publication: Workshop on Personal Visualization, IEEE Visualization Conference (Vis) 2015.
   Authors: Sidharth Thakur, **Paul Jones**, Steve Cox.

8. Title: *Mining Aspect-Specific Opinions from Online Reviews Using a Latent Embedding Structured Topic Model*
   Publication: Conference on Computational Linguistics and Natural Language Processing (CICLing) 2017.
   Authors: Mingyang Xu, Ruixin Yang, **Paul Jones**, and Nagiza Samatova.

1.4.2 Accepted

9. Title: *Understanding Web Search Behavior for Analytical Tasks*
   Publication: Journal of the International Association of Law Enforcement Intelligence Analysts (IALEIA) 2017.
   Authors: Joann Keyton, Grant Harned, **Paul Jones**, Sean Streck.

10. Title: *Bringing NSA into the Classroom: Lessons Learned from Studying Student Workflows*
    Authors: Christopher Kampe, Gwendolynne Reid, **Paul Jones**, Colleen Stacey, Sean Streck, Kathleen Vogel

1.4.3 Under Submission

11. Title: *Real Time Utility-Based Recommendation for Revenue Optimization via An Adaptive Online Top-K High Utility Itemsets Mining Model*
    Authors: Ruixin Yang, Mingyang Xu, **Paul Jones**, Nagiza Samatova.
1.5 Human Subject Participation

Much of the work in this dissertation necessarily involved gathering user workflow data and conducting studies with human subjects. The relevant NCSU human subject protocol (e-IRB) protocol numbers that authorized or exempted this research are as follows:

- **3996**: CHASS Instrumentation Experiment in 2014, as described in section 3.6.3; also includes the capture of screenshot data as described in section 4.4. An addendum to this IRB covers the public release of portions of the data corpus.¹ (Principal Investigator: Dr Joann Keyton).

- **5521**: Task Journalling Behaviors and Goals - interviews to gather human feedback on journaling prototypes, as described in sections 5.7.2 and 5.7.4. (Principle Investigator: Dr Chris Anson)

- **6195**: Evaluating the Effects of Journaling Software on Student Driven Analysis in the Political Science Classroom, covering the evaluation in section 5.7.3. (Principle Investigator: Dr Kathleen Vogel)

- **9012**: Big Data and Open Source Analysis: Social, Ethical, and Policy Issues, covering some of the issues described in section 1.1 and section 5.9. (Principle Investigator: Dr Kathleen Vogel)

- **9136**: New CHASS Instrumentation Experiment in 2016 - a follow-up instrumentation study to the one in 2014, as described in section 3.8. (Principle Investigator: Dr Joann Keyton)

- **9222**: Recent-work dashboard and Journaling evaluation studies with Computer Science students and LAS staff, covering some motivating research in section 1.1, data collection used in section 3.6.1, and the evaluations in sections 5.7.2 and 5.7.4. (Principle Investigator: Dr Nagiza Samatova)

¹filtered data from this experiment is available for download at: http://go.ncsu.edu/cic
2.1 Problem Statement

From our survey in Chapter 1, we learned that future systems that incorporate ‘smart digital assistant’ technologies will need to be able to provide task-centric assistance to users. For such systems to be minimally disruptive to users, they will need to automatically learn associations between information artifacts and tasks, and to organize them accordingly (in user interfaces such as the ‘recent-work dashboard’). We refer to this combined association and organizing capability as ‘task-centric document curation’.

Also recall from Chapter 1 that a core challenge for a task-centric document curation system is for computers to quickly and accurately predict task associations for each information artifact (henceforth referred to as simply a ‘document’) that is accessed by a knowledge worker. Some of the sub-challenges involved in this are as follows:

- although some human-provided task labels will be available, they can be assumed to be sparse and infrequent. If a long period of time has passed since the most recent label (perhaps a few hours), confidence levels in new predictions may be lower since it is more likely that the user has switched tasks in the mean time.

- task predictions must be made in close to real-time, so that documents get added to the correct section of the dashboard in a timely manner. Without this capability, the utility of the
dashboard in saving user’s time when resuming tasks will be greatly reduced. This means that
the transformation step from document to features must also be done in real-time.

- there are many factors that may influence the association between a document and a task. Clearly, the document contents (such as the topic distribution) may provide a good clue but often tasks involve a heterogeneous collection of documents. The sequence of tasks that a user has been working on in the recent past also provides a clue, as does the overall distribution of time spent on each of their tasks (their underlying workflow patterns). All three of these types of information could be taken into account when making task classification predictions.

- it cannot be assumed that document-task associations are unilateral or exclusive - a given document may be associated with many different tasks, and a given task may be associated with many different documents. In a similar way, certain documents are related to each other (e.g. from topic similarity), as are tasks.

- document curation must also be user-centric, as well as task-centric. Users may have very different workflows, and associations may be subjective. Updates must be made to each user’s dashboard in real-time, which may constrain the types of analytics that are possible.

2.1.1 Formulation

Let \( U = \{u_1, u_2, \ldots \} \) be a set of users of the dashboard interface, \( Y = \{y_1, y_2, \ldots \} \) be a set of task labels, and \( D = \{d_1, d_2, \ldots \} \) be a set of documents. All of these sets are unbounded - new users, task labels or documents can appear at any time while the system is running.

During the course of their work, users will access documents and may or may not manually apply task labels. This will generate a sequence of ‘document access events’ for each user. Each event (for example, the \( j^{th} \) event for user \( u_i \)) will contain a timestamp \( t_{ij} \), a document name \( d_{ij} \in D \) and, optionally, a task label \( y_{ij} \in Y \) and an associated confidence \( c_{ij} \). For user \( u_i \), let this sequence of events be denoted as \( A_i = (a_{i1}, a_{i2}, \ldots, a_{in}) \), where each \( a_{ij} \) is a quad \( \{t_{ij}, d_{ij} \in D, y_{ij} \in Y, c_{ij}\} \). \( a_{in} \) is the most recently observed document access event for user \( u_i \).

At any given time, the system may be assumed to have access to the previous \( k + 1 \) documents (from a total of \( n \) documents) that have so far been accessed by a given user, \( u_i \), during the course of their work. The most recently accessed document is denoted as \( d_{ni} \) and the next document that they may access is denoted as \( d_{n+1} \). This forms the following sequence for each user \( u_i \in U \):

\[
  u_i : (d_{n-k}, d_{n-k+1}, d_{n-k+2}, \ldots, d_{n-1}, d_n, d_{n+1})
\]

Note that these documents are not necessarily unique - the same document may be accessed multiple times. Each document also has an associated task label, the most recent being \( y_{ni} \). There is
also a confidence value associated with the document-task pairing $c_{in}$. A given document may have different task labels and different confidences at each position in the sequence.

Furthermore, the time that the knowledge worker spends accessing (either viewing or editing) a given document, the dwell time, is also available from instrumentation and is denoted as $t_n$ for the most recent document. The periods of time in between document accesses (i.e. the 'gaps') are also available, the most recent being $g_n$.

This sequence is illustrated for two users in Figure 2.1, which forms the input to our task-centric document curation system. Note that the dwell times and gaps are omitted from the diagram for clarity. Given this document sequence, two key research challenges present themselves:

- **Sequence prediction problem** - to what extent can the identity of the next document, $d_{n+1}$, be predicted in the sequence $A_i$ for user $u_i$, either with or without the presence of task labels and confidences? The case without task labels or confidences was explored in [Moo16] and is not the focus of this dissertation. However, ideas from that work can be re-applied for predicting the next task label $y_{n+1}$ given a sequence of previous task labels, irrespective of the document itself. These ideas are outlined in section 2.2.3.

- **Task-centric document curation** - to what extent can the task label, $y_{n+1}$ for a newly observed document, $d_{n+1}$ be predicted, along with a confidence value $c_{n+1}$? Also, to what extent can missing task labels from earlier in the sequence be automatically added at a later time? These prediction problems can take into account context from the previous $k+1$ documents for each

---

**Figure 2.1** Sequence of document accesses for users $u_1$ and $u_2$. Colored headers denote different task labels, with unlabelled documents shown in white. The most recently viewed document for $u_1$ is denoted $d_{1n}$, and it has associated task label $y_{1n} = y_3$ (blue header), which has been assigned with confidence $c_{1n}$ (depicted as different levels of gray shading). The most recent document is accessed at time $t_n$ (in Time bin 4). For the sequence prediction problem, we are interested in guessing the identity of the next document $d_{n+1}$ [Moo16]. For the task-centric document curation problem, we assume we are given a new document $d_{n+1}$, and we want to predict its task label $y_{n+1}$ and associated confidence $c_{n+1}$. Image reproduced from [Jon17a] with permission. ©ACM 2017.
user, as well as their associated task labels and confidences. Furthermore, we can use associations between users \( u_i \in U \) and any known associations between tasks \( y_i \in Y \) to improve the predicted task-document associations.

The task-centric document curation problem is the focus of this chapter.

2.2 Related Work

This section comprises a survey of related approaches in five areas that are relevant to the task-centric document curation problem. Firstly, previous approaches to detection of task switches and task prediction are considered, and it is explained why the task-centric document curation formulation described here is a more tractable (and hopefully useful) angle on this problem. Secondly, state-of-the-art approaches to measuring document similarity are described, which is a core component of our formulation; thirdly, approaches to predicting user tasks by sequence analysis are considered; fourthly, an analysis of heterogeneous network fusion approaches is carried out, with a view to combining information on users, tasks and documents in a single graph; finally, approaches to fusing the classifiers derived from these approaches are considered.

2.2.1 Task Switch Detection and Task Prediction

First, we consider general approaches to task switch detection and prediction that have been proposed by other researchers. These projects tend to have somewhat different aims to that of ours but they all include elements that can be learned from, or built upon. Some attempt to learn about tasks in an unsupervised manner (i.e. without any manually provided task labels) while others take supervised or semi-supervised approaches; some are designed for real-time operation, others only learn in a batch mode; and each project exploits a slightly different set of features.

A summary of the main characteristics of each of the eight approaches considered here is shown in Table 2.1, and each of these approaches is briefly described below:

- **SWISH [Oli06]** - this approach from Microsoft was based on semantic analysis of window titles and switching history in order to detect task switches. Firstly, a model of user tasks is constructed based on terms in the title of application windows and in windows switching history. These terms are weighted by a TF-IDF filtering process, and a number of clusters of the filtered terms are isolated - the idea being that these clusters correspond to the user's tasks. Probabilistic Latent Semantic Indexing as well as switching history is used to create the clusters that correspond to the user's tasks. The prototype was only evaluated on 4 hours of real data and accuracy was limited to around 70%. However, this work did constitute a step towards solving the task-assignment problem in an unsupervised manner.
• CAAD [RC07] - again, the intention here was to create an automatic task support system that did not use explicit task labels. The authors used some of the same metadata we are providing through our Instrumentation (Chapter 3), including use of files, URLs and applications. This metadata was used to detect structures (called task context structures) from the user's workflows once per day. These offline computations were then used to inform a simple online model of document-task associations. Feedback gathered from user trials found several types of contradictions between the mental models participants had of their own work and the temporally correlated context structures captured by CAAD. Also the work was limited to a small subset of files on the user's computer. Whilst a good step forward, this work highlighted the need for a semi-supervised approach.

• TaskPredictor [She06] - similar to SWISH, this approach also exploited a variety of window-based text content, such as window titles, file pathnames and website URLs, for task switch detection. The algorithm began by segmenting the text from these into a set of ‘words’ and then created a binary variable in the feature set for each unique word. They then applied three simple machine learning techniques: (a) classification thresholds using Bayes rule, (b) mutual information feature selection, and (c) a hybrid Naive Bayes/Support Vector Machine classifier. An advantage of this approach was that the predictions could be computed immediately and rapidly each time the user switched focus from one window to another. The disadvantage was that, with such impoverished features, the accuracy of the predictions was poor, although they did show that accuracy could be improved by using features from windows other than just the one currently in focus. Other major limitations were that document content was not taken into account, and the model didn't make use of information from collaborators.

• Real-time detection of task switches [She07] - this approach built on the work carried out in TaskPredictor to implement a variant of change-point detection based on simple features from a Single Window Classifier. It was found that a metric based on an assumed task switch transition cost and the Viterbi algorithm gave the best results. This learning method was based on analyzing the sequence of recently-visited resources. It first applied the single-window classifier to each of these resource visits. Then it detected activity switches by applying a Viterbi algorithm with a fixed switch cost. This led to somewhat improved accuracy but had the side effect of introducing a potentially significant delay between the time of the switch and the time the switch was detected. This led in turn to major usability problems when implemented in the TaskTracer framework [Dra05].

• TaskPredictor2 [She09] - the purpose here was to both detect and correct user activity switches. Learning the lessons of [She06] and [She07], this was an improved variant that aimed to eliminate the SVM classifier and perform classifications in close to real-time. An additional draw-
back of both of the previous learning methods is that they employed a batch SVM algorithm. This required the storage of all of the training examples and they all had to be reprocessed when retraining was required. SVM training time was found to be roughly quadratic in the number of examples, so the longer TaskTracer was used, the slower the training became. Furthermore, activity prediction is a multi-class prediction problem over a potentially large number of classes. TaskPredictor2 included a richer set of contextual features and provided a better user experience but had a complex architecture. It adopted a ‘regularized passive-aggressive algorithm’ [Cra06] to ensure that, when user feedback was provided, the algorithm would be adjusted enough so that the same mistake would not be repeated.

• AUGUR [Har09] - the goal of this research system was to understand the user's current ‘context’ just enough to be able to provide useful recommendations for completing forms on the web. For instance, when booking a train, the system might provide suggestions for the ‘From’ field based on locations found in current calendar entries. Although the context referred to here doesn't correspond directly to a ‘task’, it does provide clues as to what the user is looking to achieve. Hence this research is relevant to systems that aim to provide task-centric support through UIs. The system used a graph-based model of relations between context and activities and suggestions are primarily based on a simple frequency count of edges connecting each of these types of nodes. The method also makes use of the FxL [HS07] sequence prediction algorithm for assigning confidence to form suggestions. Whilst limited to web forms, this work did demonstrate the potential benefits of this kind of pro-active support, with 85% of participants stating that they would like to see more of this kind of context-aware support.

• Deriving user tasks from document usage patterns [Brd10] (and patent [BA12]) - this approach for task recognition only used document identifiers and the temporal switch history between them as input. Unlike most other approaches, it didn't use any text from windows or from document contents. Retrieved documents were filtered by their dwell times (i.e. the amount of time a user spent studying them) and a document similarity matrix was estimated based on document usage frequencies and switches. A spectral clustering algorithm then grouped documents into tasks using the derived similarity matrix. Their results showed that worked well in identifying a small set of high-level tasks but the precision of the isolated clusters decreased as the number of tasks (and in particular, the granularity of tasks), increased. However, they did claim better results than content-based approaches like SWISH, and show the potential of creating initial task representations in an unsupervised manner, which can then be labelled and refined by users.

• Switch detector: an activity spotting system for desktop [Mir11] - finally, this approach simply used temporal aspects of user activity behavior to infer when it was most likely that an activity
Table 2.1 Comparison of main features of related task switch detection and task prediction research.

<table>
<thead>
<tr>
<th>Feature</th>
<th>[Oli06]</th>
<th>[RC07]</th>
<th>[She06]</th>
<th>[She07]</th>
<th>[She09]</th>
<th>[Har09]</th>
<th>[Brd10]</th>
<th>[Mir11]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch Detection</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Task Prediction/Labelling</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>(Semi)-supervised Training</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Used Content Features</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Used Sequence Features</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Used Temporal Features</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Real-time algorithm</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

switch was occurring, i.e. the $t_n$ and $g_n$ from Figure 2.1. The authors demonstrated that temporal aspects do provide some predictive power but, on their own, the prediction accuracy was relatively poor.

In summary, research in the field of task switch detection and prediction was particularly active between 2006 and 2011. Over the past 5 years, there seems to have been a slowing down in the pace of new research in this field. This does not appear to be because many of the problems have been solved - unfortunately, there are very few task recognition or context-aware recommendation systems in widespread use by knowledge workers. This lack of task-centric systems in the workplace suggests that the problems taken on above were too ambitious and solutions could not be found that were considered acceptably unintrusive and useful by users.

However, several developments in other research fields have come together to make this a potentially good time to re-examine this research. Before describing some of those, the main limitations of previous systems can be summarized as follows:

- most of these previous attempts did not make use of content-based features from inside documents when classifying documents across tasks - the ones that did use content features mostly just processed text from window tiles or web URLs (rather than from the document content itself). The field of document content analysis has moved on considerably in recent years, and it seems worthwhile to re-examine this approach in the light of state-of-the-art methods.

- few previous approaches took into account task or document sequence information, and those that did were using sequence prediction approaches from several years ago, that would not be considered state-of-the-art today.

- previous approaches did not take into account collaborating user’s documents and their task associations, nor did they take into account that certain users and certain tasks have similarities that can be exploited. These different types of heterogeneous relationships can naturally be represented in a network/graph model.
• only two of the approaches considered above made use of temporal information when detecting task switches or predicting task labels. The ones that did showed that there is potentially some value here, but only when combined with content and/or sequence-based features.

• previous approaches typically included complex feature selection procedures and hence could not be easily generalized to new desktop environments and new types of tasks. Furthermore, they could not easily be maintained or extended due to low-level instrumentation that was typically closely coupled to the underlying system.

As a result of these limitations, accuracy on task switch detection and prediction has typically been too low to be useful (rather than frustrating) for users, and solutions are difficult to generalize. The following sections examine state-of-the-art research that may help to address the first three of the limitations described - content features, sequence information, and heterogeneous relationships.

2.2.2 Document Content Analysis

As we have seen, many existing methods are limited by their minimal analysis of document content when attempting to cluster them, or to associate them with tasks. The types of content found in documents may not be relevant for some tasks but it does provide one potential approach to assigning task labels if there are one or more ‘topics’ associated with each task. This is especially likely to be the case in learning/educational tasks, where each sub-task likely corresponds to a slightly different subject to be learned.

There may be many other clues to be found in documents, such as in titles, filenames, folder locations etc but these are considered to be metadata for our purposes, not content. There follows a brief review of some of the methods used to examine document contents and to compute similarity between pairs or groups of documents:

• TF-IDF [SM83] - computing Term Frequency/Inverse Document Frequency weightings on word counts is probably one of the most well-known methods for producing a simple vector representation of a document. One simply counts words (optionally with prior stop-word removal and stemming), and computes a weighting for each word that represents its relative frequency in a wider corpus. The TF-IDF method is a simple, fast, and sometimes effective bag-of-words approach for creating document vectors. These vectors can then easily be compared using cosine similarity.

• LDA [Ble03] - Latent Dirichlet Allocation is a classic method for topic modelling of documents - it assumes that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the topics. The topic distributions are assumed to have priors
that correspond to Dirichlet distributions. LDA has been shown to produce reasonable topic distributions but unfortunately they are often seen as unintuitive by human observers. The topic distributions can be used directly as vector representations and hence support direct document similarity measurement but the number of topics must be known in advance, which is often not realistic.

- **word2vec** [Mik13] - this refers to a group of related models for producing vector representations of words. They use two-layer neural nets to learn contexts for words in large corpora (such as Wikipedia). The resulting word embeddings are useful for performing ‘computations’ over words along hidden semantic dimensions (for instance, the computation “Brother” - “Man” + “Woman” = “Sister” works because we’ve found a hidden dimension for ‘sibling’ words). Whilst these word embeddings are clearly effective, it is not clear how to combine word embeddings to represent entire documents - a simple approach is simply to average the word vectors but has been shown to work poorly in most circumstances.

- **doc2vec** [LM14] - this is a more recent algorithm that builds on the neural net representation approach of word2vec to create vector representations of short pieces of text. It is an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. doc2vec has been shown to overcome weakness of some bag-of-words approaches but it is very expensive to train.

- **LINE** [Tan15a] - it is possible to model text as a word co-occurrence network (where edges are created between words if they occur within short, say 5-word, sliding windows). The Large Scale Information Network embedding approach then computes first and second order proximities in this graph. First order proximity simply refers to weight on edges connecting two nodes; second order proximity refers to the similarity of their shared neighborhood. LINE is suitable for arbitrary types of information networks: undirected or directed, binary or weighted. It optimizes an objective function which aims to preserve both the local and global network structures, and is appropriate even when no labelled data is available, or when documents are short.

- **PTE** [Tan15b] - Predictive Text Embedding is a semi-supervised representation learning method for text data, which utilizes both labeled and unlabeled data to learn embeddings for chunks of text. Word co-occurrence information is represented as a large-scale heterogeneous text network (containing word-word edges, word-document edges and word-label edges), which is then embedded into a low dimensional space. This approach essentially extends LINE to deal with heterogeneous networks, in which multiple types of vertices (including the class labels) and edges exist. When the labeled data are abundant, on long documents, the authors
strongly suggest using the PTE model to jointly train the embedding with both the unlabeled and labeled data.

To summarize, for our problem, we require a semi-supervised method for learning vector representations that works on short documents as well as long ones. Often we may only get a small amount of usable text from web pages, or from documents that the user is creating themselves. Therefore it seems sensible to consider several of these algorithms for computing document similarity, including TF-IDF, word2vec (perhaps with vector averaging), doc2vec, LINE and PTE.

2.2.3 Task Sequence Prediction

As discussed in section 2.2.1, few existing research efforts have considered the sequence of tasks that multi-tasking knowledge workers are engaged in. The ones that did found that even simple heuristics can produce reasonable guesses as to the likely task that a worker is occupied with.

This approach assumes that a user is likely to continue working on tasks that they have worked on previously, and hence the sequence of past tasks holds some predictive power for guessing their current or future tasks. The advantage of this approach is that it is completely orthogonal to document similarity and does not depend on the tasks having any inherent 'topics' associated with them. Also, the problem of modeling a sequence of tasks, rather than a sequence of documents, may be easier due to the smaller number of possible classes to guess from - a person's task list is typically bounded.

This section reviews some sequence analysis methods that may be applicable for this problem, ranging from very simple to state-of-the-art. Note that we may not just want to predict a single task at any give time-step; instead, we can make $n$ predictions, and all of them can be used to organize documents in the dashboard interface (section 1.3.1).

- Static - in the simplest possible case, we could simply choose the previous $n$ tasks for the user as the top-$n$ predictions. This has been shown to work surprisingly well in related work [Moo16], and is certainly a valuable baseline algorithm.

- HMM - in a Hidden Markov Model, we assume that our sequence of tasks is a Markov process with hidden states. This makes the assumption that the next task depends only on the current hidden states of the model (which might correspond roughly to a user's workflow). However, in a related investigation [Moo16] where we were interested in predicting user ‘actions’ (application of web page opening events from instrumentation), a simple HMM (based on the Baum-Welch algorithm) performed poorly.

- RNN - a Recurrent Neural Network is a class of neural network that exhibits temporal behavior as a result of feedback between units. This allows them to easily maintain states using an
arbitrary number of previous inputs. A common implementation is in the form of a partially
recurrent neural network known as an Elman network. This network consists of three layers
(input, hidden, and output), with the addition of ‘context units’, which take their input from
hidden units. The number of context units and hidden units is the same. An advantage of El-
man nets over other forms of RNN is that the number of context units is not determined by
the output dimension but, instead, by the number of hidden units, which are easy to add and
remove. In our study of user actions [Moo16], an Elman network-based RNN performed badly,
although there is much room for further optimization here, such as by using Long-Short Term
Memory (LSTM) units [HS97] in the network.

• FxL [HS07] - this method calculates a score for each potential action (or task in this case) by
multiplying the frequency of an action (F) with the length of the context (L). FxL Builds upon
an n-gram trie containing the frequencies of different input sub-sequences. The frequencies
are weighted according to the lengths of the sub-sequences. This was the best performing al-
gorithm among the existing prominent algorithms.

• IPAM [DH98] - the Incremental Probabilistic Action Modeling method assumes that each ac-
tion depends only on the previous action. It stores the probabilities of transitions from an
action to a next action in a table, and the probabilities are updated by the use of an update
function. In our tests [Moo16], IPAM did not perform as well as FxL, although it is simple to
implement and fast.

• FVEC [Moo16] - the Frequency-Vector model is based on an ensemble prediction that com-
bines a frequency model with a vector representation, and beats either one in isolation. From
online performance tests on user actions [Moo16], FVEC showed a higher accuracy and a lower
standard deviation than the other online sequential prediction algorithms.

There appears not to be any related work in the literature on sequence prediction that also takes
into account time spent on each action and the gaps between actions (i.e. combining a prediction
from sequence analysis, with task switch detection from an approach such as the Switch Detector
[Mir11] approach in section 2.2.1). Such an approach may provide a fruitful avenue for future re-
search.

In summary, it would seem sensible to combine a sequence analysis approach (such as Static,
IPAM, FxL or FVEC) with a content analysis approach in order to address the task-centric document
curation problem.

2.2.4 Heterogeneous Network Fusion

In addition to document similarity and sequence analysis, other ‘clues’ to task associations are avail-
able that neither of these approaches take into account, such as associations between similar users
and similar tasks, and also document-task associations made by collaborating users. One approach to modeling all of these associations is as a network, or graph, in which edges represent associations between entities of the same or different types.

This section reviews some of the methods in the literature for creating such heterogeneous networks and for prediction of ‘missing’ edges in the network. These methods include the following:

- **Personalized Entity Recommendation [Yu14]** - although this is solving a slightly different problem to ours, the recommendation problem is closely related to that of link prediction. This work describes a heterogeneous information network approach, which incorporates a variety of different types of entity relationships, and uses a Bayesian ranking process for recommending entities. Latest features for users and items are calculated with matrix factorization techniques. This would be one possible approach for our problem, although matrix factorization is known not to scale well for large networks, and when using such techniques it isn’t possible only to consider a small part of the overall graph.

- **Neural Word Embedding [LG14]** - in related work to [Yu14], the authors show that skip-gram with negative-sampling [Mik13] is implicitly factorizing a word-context matrix. Skip-gram is a widely-used distributed word representation method. This insight suggests that it may not be necessary to factorize the entire matrix in order to compute similarity between nodes. Instead, we can create an embedding of each node in a shared vector space. Recall that we desire a means to build an embedding of users, tasks and their documents, so that the ‘similarity’ of any pair of these can be computed through a simple distance function. In this way, task association can be carried out by computing the distance of each task to a given new document.

- **DeepWalk [Per14]** - building on the insight that large-scale matrix factorization is undesirable, this work introduced skip-gram [Mik13] into the study of social networks for the first time. By applying skip-gram to random walks of a graph, DeepWalk learns vertex representations that correspond to the network structure - the idea being that nodes that frequently co-occur in short random walks should also be close to each other in the vector space. This method was shown to work well both for small and for large graphs, and may be a good approach to combine User, Task and Document networks. Also, this approach potentially lends itself well to an online algorithm since it is possible to consider only a subgraph when creating an embedding for a given node, and parameters such as the number of random walks, or skip-gram window size are configurable.

- **Network Representation Learning with Rich Text Information [Yan15]** - in this work, the authors prove that DeepWalk is actually equivalent to matrix factorization and then propose a ‘text-associated DeepWalk’ algorithm, which incorporates text features of vertices into representation learning under the framework of matrix factorization. This may provide a possible
method to take into account document content features in the association graph, when these features are available.

- Heterogeneous Network Embedding via Deep Architectures [Cha15] - in this work, the authors use deep learning techniques to capture the complex interactions between heterogeneous components. For our problem scenario, we don't need the additional complexity or computational expense that is incurred by this approach, as we can simply use the same generic 'association' relationship between each pair of entities. This approach may also require large networks for effective training.

- Multi-layer Networks [Kiv14] - in this paper, the authors describe many different variants of multi-layer networks. They examine the history of the field and why literature on multilayer networks has rapidly become 'extremely messy'. They present a general framework for studying multilayer networks and have constructed a dictionary of terminology to relate the numerous disparate existing notions to each other.

- Knowledge Graph Joint Embedding [Wan14] - this paper presents a novel method of jointly embedding entities and words into the same continuous vector space, which is comparable to, or perhaps slightly better than, word2vec (particularly the skip-gram variant). The embedding aims to preserve associations between entities in a knowledge graph and co-occurrences of words in the text corpus. Again, this might provide a means to model our workflow data but the richness of a knowledge graph model may not be required here.

- Review of Relational Machine Learning for Knowledge Graphs [Nic15] - to complete our brief review, this paper provides a summary of many approaches to construction and learning over Knowledge Graphs. Whilst this is a potential approach, we do not require the full expressivity of a knowledge graph model for this problem.

To summarize, a simple homogeneous network with node attributes (types) should actually suffice in this context, rather than a true heterogeneous network. Also DeepWalk with word2vec (skip-gram) could provide a relatively simple means to measure embed the different node types in a common vector space, from which similarity between different types of nodes can easily be computed using distance metrics.

2.2.5 Classifier Combination

Whilst a network fusion approach can take into account a lot of different relationship types between tasks, users and documents, it does not provide a natural means to incorporate document or task sequence information, as described in 2.2.3, or temporal information such as dwell times and gaps between document accesses.
With this in mind, there may still be a need to combine classification results from different classifiers. This section very briefly reviews some literature that may suggest approaches that can be taken. We first note that bagging and boosting can't be applied with different types of classifiers, so we require methods that can take into account predictions from orthogonal approaches. These include the following:

- Combining expert’s probability assessments [Jac95] - this paper compares Bayesian procedures and linear opinion pools (simple weighted combinations), for combining probability assessments from more than one ‘expert’ in a principled way.

- Dynamic Weighting [KB99] - this work discusses dynamic combination of classifier results using weightings. This provides a simple and intuitive approach but introduces additional parameters (the weights) that must be set either manually by users, or learned by the algorithm, perhaps from user feedback.

- Stacked Generalization [Wol92] - this work described a different way of combining multiple models that introduced the concept of a meta learner. Unlike bagging and boosting, ‘stacking’ can be used to combine models of different types. It attempts to learn the bias of each of the separate classifiers and use these to learn a higher-level classifier.

In summary, stacked generalization could be the best fit for our approach of combining a heterogeneous network fusion algorithm with a sequential model, and possibly other classifiers (e.g. from temporal analysis, or from workflow priors). However, it might be possible to find a simpler way to combine classifiers (such as simply resorting to a back-up sequence approach if the confidence from the network-fusion approach is below a threshold).

2.3 Major Research Contributions

In addressing the problem of task-centric document curation, the following research contributions are made:

- A formulation of the task-centric document curation problem is provided (section 2.1.1), and a survey of current state-of-the-art approaches for document-task association from the literature (section 2.2) is conducted. These related approaches inform the design of a new class of document curation algorithms for this setting.

- A characterization of user multi-tasking behavior from analysis of three real-world datasets (section 2.4), and use of this characterization to build a simple baseline document curation algorithm (section 2.5.1) that exploits the observation that users tend to spend a certain amount of time focussed on a given task before moving on to the next.
Demonstrations that document similarity and task sequence approaches are by themselves poorly suited for addressing this problem (sections 2.5.2 and 2.5.3 respectively), which motivates the need for a novel approach.

Demonstration of a new approach to task-centric document curation based on a multi-layer graph, which combines nodes representing Users, Tasks and Documents, and allows them to be represented in a common vector space (section 2.5.4). Furthermore, we demonstrate several ways to optimize the vector representations that can be obtained (section 2.5.5).

An empirical evaluation of these algorithms (section 2.6) against three labelled datasets created from Journaling trials described in a subsequent chapter (section 5.7). The evaluation is conducted in both batch and streaming modes of operation.

These contributions have enabled our best-performing streaming task-centric document curation algorithm to be incorporated into the recent-work dashboard described in section 1.3.1. The dashboard is driven by data from passive instrumentation and active journaling interfaces described in Chapters 3 and 5 respectively, and is currently being trialled with analyst users.

### 2.4 User Behavior Characterization

This section describes some observations of user multi-tasking behavior from three instrumentation and journaling studies. This will inform aspects of the algorithm design in section 2.5.

#### 2.4.1 Datasets

Three studies were conducted in 2015 using groups of students at NCSU and professional analysts. The purpose of the studies was twofold: (1) to evaluate the instrumentation and journaling prototype tools described in chapters 3 and 5 of this dissertation, and (2) to gather initial corpora of data for analysis of user behaviors and for development of task-centric document curation algorithms. It is from the second of these objectives that it is possible to derive some initial insights into user behavior that could steer research on algorithmic approaches.

The studies are described in more detail in section 5.7 but the main characteristics of each study are presented in Table 2.2. In all three cases, users manually tagged documents (including web-based documents) with task labels from their ‘task-tree’ (see section 5.4). The number of task labels provided by users in each study varied considerably - from an average of 77 per student in Study A (the Computer Science class), through to 630 in Study C (the group of Intelligence Analysts). A small number of participants contributed less than one label per week, and were discounted from the analysis (considered ‘inactive’).
Table 2.2 Basic characterization of datasets gathered during Journaling trials described in section 5.7.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Study A: CS Class</th>
<th>Study B: PS Class</th>
<th>Study C: Analysts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of active users</td>
<td>17</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>Duration of study (days)</td>
<td>133</td>
<td>90</td>
<td>118</td>
</tr>
<tr>
<td>Document-based events obtained</td>
<td>6212 (365)</td>
<td>5035 (504)</td>
<td>6707 (1118)</td>
</tr>
<tr>
<td>(Average per user)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document-based events with task labels</td>
<td>1308 (77)</td>
<td>4096 (410)</td>
<td>3780 (630)</td>
</tr>
<tr>
<td>(Average per user)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unique labelled documents, ( d_n )</td>
<td>1085</td>
<td>916</td>
<td>2536</td>
</tr>
<tr>
<td>Unique task labels, ( y_n )</td>
<td>75</td>
<td>20</td>
<td>33</td>
</tr>
<tr>
<td>Average team size</td>
<td>1</td>
<td>3.4</td>
<td>2</td>
</tr>
</tbody>
</table>

2.4.2 Observations

It can be seen from Table 2.2 that, although each study was conducted over a similar time period (of approximately 3 months), the number of task labels we made available to each set of participants varied considerably - for Study A, 75 task labels were available, whereas a simpler task tree of only 20 labels was used for Study B. An interesting question (one of many considered in Chapter 5) is whether this will have affected the level of user participation in providing task labels - do users get overwhelmed by too granular a task-tree?

For now though, we would like to understand some of the different multi-tasking behaviors present in the data. In study A, typically 2-3 ‘homework’ projects or tutorials were assigned at any given time, and it would have been possible for students to ‘multi-task’ on these. In study B, students were steered through a more linear flow of tasks (according to a structured analytic workflow described in chapter 5), so they would be expected to exhibit less task switching behavior. The analysts in study C were working on a variety of tasks, which were deliberately left unstructured (at least from the perspective of the study).

To get a sense of the kinds of multi-tasking behaviors that are visible in the data, Figure 2.2 shows dispersion plots for eight of the users in study A. Each vertical bar represents a labelled instance of a particular task, which corresponds to one of 14 possible tasks. The horizontal axis represents the event time, which spanned approximately 4 months. It can be seen that some users (such as users 1 and 6) seemed to do their homework tasks in ‘bursts’, perhaps close to the mid-term and end-of-term deadlines. Other user’s data was better distributed throughout the semester. It is also possible that some users turned off the instrumentation at various points, so we may not have a complete record but, what is most evident, is that users were frequently multi-tasking between four or more different tasks within a short time period.
Figure 2.2 Observed task switching behavior for 8 users in Study A. In each plot, the x-axis represents the time span of the study; the y-axis represents task numbers. Different working patterns can be observed. For instance, User 1 appears to be completing tasks in a fairly sequential manner; User 6 appears to be completing tasks in ‘batches’.

Next, a simple analysis was carried out to try to determine the impact of temporal gaps between document accesses (denoted $g_n$ from section 2.1.1), and the associated probability of a task switch. The results are shown in Figure 2.3 for all three studies. It can be seen that, with a gap between document accesses of around 20 minutes (approximately $10^3$ seconds), the probability that two distinct documents are associated with the same task seems to decrease markedly. This effect is less noticeable for study C (the real-world analysts), due to the reduced levels of task stability overall, which is indicative of both increased multi-tasking behavior and an increased propensity for each document to be associated with multiple tasks, which could have led to many ‘false’ task switches in this analysis.

Finally, we observe that the studies consist of a mixture of mostly individual-based working in Study A, where most tasks corresponded to individual homework projects, and mostly team-based working in Study B, where the participants were split into two groups of 3 and one group of 4. This
allowed us to observe collaborative working patterns, which are explored in section 5.7.3.

In summary, although this is a limited initial analysis, we have seen that most people tend to focus on a particular task for approximately 20 minutes, after which time the probability of switch dramatically increases. As an aside, this agrees well with the underlying observation of the Pomodoro [Cir06] method, which was used as the basis for the design of the ‘task-tree’ described in section 5.4. Furthermore, we have seen from dispersion plots that users tended to be working on up to 4-5 tasks in a particular time period, and different users take very different approaches towards completing their tasks. All this indicates that a simple algorithm for detecting task switches based only on temporal patterns is unlikely to suffice for our needs.

2.5 Algorithmic Approach

In this section, we propose a novel mathematical approach for the task-centric document curation problem by building on the related work in section 2.2 and by taking into account our unique context described in section 2.4. We describe how we build upon these to create a generic approach for task-centric document classification.
2.5.1 Simplest Baseline

Based on the principle observation in section 2.4 that users tend to remain focussed on a given task for a certain period of time, a very simple baseline algorithm can be constructed that just guesses that task labels do not change. However, as time since the last known label increases, confidence in the predicted labels should decrease. Below a certain confidence threshold, it may be better not to present a predicted label to the user at all, since having to correct an erroneous label is likely to be more annoying than not applying one at all (i.e. simply adding the document to the UNTAGGED section of the dashboard).

Such a procedure is outlined in Algorithm 1. Confidence is decreased exponentially since the time of the last known label - this provides the desirable property that each known label has a certain ‘half life’, which can be varied (and perhaps learned) for each user. The confidence threshold can also be varied for each user, depending on their tolerance to classification mistakes.

Note that this baseline classification algorithm is closely related to the ‘Static’ sequence-based approach considered and evaluated in section 2.5.3. Next, we consider how we might be able to build more sophisticated approaches that take into account the many other available features.

**Algorithm 1** Task Association Baseline Algorithm - assumes no change in task but decreases confidence exponentially (with parameter $a$) using the time since the last known label ($t_{\text{current}} - t_{\text{known}}$). If confidence falls below a threshold, return a special UNTAGGED label.

1. **function** PredictLabelAndConfidence($d_{n+1}, y_{n+1}$) 

   ▷ INPUT: $d_{n+1}$: A new information artifact (document)  
   ▷ INPUT: $y_{n+1}$: A task label for the new document (if present - this will usually be NULL)  
   ▷ OUTPUT: $y_{n+1}$: A predicted label for the new document  
   ▷ OUTPUT: $c_{n+1}$: A confidence value for the predicted label

2: \[ \text{if } y_{n+1} \neq \text{NULL then } c_{n+1} = 1, t_{\text{known}} = t_{\text{current}} \]

3: \[ \text{else } c_{n+1} = c_n \cdot e^{-a(t_{\text{current}} - t_{\text{known}})} \]

4: \[ \text{if } c_{n+1} \geq \text{threshold then } y_{n+1} = y_n \]

5: \[ \text{else } y_{n+1} = \text{UNTAGGED} \]

6: \[ \text{end if} \]

7: \[ \text{return } y_{n+1}, c_{n+1} \]

8: \[ \text{end function} \]
2.5.2 Classification by Document Similarity

For tasks that are topic-related, it may be possible to directly associate task labels from document similarity methods.

As discussed in section 2.2.2, the field of document classification based on content is large, diverse and active. As we saw, for classifying documents using standard machine learning techniques, it is helpful to create a vector representation of the document. Once such a vector is created, it is easy to calculate similarity between documents (using a variety of well-known distance measures), and hence to classify them.

We saw that a well-known popular and simple technique for creating document vectors based on their key terms is Term Frequency-Inverse Document Frequency [SM83], which allows a simple vector representation of documents to be constructed from the frequency of terms in a given document relative to the frequency in the overall corpus.

Figure 2.4 shows an example of how this process of creating document vectors and calculating the ‘similarity’ between them can render the documents easily amenable to classification. The figure shows t-SNE [MH08a] embeddings derived from vector representations of the content of 270 web pages associated with study B (the Political Science class), as described in section 2.4. Each document roughly corresponded to one of the three topics that the class was asked to investigate. More detailed information on how this data was collected can be found in section 5.7.

Given such a vector representation, it is easy to see how standard classification algorithms such as k-Nearest-Neighbors, Naive Bayes (NB) and Random Forest can be applied. We ran a brief study using document vector representations alone to classify the web pages from study B. The results from this are shown in Table 2.3, in which we show accuracy from runs with 60-20-20 train-validation-test splits of the data, with 10-fold cross-validation. These results show that, for task-related topics at least, there is some value in classification by document content alone, although accuracies of around 0.6-0.7 are probably not good enough to provide a satisfying experience to users. Also, it seems that the simpler vector representations (of simple word counts and TF-IDF weightings) consistently outperform the more expensive doc2vec embedding method - this may be because the document vectors used word vectors that were trained on Wikipedia data, which may not relate well to the types of news-oriented web pages found in study B.

This preliminary study indicates that document content is likely to provide a useful feature for task-centric document curation but, by itself, is not enough. An R-Shiny visualization\(^1\) was created to allow the reader to further explore the document vector embeddings methods described in this section.

In addition to those classification algorithms tested in Table 2.3, an alternative graph-based approach can also be constructed by observing that a graphical representation of the document corpus

\(^1\)https://las-goal-shiny.oscar.ncsu.edu/pjones/journaling-ps598-url-embeddings/
Figure 2.4 Two-dimensional t-SNE [MH08a] embedding of human-readable content from 270 web pages tagged by the Political Science class during the second Journaling trial (Study B). ASCII text was extracted from the pages using Diffbot (http://www.diffbot.com) and only those pages with >1kB of content retained. Stop-word, whitespace and punctuation-removal was conducted followed by word stemming. Next, TF-IDF weights were obtained, and Cosine Similarity calculated between all pairs of documents, giving a 270x270 matrix, which was provided to t-SNE for 2D embedding. Colours indicate the student group to which each document was assigned (ISIS, DRONES or IRAN); those with missing labels are marked NONE. Reasonably well-defined clusters can be seen for the three student groups.

can be obtained by thresholding on document similarity and creating an ‘edge’ between documents when the threshold is exceeded. Community detection algorithms can be applied on the resulting
Table 2.3 Classification accuracy results using various document vector representations, and using several different classification algorithms against the Study B (PS class) dataset. These results show the average classification accuracy using the one-vs-all methodology for multi-class classification (assuming only one label per document), and 60/20/20 training/validation/test data split, with 10-fold cross validation. Optimum parameters were found using a Grid Search across the training data. *For the Naive Bayes classifier, a Multinomial variant was used on the TF-IDF vectors, and a Gaussian variant on the doc2vec vectors, since the latter contained negative values. **For the Random Forest classifier, the default option of 10 trees was used.

<table>
<thead>
<tr>
<th>Document Vector Representation</th>
<th>k-NN (k=5)</th>
<th>NaiveBayes*</th>
<th>RandomForest**</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple word counts</td>
<td>0.54</td>
<td>0.70</td>
<td>0.60</td>
</tr>
<tr>
<td>TF-IDF (Weighted word counts)</td>
<td>0.66</td>
<td>0.74</td>
<td>0.61</td>
</tr>
<tr>
<td>doc2vec (trained on Wikipedia)</td>
<td>0.48</td>
<td>0.43</td>
<td>0.44</td>
</tr>
</tbody>
</table>

graphs, and a ‘majority vote’ (or other metrics) can be calculated inside each community in order to classify new documents. An interactive visualization\(^2\) exploring this idea is also available.

2.5.3 Classification by Task Sequence Prediction

As discussed in section 2.2.3, there are a variety of approaches available to task sequence prediction, including simple ‘Static’ models, HMM, RNN, FxL, IPAM and FVEC [Moo16]. Any of these could be applied to the problem of predicting task sequences. It would appear to be especially fruitful to combine this with a ‘task switch’ detector based on analysis of dwell times and gaps between tasks [Mir11]. Some examples of the kinds of sequences we want to be able to model are shown in the dispersion plots in Figure 2.2.

However, many of these techniques (particularly HMM, RNN, FxL and FVEC) require large amounts of training data for each user, which wasn’t available for our datasets. Also, they cannot take into account collaborative user’s behavior, predicting only on the basis of past behavior for individual users.

Task prediction results from our three datasets from the Static and IPAM models are shown in Table 2.4, for the best case where correct feedback is always provided to the algorithm at each time step. Although these techniques appear to do surprisingly well, this is simply because they tend to assume that the user does not switch tasks. We ran each test on the original dataset and also on ‘deduped’ datasets, where the task switched every time. When such a task switch occurs, it can be seen that these methods perform poorly, especially the Static algorithm, which, by definition, always makes wrong predictions. Even IPAM and other more sophisticated sequence models cannot take into account other contextual ‘clues’ that may point to the correct task association.

\(^2\)https://las-goal-shiny.oscar.ncsu.edu/pjones/journaling-csc791-doc2vec-graphs-communities-optimize/
Table 2.4 Accuracy of next-task prediction for each document using simple sequence prediction techniques alone, and assuming 100% correct feedback at each time step (best case). Results are shown both on the original datasets and on a ‘de-duped’ dataset in which the task changes every time - the Static algorithm always gets it wrong in this case. For each value in the table, a test dataset was created with the last 100 document-task associations in the sequence. The IPAM algorithm was trained using the 200 associations prior to this. Note that not all users had enough data to meet this training requirement, so these results are based on a subset of (at least 2) of the users.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Static</th>
<th>Static (deduped data)</th>
<th>IPAM</th>
<th>IPAM (deduped data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study A</td>
<td>0.784</td>
<td>0.0</td>
<td>0.796</td>
<td>0.560</td>
</tr>
<tr>
<td>Study B</td>
<td>0.510</td>
<td>0.0</td>
<td>0.753</td>
<td>0.790</td>
</tr>
<tr>
<td>Study C</td>
<td>0.858</td>
<td>0.0</td>
<td>0.873</td>
<td>0.540</td>
</tr>
</tbody>
</table>

From analysis of the performance of these simple sequence models on their own, it is clear that they are unsuitable for providing consistently accurate predictions in an environment where feedback from users may be scarce. Furthermore, the priority here is to be able to predict effectively in environments where users are not only multi-tasking but also multi-tasking in inconsistent ways, as was found from the analysis in Figure 2.2. Simple sequence-based methods may work well when working patterns are routine but, in our environments, we must be able to take into account a variety of heterogeneous associations in order to make better predictions.

We briefly revisit sequence models in section 2.6.6 where we apply them as part of an ensemble with an orthogonal classification approach that is described next.

### 2.5.4 Classification from Heterogeneous Network Fusion

Recall from section 2.1, we seek an approach to the task-centric document curation problem that learns representations of users, tasks and documents, with the following characteristics:
Figure 2.5 Cumulative classification accuracy of streaming sequence prediction algorithms (Static and IPAM) on the Study B dataset, with varying probability of (correct) feedback provided to the algorithm after each time step. It can be seen that the IPAM model rapidly breaks down when the probability of feedback is low.

- **Heterogeneous** - can take into account a variety of different types of associations between entities, such as teams of users, task hierarchies, document similarity, and temporal proximity.

- **Multi-user** - can take into account task-document associations made by all users when making a prediction for any given user.

- **Probabilistic** - ability to associate confidence values for each tentative task-document association, which can be used to make top-N predictions for associations, and to decide whether to automatically organize a given document in a user's task workflow.

- **Adaptable** - new associations can be added to the model, and defunct/erroneous associations easily removed, without re-learning all entity representations. In an environment of frequent multi-tasking, mistakes will be made and should be handled gracefully.
• **Scalable** - must be able to handle both small (for instance, less than 10 documents), and large (e.g. thousands of documents) volumes of historic data related to any given task. It is particularly important to gracefully handle the cold-start problem when a previously unseen task is encountered.

• **Online** - ability to operate in a streaming environment, where entity representations can be learned one at a time, and resulting associations can be made in close to real-time.

After examining a variety of previous approaches to this problem in section 2.2.4, we propose a network fusion approach in which all currently known relationships between users, tasks and documents, as well as temporal relationships, can be expressed in a single graph. Furthermore, distributed representations (embeddings) of each entity can be constructed in a low-dimensional vector space. We refer to these representations as ‘embeddings’ because they are not directly observed in the data but instead are automatically inferred from the data. In this way, new associations can be discovered using simple distance metrics between embeddings for nodes of interest.

### 2.5.4.1 Multi-Layer Graph Embedding

In the simplest case, we construct a graph, $G = (V, E)$, in which the vertices (or nodes), $V = \{U \cup Y \cup D\}$, correspond to users, tasks and documents, and the edges, $E$, denote associations between vertices (see section 2.5.5). An illustration of such a graph is shown in Figure 2.6. We say that this is a layer-disjoint multilayer network [Kiv14] since each node exists in at most one layer.

The challenge of task-centric document curation is to associate each newly observed document $d_{n+1}$ for a user $u_i$ with a task label $y_{n+1}$, and to assign a confidence $c_{n+1}$ to the association. In order to achieve this, we need to derive a distance (or proximity) metric between the new document and existing tasks in a common vector space. Our approach to this is inspired by the DeepWalk [Per14] algorithm, which learns representations of graph nodes in such a space. We extend this idea to our multi-layer graph scenario, and proceed as follows:

- We first generate random walks of length $W$ from each node in $V$ by sampling neighbor vertices uniformly at random at each step of the walk. This forms a dataset, $S$, of task nodes and the ‘context’ that they appear in, which consists of user nodes, document nodes and other task nodes.

- We then use the SkipGram algorithm [Mik13] to efficiently create a distributed representation of each vertex, where the distance between each pair of vertices is inversely proportional to the co-occurrence probability in windows of length $w$ (where $w \leq W$). Skipgram tries to predict each of the context nodes, $h$, given a particular ‘target node’, $n$. It achieves this by maximizing the product of conditional probabilities of each context node, $h$, given the target node, $n$, and a set of vector embeddings, $\theta$, over the entire set of all target and context pairs, $S$: 

37
**Figure 2.6** Illustration of a multi-layer graph relating users, tasks and documents. Inter-layer edges are shown as broken lines; intra-layer edges (considered in section 2.5.5) are unbroken. The white highlighted node at the bottom represents a new document for which the system has a user association, and can create document associations (included in section 2.5.5) but currently has no direct task association. ©ACM 2017.

\[ \arg\max_{\theta} \prod_{(h,n) \in S} p(h|n; \theta) \]  

- The probability \( p(h|n; \theta) \) is computed using a softmax function as follows:

\[ p(h|n; \theta) = \frac{e^{v_h \cdot v_n}}{\sum_{h' \in H(n)} e^{v_{h'} \cdot v_n}} \]  

where \( v_h \in \mathbb{R}^d \) and \( v_n \in \mathbb{R}^d \) are the embedding vectors of \( h \) and \( n \), respectively in \( \theta \). \( H(n) \) is the set of all available context nodes for target node \( n \) that have occurred in windows of length \( w \). Unfortunately, it is too computationally expensive to compute this probability for all target nodes since we have to sum over all possible contexts, \( h' \).

- Instead, we generate a set \( S' \) of random \((h,n)\) pairs, which we assume will be incorrect. This is known as *negative sampling* [MD13], which refers to a set \( S' \) of randomly sampled negative examples. The optimization objective now becomes:

\[ \arg\max_{\theta} \left( \sum_{(h,n) \in S} \log p(S = 1|n,h; \theta) + \sum_{(h,n) \in S'} \log p(S = 0|n,h; \theta) \right) \]  

where \( p(S = 1|n,h; \theta) \) is the binary logistic regression probability of seeing the node \( n \) with the context \( h \) in the dataset \( S \), calculated in terms of parameters \( \theta \). Correspondingly, \( p(S = 0|n,h; \theta) = 1 - p(S = 1|n,h; \theta) \) is the probability that \( n \) has never appeared with the context \( h \) in \( S \). These probabilities are computed using a sigmoid function, which is fast to compute:
By choosing $k$ negative samples for each training sample, $|S'| = k|S|$, computing the loss function now scales with $k$ as opposed to all possible contexts, which speeds up training considerably. The node-embedding vectors, $\theta$, are updated iteratively using *Stochastic Gradient Descent* until they can discriminate the real target nodes, $n$, from the ‘noise’ nodes that have never been seen in the context, $h$.

- From this embedded representation, we can compute a ‘similarity’ score between each candidate task $y_i$ and the newly arrived document $d_{n+1}$. For this we use *Cosine similarity* as is common in language models. Finally, we rank these similarity scores to produce top-1 or top-N predictions for the task(s) to which to assign the user’s document.

- Note that, optionally, we can use the *Hierarchical Softmax* [MH09] algorithm to approximate a probability distribution over all the candidate tasks which, if well calibrated, allows us to estimate a confidence probability $c_{n+1}$ for each candidate task. These confidences can then be thresholded in order to assign the document to one or more tasks in the user’s workflow.

The objective function for *SkipGram* is based on the *distributional hypothesis* [Har54] that states that words in similar contexts tend to have similar meanings. In *DeepWalk*, the authors present a generalization of language modeling that treats walks in a graph of associations as analogous to sentences and phrases in a special language. They demonstrate that this analogy holds for several social-network datasets. While not a pre-requisite, an approximate power-law distribution of nodes in short random walks also holds for our collected datasets (Table 2.2), so we extend the ideas of *SkipGram* and *DeepWalk* to the context of a user’s workflow, in which users, tasks and documents are related to each other in potentially complex ways.

2.5.5 Enhancing the Node Embeddings

A helpful way to think about this approach is as follows: the distance between embeddings of the node for document $d_{n+1}$ and the node for a given task $y_i$ is inversely proportional to the probability of walking from node $d_{n+1}$ to node $y_i$ in $w$ steps. If these nodes are directly attached (i.e. the walk can be achieved in $w = 1$ steps), the probability is high and the distance is short. If the shortest path in the graph between the nodes is large, then the distance in vector space will be large too. Hence we need to construct the graph in such a way that the ‘correct’ task node is more connected to the target document node. These connections can occur through other users, tasks or documents. We now explore several options for achieving this, the results of which are presented in section 2.6.3.
Figure 2.7 Automatic construction of edges in the Task Layer using common task parents from a hierarchical ‘task tree’ shared between the users. ©ACM 2017.

**Task and User Associations**

User workflows are typically hierarchical in nature: projects are broken down into tasks, tasks into sub-tasks, and so on. Hence a simple method for modeling tasks is as a ‘tree’. We use the idea of a ‘task-tree’ [Jon16b] to represent tasks hierarchically. This has the desirable property that all tasks with a common ‘parent’ are likely to be related in some way. Hence, edges can be automatically added to the multi-layer graph to represent these implicit associations between tasks that can be extracted simply from a user’s goal tree (Figure 2.7). In a similar way, edges between users can be created when they are known to be on the same team, or when they are known to be collaborating on common tasks.

Furthermore, it is possible to create nodes in the graph that represent both Users and Tasks at the same time - we denote the set of these concatenated nodes as $\{U\|Y\}$. So now, the vertices in the multi-layer graph are $V = \{U \cup \{U\|Y\} \cup D\}$. The idea is that these concatenated nodes will allow different embeddings to be created for a given task to allow for the situation where each user accesses different documents to perform the same task. This removes the effect of ‘averaging’ the task contexts for all users, as would be the case in the simple multi-layer graph in Figure 2.6. It might be expected that these additional ‘User\|Task’ nodes would improve overall classification accuracy, while still allowing associations to be made with tasks that a given user has never worked on before (but their colleagues have).

**Document Similarity Methods**

Often documents related to a particular task share common topics, especially if the task is related to learning a particular skill or subject area (such as a homework task). In such cases, it makes sense to add edges to the graph to associate similar documents, so that they become more likely to appear in the same contexts as their associated tasks. Many methods exist for measuring document similarity - typically a vector representation is created so that proximity metrics (such as cosine sim-
ilarity) can be applied. Simple vector representations can be constructed from counting words and weighting according to the frequency in the overall corpus, such as in TF/IDF [SM83].

It is also common to represent documents as vectors of ‘topic proportions’, where the topic distributions can be obtained from LDA [Ble03] or other techniques. More recently, neural nets have been used to compute ‘document vectors’ [LM14]. The training step for these is prohibitively slow, especially for an online environment, but much faster alternatives, such as fasttext [Jou16] are emerging. Any of these techniques could be applied with proximity thresholding to create Document-Document edges in the multi-layer graph. We initially use TF/IDF in order to demonstrate the effect of these additional edges.

**Task Sequence Methods**

As we saw from section 2.5.3, an orthogonal approach to predicting task associations is simply to analyze the sequence of tasks that a user has worked on in the past and use that to guess the likely next task(s). Established approaches to this include IPAM [DH98], which assumes that each task depends only on the previous task, and FxL [HS07], which calculates a score for each potential task by multiplying the frequency of a task with the length of the context in which it is seen. Since then, an approach based on an ensemble of vector and frequency representations, FVEC [Moo16], has been demonstrated to outperform both IPAM and FxL.

While the output from these methods could be combined (or ensembled) with predictions from our network fusion approach in order to increase precision, it does not make sense to incorporate sequence information directly into the graph (for instance, as Task-to-Task edges). Adding such edges would likely bring different tasks together in the embedded space simply because they often occur in close temporal proximity when the user is multi-tasking. This is the opposite of what we want to do, which is to clearly separate the different tasks in the embedded space. So instead of exploring sequence-based methods in this work, we focus on how temporal characteristics of tasks can be taken into account in the graph.

**Temporal Analysis Methods**

In the same way that characteristics of a given task may vary across users, they may also vary across time. The documents required to complete a given task during its first week may be very different to those in its second week. For this reason, in the same way that we can split Task nodes into ‘User||Task’ nodes, we can also split Task nodes into ‘Task||Time’ nodes that represent a given task during a given time period. This concept of time binning is illustrated in Figure 2.1. The hypothesis is that these will be better spread across the ‘document space’ and lead to higher accuracy associations. Now we can try building the graph with nodes $V = \{U \cup \{Y||T\} \cup D\}$ and also $V = \{U \cup \{U||Y||T\} \cup D\}$,
where \( T \) represents a time bin.

**Biasing the Random Walks**

Some recent work on an algorithm called *node2vec* [GL16] attempted to achieve better results than *DeepWalk* by biasing the random walks according to two parameters: a ‘return parameter’, which controls the likelihood of immediately revisiting a node in the walk, and an ‘in-out parameter’, which controls the tendency to conduct a breadth-first (exploring all communities at a shallow level) versus depth-first search of the graph. This has the potential to improve the embeddings from our network fusion algorithm, particularly if large communities of related documents are present - we can use the ‘in-out parameter’ to ensure that random walks don’t get ‘stuck’ in such communities.

### 2.5.6 Combination of Classifiers

Detailed analysis of approaches to combining orthogonal classifiers, such as that from our heterogeneous network fusion approach in section 2.5.4 and from our sequence analysis approach in section 2.5.3, is beyond the scope of this dissertation. However, an initial survey of a few approaches that may provide initial direction was provided in section 2.2.5, and it appears that a stacked generalizer [Wol92] may be the most fruitful. There is also a good reason to keep these classifiers separate since we may want to assess which is doing better for each user, and adjust the weights of each accordingly. Another simple ensembling approach is simply to note that the network fusion algorithm outputs a confidence value along with every prediction. If this confidence is low, it may be sensible to revert to a simple sequence prediction approach (including to simply guess that the task has not changed) in preference to using a low confidence prediction.

### 2.5.7 Approaches Chosen for Evaluation

To summarize this section, the following approaches have been implemented and evaluated as part of this dissertation:

- Network Fusion model of User, Task and Document relations building on DeepWalk and Skip-Gram with Negative Sampling to create node embeddings (from section 2.5.4),
- Evaluation of impact of intra-layer edges (User-User, Task-Task and Document-Document), and of concatenated nodes (User||Task, Task||Time, and User||Task||Time) (sections 2.5.2 and 2.5.5),
- Evaluation of impact of biasing random walks using node2vec [GL16], rather than DeepWalk [Per14],
• Evaluation of algorithms in batch (offline) mode, against journaling datasets.

• Evaluation of algorithms in a streaming mode of operation, against journaling datasets.

• Combination of results from network fusion classifier with one or more sequence classification approaches (from section 2.5.3 and 2.5.6), using a simple ensemble.

Full results from these evaluations are now presented.

2.6 Empirical Evaluation

Our goal was to evaluate the network-fusion based document curation algorithm through a series of user studies. We first describe the data collected from these studies, before presenting empirical results from a variety of different configurations of the algorithm described in section 2.5.4.

2.6.1 Workflow datasets

To evaluate the task classification accuracy derived from our network fusion approach, we used three datasets collected in our group, as described in section 4.5. These were from a graduate Computer Science class (Study A), a graduate Political Science class (Study B), and from a small group of Intelligence Analysts (Study C). The main characteristics of these datasets as they relate to document curation are shown in Table 2.5.

In each study, users manually labelled documents and URLs from a fixed set of task labels using journaling [Jon16b] interfaces. In total, 25 users actively contributed to these studies, providing task labels for 16,040 documents (2,715 of which were unique). During each of the studies, the participants used between 36 and 59 different task labels, which corresponded to elements of their homework tasks and analytic workflows.

As well as differences in the number of active users, and in the number of unique task labels used for each study, there were also differences in how the task-trees were constructed for each group. In study A, each task closely corresponded to a 'homework topic', such as Bloom Filters or Count-Min sketches, and the documents used by the students matched these topics well. In studies B and C, the task trees corresponded to steps in a generic analysis workflow [DC15b], and there was a less direct relation between tasks and topics. For study B, we expanded the number of available tasks in the task tree (from 20 initially to 60, of which 36 were used) based on three teams.

2.6.2 Parameter Optimization and Calibration

The multi-layer graph embedding classifier described in section 2.5.4.1 was implemented in Python to conform to the Scikit-learn [Ped11] framework API. Parameter optimization was first carried out in
Table 2.5 Characteristics of our 3 workflow datasets for algorithm evaluation.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Study A</th>
<th>Study B</th>
<th>Study C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active participant users, $</td>
<td>U</td>
<td>$</td>
<td>10</td>
</tr>
<tr>
<td>Total documents labelled</td>
<td>6211</td>
<td>3122</td>
<td>6707</td>
</tr>
<tr>
<td>Unique docs labelled, $</td>
<td>D</td>
<td>$</td>
<td>1015</td>
</tr>
<tr>
<td>Levels in task hierarchy</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Unique task labels used, $</td>
<td>Y</td>
<td>$</td>
<td>59</td>
</tr>
</tbody>
</table>

To determine good values for the DeepWalk parameters, which included the number of random walks to take from each node, the length of each walk, and the window size for Skipgram contexts. To achieve this, a grid-search was carried out for each dataset using 3-fold cross-validation. As found in the original DeepWalk paper [Per14], accuracy improved as the number of walks was increased up to around 100 but remained stable beyond this. Furthermore, the optimum walk length was found to be around 40 and the window size around 10, although the measured accuracy was not very sensitive to these values. In each case, the vector representation size was fixed at 64, since no improvement was found using larger values in any of the tests. More effort could have been invested in further optimization but, in the interests of simplicity, these parameter values were used for all subsequent experiments and for all datasets.

Next, we performed experiments to measure the calibration of the confidence values output by the network fusion algorithm, to determine whether they can be interpreted as probability values. For example, if the confidence of a given task prediction is 0.9, we would like the label to be confirmed as correct by users roughly 9 out of 10 times. This is typically measured using a Brier score [Bri50], which computes the mean-squared error of probabilistic predictions, and is scaled to fall in a range from 0 to 1. Across all the datasets, we measured the accuracy of predictions for each task label considering each label as a ‘one vs rest’ binary classifier. We obtained a mean Brier score of 0.04, which indicates that the classifier is well calibrated. A further observation was that calibration is especially good for confidence values above approximately 0.8; below that, confidence values did not match empirical probabilities so well.

### 2.6.3 Batch-mode Evaluation Results

To provide a fine-grained analysis of the task classification precision and recall, we compare F1 measures for each dataset while varying the train-test split from 10% to 90%. In all datasets, the number of labelled samples for each task varies, producing somewhat imbalanced classes. To account for this, we characterize the performance using both Macro and Micro-averaged F1 scores. Macro-averaging gives equal weight to each class, whereas Micro-averaging gives equal weight to each per-document classification decision. The results are summarized in the plots in Figure 2.8.
Figure 2.8 Performance evaluation of the multi-layer graph classifier on varying the amount of labeled data used for training. In all plots, the x axis denotes the fraction of labeled data. The y axis in the top and bottom rows denote the Micro-F1 and Macro-F1 scores respectively (averaged over 5 runs using different labelled data in each). Performance is compared for the baseline algorithm, and for selected variants with additional types of edges and nodes described in section 2.5.5. Image reproduced from [Jon17a] with permission. ©ACM 2017.
Table 2.6 Percentage changes in accuracy from baseline, obtained from adding additional node and edge types described in section 2.5.5. Each value was obtained with 50% training data and averaged over 10 runs. Bold values show where the biggest improvement was obtained for each dataset.

<table>
<thead>
<tr>
<th>Node/Edge type</th>
<th>Study A</th>
<th>Study B</th>
<th>Study C</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-User edges</td>
<td>0.78</td>
<td>-1.93</td>
<td>0.30</td>
</tr>
<tr>
<td>Task-Task edges</td>
<td>1.00</td>
<td>-1.47</td>
<td>-3.70</td>
</tr>
<tr>
<td>Doc-Doc edges</td>
<td>1.84</td>
<td>-0.86</td>
<td>-0.58</td>
</tr>
<tr>
<td>User</td>
<td></td>
<td>Task nodes</td>
<td>8.21</td>
</tr>
<tr>
<td>Task</td>
<td></td>
<td>Time nodes</td>
<td>8.07</td>
</tr>
<tr>
<td>User</td>
<td></td>
<td>Task &amp; Task</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td></td>
<td>Task</td>
<td></td>
</tr>
</tbody>
</table>

The baseline results are obtained from a simple 3-layer graph like that shown in Figure 2.6 but without any intra-layer edges (i.e. none of the User-User, Task-Task or Document-Document similarity-based edges were included). Each of the possible modifications described in section 2.5.5 were made to the graph and new Macro- Micro-F1 results obtained. Only those changes that made an observable difference to the performance are included in Figure 2.8.

A snapshot of averaged accuracy results is shown in Table 4.3. It can be seen that addition of User-User edges made very little difference for any of the datasets. Adding some Task-Task edges gave a small improvement for Study A, where the document corpora was highly topic-based, but made things slightly worse for the other datasets. Likewise addition of Document-Document edges did not produce a noticeable increase in accuracy, except for a very small effect for Study A. The cosine similarity threshold for creation of edges between documents was chosen to be 0.7, which produced around 30 of these edges for each dataset. It was found that decreasing this threshold to add more edges tended to decrease the classification performance - the intuition being that more intra-document edges cause random walks to get ‘stuck’ in the document layer, thus reducing the quality of task node embeddings.

It can also be seen from both Figure 2.8 and Table 4.3 that adding concatenated ‘User||Task’ nodes (as described in section 2.5.5) does produce a considerable improvement in F1 scores and accuracy across all datasets. This increase can be explained intuitively from the t-SNE [MH08b] plot in Figure 2.9. It can be seen that the joint ‘User||Task’ nodes are much better distributed throughout the document space than the Task or User nodes alone. Indeed, the simple 3-layer graph depicted in Figure 2.6 has the effect of ‘averaging’ the task node embeddings across all user contexts. This may sometimes be desirable (especially for tasks for which there is a ‘best’ set of documents to use), since it allows tasks to be predicted for users that have never done that task before. However, for the current case studies, it is clear that better performance is obtained if each task is embedded separately for each user.
Similarly, concatenated ‘Task||Time’ nodes (as described in section 2.5.5) produced a noticeable improvement in Macro and Micro F1 and accuracy for all three studies. For these trials we used 2-week time bins. This verifies our hypothesis that such nodes are better for classification than the task nodes alone - the intuition being that document associations for each task tend to change over time. Perhaps surprisingly though, enabling ‘User||Task’ nodes as well as ‘Task||Time’ nodes did not produce a greater improvement than either one alone. Instead, we tried an alternative scheme using triple-concatenated nodes (i.e. made up of the set \( \{U||Y||T\} \)) to see if this would further improve the embeddings, and indeed this appeared to be the case across all datasets, with up to 20% accuracy gains being achieved.

To generate the results in Figure 2.8 and Table 4.3, concatenated nodes that corresponded to tasks that the user had been associated with before, and to tasks that were seen in the current time window, were considered as possible classification classes prior to other possible concatenated nodes. Running the algorithm without this additional step produced slightly lower overall accuracy and F1 scores, but with the benefit that a wider variety of tasks would regularly be included in predictions. Based on this, we intend to add an option to the dashboard to allow users to choose between these two slight algorithm variants (i.e. between associations that tend to be ‘safer’ since they are based primarily on their own behavior, and ‘riskier’ associations that take into account a wider variety of users and time periods).

Finally, it is possible to obtain more than one task association for each document since we can use a distance metric to obtain a ranked list of \( n_p \) associations. The value of \( n_p \) is a user-selected preference in the dashboard that allows users to trade-off missing associations for a potential increase in incorrect associations. As an illustration of this, a plot showing the overall probability of a correct task association is shown in Figure 2.11, as \( n_p \) is increased for each dataset. For \( n_p = 5 \), the ‘correct’ task is present in the guesses around 90% of the time for all datasets.

In order to evaluate our approach in a ‘batch’ context, we removed a proportion of labels from random positions in each dataset to create test sets for each run. Although this is a valid scenario for operation of the dashboard (i.e. the user works for a period of time labelling some documents manually but not others, then asks the system to fill in the blanks), this is not the same as a ‘streaming’ context, where missing labels are computed one at a time as soon as each document is accessed. A separate evaluation in this context is presented in section 2.6.5.

### 2.6.4 Effect of Biasing the Random Walks

As introduced in section 2.5.5, one approach to producing higher quality embeddings of nodes in the multi-layer graph may be to bias the random walks across the graph, rather than walking from node to node uniformly at random. The intuition is that it may be possible to produce more representative context datasets for embedding task nodes if there is a tendency for walks to return to the task layer.
(or to a 'concatenated node' layer) rather than 'getting stuck' in the document layer, which is likely to contain a larger number of nodes and connections.

In order to test this hypothesis, the node2vec [GL16] algorithm was implemented and run on our datasets using the optimal parameters found using a GridSearch across all datasets (In-Out Parameter, $p = 0.5$; Return parameter, $q = 1$; Feature Dimensionality, $d = 64$). Setting $p < 1$ creates a biased walk behavior that is reflective of a Breadth-First search, which encourages detailed exploration of local graph regions - this is what we want in our setting in order to encourage the algorithm to find effective context embeddings for particular tasks. Conversely, setting $q < 1$ creates a biased walk behavior that is reflective of Depth-First Search, which encourages outward exploration of the graph, as opposed to a tendency to focus on a local area. The results are shown in Figure 2.12, and can be compared with those in Figure 2.8.

It can be seen from Figure 2.12 that node2vec appears to make minimal difference to the F1 scores obtained for all three of the graph configurations tried (Baseline, 'User||Task||Time' nodes, and Doc-Doc edges). From this we can determine that biasing the random walks does not make a significant difference to the quality of the node embeddings, and hence the accuracy of task predictions, for our datasets. Even in the case where we added extra Doc-Doc edges, there was no appreciable effect of 'getting stuck' in the document layer, which biasing the random walks was able to alleviate.

This is a little surprising, since the authors of node2vec found that it improved F1 scores for multi-label classification on some datasets by around 20%, although they acknowledged that, for other datasets, the gain was close to zero. They also indicated that biasing the random walks is likely to make a bigger difference on very large graphs, where it is typically not possible to walk the entire graph when creating a context dataset from a particular target node (so the walk biasing ensures that a more useful portion of the graph is explored - typically that obtained from a breadth-first, rather than depth-first traversal). It is possible that our datasets are currently too small for this effect to become noticeable, but it may be instructive to repeat this analysis on a larger dataset.
Figure 2.9 t-SNE embedding of Document, Task, User, and ‘User||Task’ nodes in the Multi-Layer Graph for Study B (the Political Science class). Entities for the three teams in the class are clearly separated; notice how ‘User||Task’ nodes are better distributed throughout the documents than the Task nodes, which explains the higher classification accuracy when these nodes are used as ‘Target’ nodes for DeepWalk.
Figure 2.10 t-SNE embeddings of nodes in the Multi-Layer Graph for Study B (the Political Science class). Document nodes are shown as blue crosses and Task (or concatenated-Task) nodes are shown as green diamonds. Notice how ‘User||Task’ nodes and ‘User||Task||Time’ nodes appear better distributed throughout the documents than the Task nodes alone, which explains the higher classification accuracy when these are used as ‘Target’ nodes for DeepWalk. The ‘Task||Time’ nodes produce a less noticeable benefit. ©ACM 2017.
Figure 2.11 Accuracy of baseline document to task association predictions, as the number of ‘guesses’ $n_p$ for each sample is increased from 1 to 5, averaged over 3 runs for each dataset. Users can set $n_p$ themselves in the dashboard interface to trade off the precision and recall of associations within each task group. ©ACM 2017.
Figure 2.12 Performance evaluation of the multi-layer graph classifier with and without random walk biasing through node2vec [GL16]. It can be seen that in each of the tested scenarios (Baseline, User||Task||Time nodes, and Doc-Doc edges), the effect of node2vec biasing is minimal. This is especially apparent for Study B where the lines for each of the three pairs of tests are approximately coincident.
2.6.5 Streaming-mode Evaluation Results

In order to evaluate the algorithm in the context of the ‘recent-work dashboard’ introduced in section 1.3.1, we make the following changes to the evaluation strategy described in section 2.6.3:

• Instead of removing a certain proportion of labels from the data uniformly at random and using our algorithm to guess them, in streaming-mode, we attempt to guess the task label for every new document, as soon as it is seen in the data. This is exactly as stated in the original problem formulation (section 2.1.1).

• We then assume that a user provides feedback to correct each predicted task label with a certain probability. This could be accomplished through the feedback buttons on the dashboard, such as shown in Figure 5.2. If an incorrect prediction remains uncorrected by the user, then an erroneous edge will exist in the multi-layer graph that could in principle reduce the accuracy of future predictions, leading to a degradation in algorithm performance.

• We also add an additional mode to the algorithm itself to incrementally produce embeddings for new nodes that appear in the data stream. Rather than commencing random walks from all task nodes at random (as in the Batch-mode evaluation above), we commence random walks only from new nodes that appear in the stream, and from their immediate neighbors. The hypothesis behind this is that, in a streaming context, we want to be able to quickly create embeddings for new nodes that are seen, without re-creating a full context dataset whenever a new node is encountered.

In this section, we evaluate our algorithm in this streaming context by measuring the cumulative accuracy on each of our three datasets as new documents arrive. We do this while varying the probability of user feedback, as well as measuring the impact of the focussed random walks, designed to embed each node as it is seen by the system.

Figure 2.13 shows the cumulative accuracy for each of the three datasets, with varying probability of user feedback for each document. Several observations can be made from these results. Firstly, in each plot, there are periods where the cumulative accuracy increases, and other periods where it decreases. Periods of increasing accuracy typically correspond to a stabilized number of currently active tasks (as can be seen from the Baseline, which shows the reciprocal of the number of active tasks) - the system is getting better at recognizing existing tasks and is not getting confused by new tasks. When new tasks are encountered, the overall accuracy often starts to decrease while high quality embeddings of the new tasks are created. Secondly, it can be seen that decreasing the probability of user feedback (to as low as 10%) does not cause a disproportionate decrease in accuracy with a widening gap - this is reassuring as it doesn't indicate a progressive degeneration of the algorithm even with a large number of potentially erroneous links in the multi-layer graph. Finally, a sudden
Table 2.7 Analysis of standard deviations in cumulative accuracy over 10 runs on each dataset with varying probability of user feedback.

<table>
<thead>
<tr>
<th>Feedback Percentage</th>
<th>Study A</th>
<th>Study B</th>
<th>Study C</th>
</tr>
</thead>
<tbody>
<tr>
<td>10% (Min/Max/SD)</td>
<td>0.27/0.42/0.05</td>
<td>0.33/0.45/0.04</td>
<td>0.21/0.44/0.07</td>
</tr>
<tr>
<td>90% (Min/Max/SD)</td>
<td>0.55/0.63/0.02</td>
<td>0.57/0.59/0.01</td>
<td>0.55/0.56/0.01</td>
</tr>
</tbody>
</table>

A drop-off in accuracy can be seen in Study C, after a long period of increasing accuracy - this is caused by a quirk in the dataset for Study C in that the final 1500 records all relate to one user, who continued using the Journaling prototype after the other users in order to test a new version of the dashboard. This user assigned the same task label to all the URLs he accessed - almost all of these URLs had been encountered before by other users and already had strong associations with other tasks, so the algorithm persistently made erroneous predictions. For this user, it may have been better not to take into account collaborator's task-document associations!

Note that, although, even with only 10% feedback, algorithm performance does not lead to increased degeneration, the cumulative accuracy does vary between runs much more than with 90% feedback, as can be seen from the results in Table 2.7. To create this table, the algorithm was run 10 times on each dataset - each time, feedback was provided randomly with the assigned probability, so different corrections were made each time. The minimum and maximum cumulative accuracies were recorded, along with the standard deviation. It can be seen that, as well as accuracy being higher for 90% feedback, the standard deviation between runs is also much lower. With only 10% feedback, the algorithm becomes particularly sensitive to initial conditions. For this reason, it is advisable for users to provide more feedback in the early stages of working on new tasks.

Finally, we examine the effect of commencing random walks only from new nodes in the graph, along with their neighbors (as opposed to walking the graph uniformly at random). This produced, on average, a small gain in classification accuracy (of between 2 and 5%) across all datasets. This demonstrates that focussing the random walks on those new nodes for which we want to create a high quality embedding is worthwhile, although the gains are minimal. Full results are not shown here for brevity.

2.6.6 Ensemble of Network Fusion and Sequence Prediction

As noted in section 2.5.6, it would seem sensible to combine the network fusion approach evaluated in this section with information from a sequence analysis model. These methods are orthogonal and rely on different information, which make them candidates for an ensemble. Furthermore, we have confidence information available for every prediction made by the network fusion algorithm so a simple way to combine the approaches is to fall back to sequence information whenever confidence
from the network fusion is low.

When ensembling in this manner with the Static sequence prediction algorithm (as described in section 2.5.3), our experiments showed only a marginal difference in classification accuracy from the ensemble, sometimes even a small reduction, compared to the baseline network fusion algorithm. These differences were usually within (or very close to) the bounds of the variances shown in Table 2.7 and, for this reason, results are not presented here. Our current hypothesis is that, when confidence from the network fusion algorithm is low, this is often indicative of a task switch, which would never be correctly predicted by the Static sequence algorithm. It would be possible to ensemble with the IPAM algorithm instead but the performance of this algorithm suffers badly with low probability of analyst feedback, as was shown in Figure 2.5, which makes this solution undesirable. Further experiments could be conducted in this space, although this is beyond the scope of the research questions presented in this thesis.

2.7 Discussion and Future Work

In this chapter, we have introduced a novel mathematical approach for modeling user workflows, and an extensible network-fusion based algorithm for associating users, documents and tasks for knowledge workers in a multi-tasking collaborative environment. The baseline algorithm achieves good classification performance (averaged micro-F1 > 0.6) on three hand-labelled datasets, when the proportion of labelled data used for training exceeds around 30%, and reasonable performance (averaged micro-F1 > 0.4) with as little as 10% labelled data. The accuracy can be increased by up to a further approximately 20% by the addition of ‘concatenated’ nodes into the graph (such as 'User||Task', 'Task||Time' and 'User||Task||Time' nodes), which tend to be better distributed throughout the documents in an embedded vector space than Task vectors alone. Any number of ranked predictions can be obtained from the algorithm so that users can trade precision and recall for particular tasks, typically achieving overall accuracy above 90% if between 3 and 5 ‘guesses’ are allowed. We have evaluated our algorithm in both batch and streaming modes of operation - the latter being the most common envisaged use-case.

We demonstrated a further optimization to the algorithm that involved biasing the random walks to encourage a breadth-first traversal of the graph, as was demonstrated in node2vec [GL16]. We did not observe significant gains with our datasets but, as noted by the authors, ‘continuous feature representations are the backbone of many deep learning algorithms, and it would be interesting to use node2vec representations as building blocks for end-to-end deep learning on graphs’. This suggests a bright future for constructing future learning algorithms based on this approach. Also, many different strategies for creating edges between documents in the graphs could be attempted, such as rapid text classification in the new fasttext [Jou16] algorithm.

It is worth noting that we have only demonstrated our approach from the perspective of asso-
ciating documents with user tasks but the techniques should, in principle, be equally applicable to other types of information artifacts (such as photos, videos, audio files, source code and so on). As we have seen, small performance gains might be expected if similarity metrics can be computed between related artifacts, and graph edges added accordingly. However, the gains are not likely to be as significant as those obtained by carefully constructing the graph using ‘concatenated’ nodes.

We have shown that the performance of our network-fusion algorithm typically exceeds that of sequence-based prediction approaches, such as IPAM [DH98]. However, sequence-based approaches are orthogonal to our network-fusion approach in that they rely on different types of information. As a result, we have shown that increased classification performance can be obtained through ensemble models.

As considered in section 2.4.2, it would also be possible to take into account document dwell times (i.e. the amount of time each user spends accessing each document), as well as the time gaps between document accesses. Both of these may have some predictive power, as demonstrated in [She07], but we have chosen not to incorporate these into our model thus far.

We have demonstrated our algorithm in the context of a ‘recent-work dashboard’ interface that automatically organizes a user’s documents according to the tasks to which they relate, and gathers user feedback on tentative associations. Further research is required to systematically measure the time saved from such an interface when used as an aid to information recall when switching tasks. Further research is also needed to determine realistic probabilities of gathering user feedback and corrections when deployed in real-world settings.

Finally, an interesting research direction would be the incorporation of ‘active learning’ [Set10] techniques into our model. Such techniques should allow us to pre-compute the likely impact on our models of each piece of feedback from dashboard users, and hence to choose which labels to specifically request from analysts, perhaps in order to remain within a specific ‘prompting budget’ that each user could specify.
Figure 2.13 Classification accuracy of network fusion algorithm in streaming mode on our three datasets, with User∥Task nodes enabled and varying probability of (correct) feedback from analysts. A simple baseline, which simply guesses uniformly at random from the currently active tasks is also shown.
3.1 Motivation

Recall from section 1.1 that we are primarily concerned with knowledge workers who are attempting to make sense of data in order to create new knowledge artifacts. This includes data analysts, researchers, and students. All of these types of workers are constantly having to deal with cognitive biases, an overload of often irrelevant information, and a difficulty in identifying the most pertinent information required for their domain. There are many modern systems that attempt to help with some aspects of these problems, but none that attempt to provide a holistic picture of the worker's activities on their computer.

Our goal for this part of the work is to develop a scalable instrumentation system to allow users to track their own workflows and provide a holistic picture of user interaction. Furthermore, we would like to do this, as far as is possible, in a passive manner - this means without disrupting or altering user's activity patterns in a way that would invalidate the potential assistance that can be provided to them. For this part of the work, we are aiming for near-zero user disruption and, whilst we need user consent for monitoring their activities, we almost want them to forget that there is any instrumentation running on their computers. Whilst not the focus of this chapter, a short overview of some of the ‘human factors’ of instrumentation (in particular, the trade-off between privacy loss and the benefits that may result from that) is included in section 5.9.

The types of data we may wish to passively capture in order to provide a near-complete picture of
Figure 3.1 Our ‘Instrumentation Hierarchy’ [Jon16a], showing the different types of information that can be captured from user desktop. We would like to use this data for task-centric document curation and, subsequently, for enabling other types of digital assistance. Ideally, we want to understand the user’s sensemaking processes through analysis of lower-level interactions with their computer. ©IEEE 2016.

human-computer interactions are illustrated in Figure 3.1. The pyramid is organized (from bottom to top) approximately based on the granularity of interaction information. At the base level are the lowest-level interactions with the computer, including key presses and mouse movements. Moving up, we can instrument various system, tool and application usage patterns. Combining these, we can observe analytic workflows and collaboration patterns. Finally, at the top of the pyramid is the process of user and group ‘Sensemaking’ (as defined in Table 1.1) that we are eventually trying to understand. However, this is also the most difficult to instrument in practice, so there is considerable value from studying the lower layers first.

Data from instrumentation of knowledge worker’s computers can be used to study all kinds of information analysis tasks. Some examples include analysis of competing hypotheses (ACH) [HP10], process/workflow [Don05], guided information search, and legal/policy compliance. In this work, we are not just considering the traditional roles of an information analyst; the definition of knowledge worker is extremely broad and our techniques apply equally well to almost anyone who works at a computer as part of their job, including most professionals and students. However, the focus of our research is on those people who are attempting to use computers to make sense of complex and often messy data.

However, our first priority for this work is to provide a feed of instrumentation data from user’s computers to allow the research challenges in Chapter 2 to be addressed. Before this was possible, preliminary research was needed into how such a data-capture system could be created. In particular, it needed to possess the following characteristics:

- **suitable for continuous use** - as outlined in Chapter 1, we are interested in enabling longitudinal studies of analyst workflow. In other words, we are not interested in enabling only short-
term experiments that require users to operate in a heavily adapted environment. Instead, we want to incorporate instrumentation onto knowledge worker’s desktops in a way that is minimally disruptive.

• **supports user activity prediction** - despite being minimally disruptive, we want to gather enough information about user’s activities that it is possible to develop algorithms to accurately predict what they might do (or need) next, as outlined in section 2.1.1. In other words, to make simple predictions about tool usage and analytic process patterns in Figure 3.1.

• **supports task-centric document curation** - furthermore, as a subsequent step towards creating useful capabilities for knowledge workers, we want to extract enough features to be able to train machine learning classifiers to accurately associate documents with user tasks. Our interfaces for manually gathering task labels from users are the subject of Chapter 5 of this dissertation but the instrumentation system needs to gather sufficient data to support automatic labelling.

• **supports analysis of collaboration patterns** - in addition to training algorithms on individual user’s data, we want to be able to understand multiple user’s workflows and to use all of this information to make better user-activity predictions and task-document associations for each user.

Finally, we would like to enable future research that gets closer to understanding user’s sensemaking processes (the top layer of our instrumentation hierarchy in Figure 3.1) by allowing the system to extract (or infer) higher-level artifacts, such as hypotheses, claims and evidence that users are working with during their analytic workflow. Some ideas for enabling this are presented in section 3.8, and one promising direction for this research is explored in chapter 4.

### 3.2 Related Work

Here, we describe other work in this area both in industry and academia. We elucidate the limitations of this other work prior to defining our contributions to the field in section 3.3.

Systems that gather data on human-machine interactions, including self-tracking systems, are common in many domains that involve analysis of user activities or execution of tasks and workflows. Applications built from this data include: **Enterprise Intelligence**, **Productivity Analysis**, and **creation of Smart Digital Assistants**. Our platform aims to support all of these, and to enable a new generation of smart assistant technologies capable of supporting the process of information analysis, or **Sensemaking**.

Many existing instrumentation systems capture a variety of low-level system ‘events’ such as keystrokes, running processes, button/menu usage and activities of user input devices. This is mostly
used for Enterprise Intelligence applications, including security (e.g. opensnoop to track running applications), and troubleshooting (e.g. dtrace [Can04] to instrument software in development and production systems). Some highly sophisticated products exist in the commercial domain that offer enterprise-level analytics on data collected from desktop and smartphone applications. Current high profile examples include Google Analytics [Cla14], New Relic, and Palantir. In the research domain, the Clotho project [Hai10] demonstrated that it is possible to use low level features and events (such as window activations and dwell times) to predict the utility of applications. This work provides a means to filter event data, so that only events that refer to ‘useful’ applications are forwarded for analysis.

Higher level information, such as a categorization of applications on the user's desktop, is used for Productivity Analysis - commercial examples include RescueTime and DeskTime, which attempt to guess how productive a user is being based on their application usage and web-browsing habits (using a dictionary of apps and sites classified by typical usage). Other user-centric instrumentation systems focus on capturing text and image content from the user's desktop, as opposed to the types of event metadata described above. For example, Gyllstrom [Gyl09] developed a tool called Passages that records all text-related activities on a user’s computer for subsequent analysis and retrieval of textual data. The tool extracts and indexes text worked on by a user, which is useful for tracing user actions and capturing provenance within documents. Waldner et al. describe WindowTrails, which is a graphical and visualization approach to browse history of user activity on a computer workstation [Wal14] primarily by capturing window and low-level events. The goal is to instrument user actions at window and application level to create interactive, navigable histories of workflows. Another example is DejaView, which is a system to record and replay entire sessions within a desktop environment [Laa07]. In addition, DejaView has the ability to extract text from screen shots of user sessions to enrich search and retrieval of past sessions.

Techniques to infer human activity from multi-modal input have also been developed. Cheng has proposed a framework to classify hierarchical and sequential activities of people [Che13]. The work is relevant because it instruments human activities using a range of different types of data such as motion, sound, and image data.

A significant ongoing trend in the computer industry is towards creation of ‘Smart Digital Assistants’ that attempt to provide timely and relevant information and recommendations to their users, and are typically voice activated. Examples include Siri, Google Now, Cortana and MindMeld. Whilst these tools can capture some high-level information about users, it usually has to be explicitly volunteered - typically these services react to user queries rather than using instrumentation for passive monitoring, which is needed for continuous anticipatory analytics. This constraint means that these tools currently lack broader contextual information about a user’s working patterns and goals, and so are somewhat limited in the scope of advice they can provide - they also cannot yet learn about a new domain automatically. A new open-source framework for smart digital assistants was released in
March 2015 - *Sirius* [Hau15]. This framework could complement the work described here by providing a platform for integrating many of the features in SDAs (particularly voice and image recognition) into enterprise applications hosting their own data centers, like the one described here.

With the benefit of pervasive desktop instrumentation that is capable of seeing all applications used and files accessed, new types of insight are possible. For instance, it becomes possible to do continuous *process mining* - extracting enterprise intelligence from event logs. A framework that provides a ‘pluggable’ environment for process mining is *ProM* [Don05]. Furthermore, once processes have been identified, it is helpful to be able to associate them with a task that the user is carrying out. This was demonstrated by a team at Oregon State University, who built a system called *TaskTracer* [Dra05], for capturing user goals in Windows XP. This is closer to our approach, but we require a system that can span multiple operating systems (including OSX, Unix, and Windows) and that will provide a state-of-the-art and scalable back-end to support future process mining, task recognition, recommendation and other analytics.

The landscape of tools and approaches for instrumenting computer workstations is vast, but there are limitations with current state-of-the-art technology that have motivated us to develop a new instrumentation platform. Specifically, each of the systems described above can be categorized as one or more of the following:

- those that capture only low-level or very specific interactions in Figure 1, which are not useful for assisting with workflow analysis or sensemaking,
- those that capture higher-level information but only when explicitly volunteered by users (i.e. they include no passive instrumentation so cannot continuously anticipate user needs),
- those that are more similar to our framework in terms of the data they collect but they lack an integrated, flexible and scalable big data analysis platform for both streaming and batch analytics, which is essential for providing timely and relevant assistance to users.

As such, we identified a need for a new instrumentation platform that could provide a more holistic picture of user activity, and that could enable the types of analytics described in Chapter 2.

### 3.3 Major Contributions

The key contributions of this part of our work include:

- Creation of a set of novel instrumentation agents for continuously and passively capturing tool usage patterns, analytic process patterns and user interaction patterns employed by knowledge workers without disrupting them, with a focus on the challenge of enabling task-centric document curation.
• Design of a state-of-the-art event-based architecture for ingesting instrumentation data, analyzing it using streaming and batch analytics, and producing actionable outputs. To do this, we employ an extensible data model for representing procedural data from instrumentation sensors. We believe the platform could be re-used in many other domains with minimal changes.

• Elucidation of several real-world use-cases for the platform and trials with students and intelligence analysts, which enabled the work in the other chapters of this dissertation, and facilitated several related projects.

These contributions are described in the following three sections.

3.4 Instrumentation Agents

Instrumentation agents are applications that gather data on analyst’s workstations and send the data to the instrumentation back end data store using the flexible event protocol format and instrumentation API described later. Agents differ according to the types of application and data that need to be instrumented. We have developed a number of instrumentation agents to support the research described in section 3.1.

3.4.1 Passive Instrumentation for macOS

Firstly, we developed a pervasive application for the macOS (formerly OSX) platform that would serve as our main continuous self-tracking agent for user actions. To ensure the agent was capable of extracting the necessary features for our principle challenge of task-centric document curation, we first had to determine a minimum set of required features. From previous research, it was determined that a minimum set of necessary (but not necessarily sufficient) features required were: use of information artifacts on the computer desktop, such as documents and URLs [Cow05][Gyl08], application usage events [Des14][Dra05][Res14], and the ability to distinguish active usage of an application from those that may reside in the background [Hai10]. Previous research projects determined that all of these features could provide strong clues as to task associations. In addition to these features, the macOSInstrumenter agent can be configured, according to user consent, to log other types of general user actions such as mouse and keyboard events, screenshots and access to file or network resources (to provide comprehensive coverage of the data types in the bottom half of our instrumentation hierarchy in Figure 3.1).

The macOSInstrumenter application is based on the Apple Cocoa framework, and built in Xcode. It uses the NSWorkspace API to subscribe to notifications of application-level events (activated, deactivated, launched and terminated), and other system APIs to capture screenshots, and mouse/keyboard events. In addition, it uses AppleScript APIs to retrieve the currently viewed URL in Safari or Chrome.
The agent generates a continuous stream of user activity sequence data, which, in addition to task-centric document curation, can be used for research into techniques for predicting the next action that a user may take, as per the use-case in section 3.6.1. It also supports the extraction of user search terms, which supported the work described in section 3.6.3. Finally, the agent is capable of capturing full-desktop screenshots, which enabled the work in chapter 4.

While this agent provides a relatively holistic picture of each user’s activity on their computer (at least in terms of the applications that they use), it is unfortunately tied to one particular desktop operating system.

3.4.2 Passive Instrumentation for Chrome web browser

We also developed an instrumentation capability for the Google Chrome web browser, which we called ChromeInstrumenter. This provides a platform-independent instrumentation capability that can be used to gather detailed information on applications running through a web browser. Whilst this is sufficient for logging usage of web-based documents, it does not provide a fully holistic picture of user’s activity on their computers, nor does it aim to extract all the necessary features for task-centric document curation, except in limited circumstances where users are working entirely in their web browsers.

This agent was built as a browser plug-in using Google’s Client Library Javascript API, and also formed the basis of our active journaling interfaces described in Chapter 5.

3.4.3 Other Instrumentation Agents

3.4.3.1 GitHub and Bash Shell Instrumenters

During development of the above agents, we used GitHub as a source code repository and collaboration tool for software development. In particular, we extensively used GitHub’s issue tracking user interface to track and discuss issues and errors during software development. To study our software development workflows, we created an instrumenter to obtain data from our GitHub account. The GitHubInstrumenter is a Python script that uses GitHub’s public API to obtain information on events such as commit actions, opening/closing of issues, and comments on issues. The instrumenter obtains the GitHub data and converts it into event protocol format, before saving it to instrumentation system back-end database.

Throughout this work, we explored a number of potential uses for our GitHub data. Firstly, we prototyped a web-based context-sensitive recommendation application to display relevant links from a StackOverflow database whenever a user clicked on an item in GitHub’s issue list. The recommendation application worked by extracting keywords from comments in the issue and matching content with a local copy of StackOverflow data. The recommendation application ran as a plug-in for the Chrome browser and served links in a pop-up box next to the clicked issue. Secondly, we are
exploring a possible use case regarding understanding collaborative development practices among software-engineering personnel. To support this, we also developed an instrumentation agent for the Bash shell that records activities of developers in a Unix terminal. The agent is itself a Bash shell script that runs in the background in a Unix environment recording commands executed on the shell prompt, and formatting them according to our event protocol. Our plan is to use the Bash command logs as a supplement to our GitHub data for analyzing collaborative software development activities. This work is ongoing but is beyond the scope of this dissertation.

3.4.3.2 Instrumenting other applications

We investigated building additional instrumentation agents to monitor user activities in a number of other applications - for example, Microsoft Excel and Gmail web pages. In Excel, it might be useful to capture user interactions with menus and within individual cells, thereby providing potentially rich information on analytical tasks. In Gmail, it might be useful to log collaboration contacts and to try to associate e-mails with user tasks.

However, we were also mindful of two things - firstly, we were conscious of privacy concerns with gathering detailed information from inside applications (see section 5.9); secondly, we were keen to learn lessons from other instrumentation projects (such as TaskTracer [Dra05]) that developed complex software 'hooks' into individual applications. For these two reasons, we wanted to avoid going down the road of instrumenting a large number of specific applications. Instead, we stuck to a small number of high-level agents, whilst also exploring the possibility of instrumentation via screenshots (the subject of chapter 4).

3.5 Event-based Architecture

3.5.1 Overview

A central aim of our instrumentation framework is to provide a rich platform for analytics on large data sets, starting with user activity prediction and task-centric document curation, but potentially many others in the future. As such, we wanted to provide tools for open ended exploration of data sets using visualization tools, and for hosting analytics, which may be computationally intensive (like map reduce, graph analytics, machine learning and streaming analytics).

Another goal for our instrumentation framework is to support investigation of both individual and group interactions that are inherent to analytical processes in human-machine systems. As such, we wanted to build a platform capable of aggregating data in a standard way from many different individual's computers.

Figure 3.2 shows a data-centric view of our event-based instrumentation architecture and its main components. Instrumentation agents (such as those in section 3.4) run on user desktops, con-
Connecting to local applications as necessary and sending event logs through our data API to a data collection/aggregation service, and on to an analytic cluster. Analytic results are then sent back to the user desktop through our web-based visualization portal.

In the remainder of this section we describe all the main aspects of our instrumentation framework, starting with our instrumentation agents. We then briefly describe the other software and hardware components of our platform.

### 3.5.2 Integration with Microblogging Platform

In addition to capturing activity logs from users in a passive manner using instrumentation, we also capture textual and multi-media data from a corporate micro-blogging service and social media platform ‘Yammer.com’. Participants involved in some of our instrumentation experiments (described in Use Cases below) were collaborating with colleagues using the Yammer platform throughout the data
capture phases - some communications were directly relevant to their current tasks, others not. This additional source of unstructured data from Yammer allowed us to help explain what we were seeing in the passive Instrumentation data. Yammer’s well-documented REST API allowed us to easily extract all the data from our groups, and send it to our instrumentation platform.

Separately, Yammer provided a convenient aggregation point for other sources of instrumentation from the web. Specifically, participants were able to link up their Google Calendars, and their Locations (via Foursquare/Swarm) to the Yammer platform using the internet ‘plumbing’ service IFTTT. Every time a new meeting was added to an analyst’s calendar, or whenever they checked in their location on Foursquare, we were able to use an IFTTT recipe to create a new Yammer post, which was then retrieved and stored in our system using the Yammer REST API. This additional metadata of meeting events and locations then formed additional context for our experiments.

3.5.3 Instrumentation API

An application programming interface (API) was developed to enable the instrumentation framework to receive data from external sources (typically direct from client computers) and allow researchers to access stored data for their analyses. The API is based on the standard REST web architecture, and provides HTTP web services that facilitate data transfer between client computers (instrumented workstations) and servers, such as our data store. In addition, a Websockets API is used to create persistent communication channels that allow the server or clients to exchange information asynchronously. The API provides data channels to receive different types of data - primarily it supports event data (in the form of JSON objects) but we did not want to preclude the possibility of receiving text content, image files and binary data files.

3.5.4 Data Model

A new data model, which we call event protocol, was developed to standardize the format of event messages handled by our instrumentation framework. The protocol is essentially a set of well-defined key-value pairs. The protocol contains two sections: a core section and an application specific section. The core section contains required fields that pertain to the identity of the user, date and time stamps, and details of events. Table 3.1 shows the fields in the core section. The constraints on the requirement of the fields were developed to capture the least amount of information that is necessary to allow researchers exploring the instrumented data to perform meaningful and useful analyses. The application specific section allows client applications to append user-defined data fields that are specific to individual applications. Some examples include URLs for web data, 3D coordinates for applications involving virtual worlds, and text from chat applications.

In addition to event data, the instrumentation framework can also receive and store binary data files, images, and other standard types of internet data. We describe a use case in Section 3.6.3 that
discusses an application of our framework to collect event and image data during a scientific study into online search behaviors.

Our data model is similar to Common Event Expression (CEE) format developed to support interoperability in electronic data exchange [The08; HM08]. In an analogous way to the core and application sections in our model, CEE specifies a core profile, which contains mandatory fields and optional fields. The CEE specification goes further, specifying a dictionary to standardize field names or labels and an event taxonomy to classify event types. CEE also specifies guidelines for event log transport and versioning. While our data model does not strictly conform to CEE, it can be thought of as loosely modeled after CEE but with fewer restrictions. For example, CEE requires certain ordering of fields, cryptic labels for field names, and places limitations on the number of nested fields (up to two are allowed) in hierarchical data. Those requirements are overly restrictive for our use case and offer less flexibility. Moreover, CEE was developed by industry vendors and government partners primarily to standardize data exchange among devices and applications (note: development of CEE was abandoned in 2012), whereas our focus is to gather data into a central hub for analysis, for which the simplified event protocol format suffices. However, because we store data in MongoDB database, which can accept arbitrary text documents, we can theoretically ingest data in other formats as well, including CEE.

### 3.5.5 Query and Visualization Portal

Data gathered from instrumentation agents can be accessed through a central web site or using the instrumentation API discussed in Section 3.5.3. Components of the site include a query page to view and download data, a visualization sandbox to host experimental visualizations of the data, and a personnel dashboard that contains plots and charts showing an individual’s data. Users in the system who run instrumentation agents can go to the dashboard to see their data and basic statistics. Figure 3.3 shows plots of a user's GitHub data - the dashboard content provides actionable insights, thus providing an incentive for users to agree to instrument their workstations. More detail on these and other visualizations of personal instrumentation data are described in our recent paper [Tha15].
Figure 3.3 Collection of some of the plots in personnel or enterprise dashboards that shows one or more user's activity patterns (in this case derived from the 'Github' social coding platform). These kind of dashboards provide managers with actionable enterprise intelligence (for example, by showing who is working on which projects, and by raising prominent or neglected issues). Reproduced from [Jon16a], ©IEEE 2016.

Figure 3.4 Architecture of the Data Collection/Aggregation service in our framework [Jon16a], showing the key software components used. It is designed for throughput, scalability and reliability. ©IEEE 2016.

3.5.6 Data Collection and Aggregation Service

At the core, our instrumentation framework is a web-based client-server system that stores event data in a key-value store and uses peripheral software components to handle event messages. Figure 3.4 shows a schematic diagram of the main components. The framework is built using open-source software packages and can handle large volumes of data while supporting high concurrency.

The back-end data store we currently use is MongoDB, but other similar NoSQL data stores (such as Redis) could be substituted. The application servers are instances of a Node.js web server, the messaging queue is handled by kafka, and the load balancer is nginx.

3.5.7 Hardware Infrastructure

Our instrumentation framework is currently deployed in a development environment that consists of a computational cluster with 14 nodes and running CentOS 6 operating system. Data and application storage solutions are provided through IBM Flex and NetApp FAS3170 systems. OpenStack is used for cloud-based services. The software components are deployed over a fault tolerant data
center compute scheduler called Mesos. We are also instrumenting the hardware layer of our system using collectd and other tools [Kau15]; this data is processed in the same way as user-derived data.

3.6 Passive Instrumentation Use Cases

Whilst the focus of this dissertation is on research relating to the use of instrumentation and journaling for task-centric document curation, the instrumentation platform was originally evaluated using two simpler use-cases that did not rely on any journaling components.

3.6.1 Prediction of User Activity

We initially collected approximately six months of data from macOS computers used by a group of analysts at the Laboratory for Analytic Sciences. The group started out at 10 analysts and grew to around 20 at the end. Using this question, we set out to answer the question ‘Can we predict what an analyst will need next?’ in the context of applications and documents on their desktop, and information searches on the web. The motivating use-case was that, given reasonably accurate predictions, it should be possible to both allocate resources in advance (e.g. for an expensive database query), and to recommend information artifacts (e.g. documents and URLs) before they are accessed.

Our platform allowed us to easily test out a number of different analytic approaches, both for temporal sequence prediction and for providing recommendations. We initially found that an online Hidden Markov Model (HMM) [Rab89] approach worked well on the macOSInstrumenter and ChromeInstrumenter sequence data, giving around 75% accuracy for some users when predicting the next 3 applications or web domains they might access. However, we were able to improve on this using a combination of a frequency-based analysis, and a vector representation of user actions [Moo16]. This approach is explored further in the context of task sequence prediction in section 2.2.3.

For providing URL recommendations, we found that a Collaborative Filtering approach using the well-known Term-Frequency/Inverse-Document-Frequency (TF/IDF) technique [Wan06] for finding related documents worked well for recommending web sites. We delivered these recommendations directly to the user’s browser in real-time through the same Chrome plugin as ChromeInstrumenter. Based on this initial work, several strands of active research into the effectiveness of recommender algorithms is ongoing in our group, although this work is not the focus of this dissertation.

3.6.2 Instrumentation for Enterprise Intelligence

As described in section 3.5.2, we also collected micro-blogging data from the same group of around 20 knowledge workers. We allowed them to submit completely freeform text into the Yammer platform, except for one convention - all ‘yams’ were prefixed with one of three labels: #r to indicate that the post was Reflecting on a past event, #o to denote posts that were Observing an ongoing activity,
and #i to indicate that the user was *imagining* or looking ahead to something in the future. From this data, we were able to create on-demand summaries of each user’s activity - for instance, we could create monthly notes, both individual and aggregate, summarizing user's achievements last month and plans for the next. The Yammer platform also allowed us to approximately answer the question of ‘Who collaborates with whom?’ based on user tagging of posts, and see this evolving over time.

The potential of this data for enterprise intelligence only really became apparent though when combined with our instrumentation data. The combination of microblogging with instrumentation allowed us to:

- add much richer detail to on-demand summaries (e.g. monthly notes), based on activity patterns displayed by the user, such as their favorite applications, software development patterns (from GitHub) and web site usage.
- attempt to automatically label certain application ‘cycles’ with an activity that the user was posting about in Yammer. The potential for using this data for machine learning is being investigated.
- attempt to automatically answer the question ‘What is each person's job role in the organization?’. We set a group of students a challenge to attempt to classify each of the 20 knowledge workers into leaders/managers, analysts, technical staff or teleworkers. Using just Yammer message metadata (with no message content), and macOSInstrumenter logs, they correctly categorized 90% of the staff.

This early work merely hints at what might be possible in the future and we are working on techniques to measure the benefits of such capabilities systematically. For now, they serve only to demonstrate the potential of instrumentation to deliver organizational insights.

### 3.6.3 Analysis of Online Search Behaviors

We supported an experiment conducted by our university collaborators at the NCSU College of Humanities and Social Sciences (CHASS) to study how online information foraging behaviors of computer users may be biased by participant’s personal interests and belief structures. We used our instrumentation framework and *macOSInstrumenter*, discussed in Section 3.4, to capture data from participant’s computers during the experiment. The instrumentation captured information such as web URLs and search terms entered into browser text boxes. The data was converted into event protocol format and sent to our platform via event data channels provided by the instrumentation API. In addition, screenshots of computer monitors were captured at periodic interval (every 10 seconds, and on mouse-clicks) and uploaded to instrumentation server using image data channel of the API. We wanted to know roughly where on the screen the user was looking - for this experiment we simply
Table 3.2 Events captured by macOSInstrumenter during a single run of the experiment, based on the activity of three people over a 90-minute period.

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Total Count</th>
<th>Unique events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search terms entered</td>
<td>51</td>
<td>48</td>
</tr>
<tr>
<td>URL requests</td>
<td>349</td>
<td>177</td>
</tr>
<tr>
<td>Screenshots captured</td>
<td>2588</td>
<td>2588</td>
</tr>
<tr>
<td>Application transitions</td>
<td>1043</td>
<td>24</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>4031</strong></td>
<td><strong>2836</strong></td>
</tr>
</tbody>
</table>

used the location of the mouse pointer as a proxy for this but, in the future, we would like to use an eye or face-tracking technique, perhaps like that in presented in [Rä09].

A query page on the instrumentation query and visualization portal web site provided options to download the collected event data as comma separated value (CSV) files or view the data in a web page. Clickable links to web URLs visited by the participants and screen capture images were inserted to allow experimenters to the information to better tag or code user activities. Each session in the experiment ran for about 90 minutes, which generated a few thousands events (3-5K). The statistics from one particular run are shown in Table 3.2.

The experimental data capture completed in November 2014. In total, 60 experiments were run, each with 3 participants. These 180 datasets are currently being analyzed in order to answer a number of research questions of importance to social sciences - in particular:

- What online search strategies are more likely to lead to high quality analysis reports?
- What is the relationship between online search behavior and confidence in decision choice?
- What is the relationship between the number of sites visited and the number of sentences participants produced in their written reports?
- Does distribution of dwell time vary by type of website?
- How does a group discussion influence user's perceptions of the quality of their information search, both when the discussion was focused on the data itself, or the process of obtaining the data?
- What individual personality factors influence a user's ability to search for information on the web, especially any particular cognitive biases?

Our desktop-instrumentation based approach represents a new paradigm for experimental methods in the humanities, which is allowing researchers to address these questions and others. Results from the analysis of our data have been submitted for publication.
Furthermore, we collected 121,000 screenshots (over 400GB) from this experiment, which enabled research into a new approach for instrumenting aspects of sensemaking, as described in chapter 4.

3.7 Performance Evaluation

This section presents a brief evaluation of some of the runtime characteristics of the agents and event-based architecture described above. The evaluation shown here is limited in scope to verifying that the first key characteristic listed in section 3.1 (that is being suitable for continuous use) has been met.

3.7.1 Performance Characteristics of Instrumentation Agents

We measured the runtime characteristics of both the macOSInstrumenter and ChromeInstrumenter agents. Over each period of running for one hour, macOSInstrumenter was found to consume 20 seconds of CPU time. This equates to an average of 0.56% CPU utilization. It also used a stable 21MB of memory. These numbers are with application, URL and file-based logging enabled, but not screenshots. The agent was also found to be extremely stable over time, with no crashes observed during 2 years of continuous running on multiple desktops. Similarly, the ChromeInstrumenter did not noticeably slow down Chrome web browsers, although the exact level of CPU and memory utilization is difficult to measure independently. This agent also showed similar levels of stability with no crashes observed during continuous operation.

We did not characterize the runtime performance of the Github and Bash shell instrumenters although CPU and memory utilization for both is expected to be negligible.

3.7.2 Performance Characteristics of Event-based Architecture

To provide a flavor of the performance characteristics of the event-based instrumentation platform, we firstly measured typical ingest rates from our experiments and the corresponding storage requirements. Next, in order to demonstrate how far the system would likely scale, we used the Apache Benchmark framework to stress-test the platform, whilst running the Mongo system profiler tool and MongoStat to capture the impact of these tests on the back-end database.

3.7.2.1 Typical metrics from current our use of the platform

We are currently using our platform for about 15 active experiments (a few of which are described in Section 3.6). On average, we are ingesting approximately 50000 events per day, and responding to approximately 2000 queries per day. In total we are holding about 500 million events (occupying 400GB of disk space), which represents around 2% of our system capacity.
Table 3.3 Benchmarking characterization for the two most common operations supported by our platform. Each request contained 350 bytes, was repeated 10000 times, and the average taken from 3 separate runs. Minimal effort was made to optimize the performance of MongoDB - these are out-of-the-box results.

<table>
<thead>
<tr>
<th>Request Type</th>
<th>Concurrency Level</th>
<th>Requests Per Second</th>
<th>Time Per Request (ms)</th>
<th>SSL Handshake Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert</td>
<td>1</td>
<td>12.6</td>
<td>79.1</td>
<td>0</td>
</tr>
<tr>
<td>Insert</td>
<td>10</td>
<td>96.4</td>
<td>10.4</td>
<td>0</td>
</tr>
<tr>
<td>Insert</td>
<td>100</td>
<td>180.4</td>
<td>5.5</td>
<td>0</td>
</tr>
<tr>
<td>Insert</td>
<td>500</td>
<td>180.9</td>
<td>5.5</td>
<td>36</td>
</tr>
<tr>
<td>Query</td>
<td>100</td>
<td>2.0</td>
<td>510</td>
<td>0</td>
</tr>
<tr>
<td>Query</td>
<td>500</td>
<td>10.0</td>
<td>100</td>
<td>15</td>
</tr>
</tbody>
</table>

3.7.2.2 Benchmarking common operations

We sent both ‘add’ and ‘query’ requests to the Data Collection and Aggregation service using the Apache Benchmark tool, while enabling various levels of concurrency. A typical event protocol record from these tests is shown below:

\{
  "content": {
    "UserId": "event_simulator", "AppName":"AB_Testing",
    "SysId": "Fedora20", "ProjId": "sim:instr", "NetAddr": "none", "EvtTime": "1426186096",
    "EvtType": "ab_testing", "EvtDesc": "ABTestingEvent"
  }
\}

The performance results are shown in Table 3.3. Note that no optimization was carried out prior to generating these results - it is possible that we could do much better by carefully optimizing system configuration and parameters.

These results show that system performance is dramatically improved when dealing with concurrent requests, up to a few hundred simultaneous connections, beyond which we started to see SSL Handshake Errors. For a concurrency level of 100, the average data insertion time was measured to be around 6ms, and query time around 500ms. These numbers could probably be significantly improved with careful setup of the platform, but they have so far amply supported all of our use cases.

3.8 Discussion and Research Directions

So far, our instrumentation platform has demonstrated the ability to log, store, query and visualize user actions from a variety of desktop and application contexts. We have also begun to demonstrate a variety of application use-cases enabled by the platform. Building on this foundation, we developed the journaling interfaces described in Chapter 5, which allowed us to gather labelled data for ma-
chine learning, and subsequently to build machine learning classifiers for automatically associating documents with tasks, as described in Chapter 2.

While we are confident that we have met our core research challenge of providing the core features necessary for task-centric document curation, we also want to explore other possible aspects of individual workflows and collaborative intelligence that might be amenable to increased understanding through the use of passive instrumentation agents. As such, there are a number of additional research directions that have been opened up as a result of this and similar research:

• **Finer-grained Usage Data:** We would like to develop the ability to log additional contextual information characterizing not only the user applications being executed from an operating system perspective, but from user and task-centric perspective. In other words, it would be valuable to collect broader contextual information such as the specific tools or features being used within an application and data that are being processed. The context will provide greater visibility of user actions and intent, and thus allow us to build more sophisticated models.

• **Generation of Summary Reports:** We intend to use this platform to automatically generate a variety of summary reports, describing various aspects of individual and enterprise working patterns. These may include summaries of how users have spent their time, whom they have been working with, and what they have achieved. These artifacts can be used by workers and managers to keep track of progress and identify opportunities for improvement.

• **Making Timely Recommendations:** A pattern we expect to see repeated in instrumentation is the need to analyze a live, continuous stream of events data in order to deliver real-time context, feedback and recommendations to the user. This streaming analysis must be done in the context of previous events, and of static reference data, in order to learn new workflow patterns, spot anomalies and changes in behavior, and to be able to recommend applications or information sources that the user may find useful.

• **Capturing the Outcome of Workflows:** Research is needed into ways to capture the outcome of each user’s workflow, so that the system can know whether or not each task they were attempting was actually met. This could be done actively (i.e. by asking the user) or perhaps it could be inferred passively. If the outcome of a task was successful, we can begin to learn ‘best practice’ and improve the recommendations made to other users. If the task was carried out unsuccessfully, our platform may be able to guide a user towards a better outcome the next time around.

• **Analytics as a Service:** We are planning to provide a catalogue of analytic elements that will allow our users to more easily build and run sophisticated analytics. Visualizations of work processes and performance will be accessible for teams to better understand their data.
• **Resource Optimization**: We are attempting to model the system’s performance at a technical level to optimize available resources. For example, we are planning to develop a predictive model that allows us to allocate resources to a computational problem in advance of the need materializing, perhaps using algorithms such as those described in section 3.6.1.

• **Instrumenting Sensemaking**: Finally, we are working to strengthen our current efforts to understand sensemaking by instrumenting new tools being developed to guide analysts through the process of information analysis, in terms of reasoning over hypotheses and finding supporting or contradictory evidence.

Regarding the last bullet above, we are designing experiments to test whether the types of instrumentation and methods described here can help us understand the process of sensemaking itself, as defined in Table 1.1. In his book [Moo12], David T Moore describes the need for a new analytic framework for sensemaking, particularly in the realm of Intelligence Analysis. This is needed to help analysts become more effective at understanding genuinely hard, open-ended, messy problems, with no clear solution; otherwise known as *wicked* problems. Based on earlier work at Ohio State University [Zel07], he describes eight aspects of the sensemaking process than can be used to assess the ‘rigor’ of a particular piece of analysis. These are: Information Search, Information Validation, Stance Analysis, Sensitivity Analysis, Specialist Collaboration, Information Synthesis, Explanation Critiquing and Hypothesis Exploration. The author advises that each analyst assess their own work against each of these eight metrics, using rankings of High, Medium or Low.

We are setting out to attempt to measure some of these through a new prototype being developed at our lab that allows analysts to ask questions about ‘wicked’ problems (e.g. *To what extent will climate change impact the world in the next 20 years?*). The system searches news feeds, and several other web-based sources for documents relevant to the question and ranks them according to carefully selected criteria. We then use our instrumentation platform to capture events based on how the analyst is processing the results, and we use these to estimate ‘High, Medium or Low’ rigor on four of the eight metrics defined in [Zel07]. The events we plan to capture are shown in Table 3.4.

We then build heuristics from each of the metrics in Table 3.4 to rank each metrics as High, Medium, or Low for a particular use of the tool. This ranking is then provided to the analyst for verification. We use the results from this verification to improve our heuristics. This capability serves to remind analysts when the rigor of their analysis may be lacking in a particular way; in the future we hope to be able to recommend ways to improve the process, or to build analytics that go some way towards automating some of these processes (such as the information search). This is a very early effort to instrument the process of sensemaking, and results are too preliminary for publication, but we are using our experience of instrumenting diverse data types and in a range of different tools, to design experiments to get closer to an understanding of this emerging field.

Separately, there is a new ongoing experiment underway that is building on the work described
Table 3.4 Aspects of Sensemaking we could attempt to measure using our platform

<table>
<thead>
<tr>
<th>Rigor Metric</th>
<th>Event types we could capture, and derived metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Search</td>
<td>Number of questions asked; proportion of returned documents examined</td>
</tr>
<tr>
<td>Information Validation</td>
<td>Number of documents examined from different sources that have high relevance to the question</td>
</tr>
<tr>
<td>Sensitivity Analysis</td>
<td>Likely veracity of each document or source; number of high veracity documents used in the analysis; number of times the question was rephrased and consistent results were obtained</td>
</tr>
<tr>
<td>Specialist Collaboration</td>
<td>Domain expertise/authority of each document; any attempt to ask advice from a specialist (e.g. via Yammer platform)</td>
</tr>
</tbody>
</table>

In section 3.6.3, this time around, participants are being asked to think about the resource requirements for a Mission to Mars and to use the web to put together a brief report. In addition to desktop-based instrumentation (the macOSInstrumenter), participants physical gestures while discussing this problem with each other, and while working at the desktops, will be measured using Xbox Kinect equipment. A future publication will explore the interplay between these two completely orthogonal, but hopefully complementary, forms of instrumentation.

Note that we are currently not instrumenting mobile platforms such as smart phones and tablet PCs due to lack of an appropriate use case, although we expect this to change. We can adapt our platform and develop instrumentation agents to support mobile platforms when they are needed. We expect almost all of our existing agents will port to mobile platforms with minimal additional effort.

Clearly, there is much more work to be done to expand the instrumentation agents and platform described here, particularly with regards to capturing the reasoning processes of knowledge workers while analyzing information. This work is ongoing at NCSU and the LAS.
4.1 Motivation

As we have seen, the need for both computers and humans to develop a better understanding of the workflows and sensemaking [Moo12] processes used by knowledge workers is becoming increasingly urgent. The volume and variety of available data is continuing to outpace automated methods that help people to analyze it. As a result, knowledge workers are increasingly falling victim to their own cognitive biases during knowledge creation tasks. A well-known example is confirmation bias, which results in people looking only for evidence that supports their existing viewpoints. This has been observed in fields as diverse as intelligence analysis [Joh05] and emergency medicine [Pin06]. Clearly, in both of these fields, such biases can have very serious consequences.

As a result of this need, much research has been carried out into approaches to understand the working patterns of knowledge workers and to automatically generate provenance information [BD06] for work processes and written reports. Traditional approaches have relied on either asking workers to self-report on their activities via surveys or interviews, or in having an observer literally ‘watch over their shoulders’ for a few hours and make notes. Both of these methods can only provide limited-time snapshots of activity and are also highly subjective.

As covered in Chapter 3, more recently, researchers have been developing ways to continuously measure workflows via instrumentation [Jon16a][Dra05][Cow05] technologies that run on computer desktops and record various aspects of user workflows. All of these technologies have relied on soft-
ware 'hooks' into the computer operating systems (OSs) and into individual applications that allow the instrumentation to 'see' what the computer is doing. Unfortunately, as the instrumentation agents get more sophisticated, they have to look deeper into individual applications, resulting in the need to build increasingly complex and brittle 'hooks' into a wide array of different applications (some of which might expose a supported API for doing this, but most do not). Previous projects, such as *TaskTracer* [Dra05], have fallen foul of exactly this - the team built a large code base in Windows XP and are having difficulty\(^1\) maintaining it, let alone upgrading it to support newer versions of the Windows operating system.

Furthermore, as we found from our initial work on knowledge worker instrumentation [Jon16a], it is not enough to just record 'button presses' or application usage events. In order to build systems that can help workers overcome confirmation bias, we need to be extracting *sensemaking artifacts*, such as hypotheses, claims and evidence (i.e. the actual pieces of relevant text that knowledge workers are reading or writing on their screen). To address this need, and to learn from past experiences of software instrumentation that became complex and brittle, we propose an entirely different approach - that of instrumenting knowledge workers by having the computer look at the exact same screen that they are looking at. This provides a couple of key advantages - reduced sensitivity to operating system and application upgrades and, most importantly, an ability for the computer to more easily 'see' exactly how the user is working with applications and assessing hypotheses, claims and evidence, at least where these exist in written form on the screen. The downside, of course, is that image processing techniques and optical character recognition (OCR) are notoriously expensive in terms of processing resources. However, this is becoming less of a problem with the introduction of increasingly powerful GPUs into desktop computers. Typically, during knowledge-working tasks that are not graphics-intensive, the GPU is essentially idle, so this resource can be repurposed.

### 4.2 Related Work

This work brings together two fields of research that were previously distinct: instrumentation of knowledge worker's computers, and semantic analysis of desktop screenshots.

Previous instrumentation efforts have varied in terms of the types of data that they attempt to collect, and in terms of how they collect it. Some systems, such as *Clotho* [Hai10], were concerned with predicting the utility of applications using low-level features (such as keyboard activity and mouse movements). Others considered higher-level workflows, such as application activations and web page accesses - a commercial example being the well known productivity tool *RescueTime* [Res14]. The *TaskTracer* [Dra05][She06] research effort extended this by also considering document accesses and application-specific actions. Our instrumentation platform [Jon16a] built on the work of these others by considering collaborative workflows and through the collection of event types that allowed

\(^1\)http://tasktracer.osuosl.org/?p=697
certain sensemaking rigor metrics to be captured (such as information validation and specialist collaboration). These systems are essentially passive in that they don't require input from users; other platforms such as Glassbox [Cow05] required analysts to manually annotate aspects of their work. All of these systems suffer from the limitation that the source code is necessarily operating-system (and sometimes application) specific, which makes it brittle and difficult to maintain. Furthermore, these systems struggle to extract high-level information relevant to the sensemaking process (which is typically difficult to obtain through standard APIs). Furthermore, they fail to place different applications into proper context when multiple applications are being used at once. Often the layout of windows on an analyst's screen says a lot about how they are gleaning information from them.

Previous work on semantic analysis of screenshots has been targeted at applications including spam detection in e-mail programs [Ara05], and on building contextual help into graphical user-interfaces [Yeh11][Yeh09]. In both of these cases, the researchers leveraged screenshot analysis approaches to create capabilities that were relatively independent of specific operating systems and applications. There are also some commercial products starting to appear, such as Polarity2 that use screenshots to extract entities from arbitrary applications, and use this information to remind users where they last saw information regarding particular people, places etc. In terms of screenshot analysis algorithms, there has also been some preliminary work on using Convolutional Neural Nets [Sam15] for simple task classification from screenshots. However, to our knowledge, our work is the first attempt to develop screenshot analysis techniques to correlate two key sensemaking artifacts (claims and evidence) through analysis of text in multiple windows.

The problem of matching related claims and evidence is also related to the fields of intrinsic (content-based) plagiarism detection, and argumentation structure analysis. Plagiarism detection is relevant in that methods based on 'local-similarity' often use 'fingerprints' based on n-gram analysis of sentences, as well as 'term occurrence analysis' using methods such as substring analysis and 'bag of words'. We employ some of these methods in our approach. Other extrinsic (or global) features of documents used in plagiarism detection are not relevant here. We also do not attempt to deconstruct the structure of claims and evidence in order to try to understand the structure of arguments being made, although this would be a natural next step.

4.3 Major Contributions

Our goal in this chapter is to use screenshot-based instrumentation to test two scientific hypotheses - firstly, that analysis of screenshots can be used in place of OS/API-based instrumentation technologies to record workflows (such as application and document/URL usage) and secondly, that screenshot-based analysis can potentially be used to identify both the claims that workers are making in written reports, and the evidence that they are using to justify these claims. For the second

---

2http://breachintel.com
hypothesis, we are making the assumption that knowledge workers are analyzing *new information under time pressure*. In other words, they are not making claims based on prior knowledge of a particular topic, and they have minimal time to contemplate and refactor new knowledge that they learn. Hence we assume that *claims and associated evidence will be found close together in time and in space on the computer screen*. This will not be the case for all knowledge workers and for all situations - particularly in the case of experts, who have built up their expertise over a long time period. However, we suspect that, even expert knowledge workers sometimes find themselves in situations where they are forced to make judgements on new information in a short period of time. This research is particularly aimed at enabling assistive technologies to be created for this situation by providing a continuous feed of relevant input data.

Key contributions of this work include:

- a demonstration that it is possible to accurately reproduce aspects of ‘traditional’ desktop instrumentation through analysis of screenshots instead of through low-level OS/API software ‘hooks’.

- creation of a novel algorithm to extract claims and associated evidence from different windows in near real-time from analysis of individual screenshots (such as the example in Figure 4.1), and from sequences of screenshots.

- empirical evaluation of our algorithms using over 121,000 screenshots from a study with 150 participants (consisting of students and knowledge workers) performing an online analysis task.

## 4.4 Approach

### 4.4.1 Passive Instrumentation System

Our approach builds on a passive instrumentation system [Jon16a] developed in our group that included several *instrumentation agents* for data gathering, as well as a backend query, analysis and visualization platform. The agents are passive in that, once installed on a computer, they require no input from the user. One of these agents, the *macOSinstrumenter*, runs on all Mac desktops and carries out both ‘traditional instrumentation’ (using hooks into the operating system and certain applications), as well as capturing screenshots of the user's desktop. Screenshots can be captured both at periodic intervals and when certain actions are performed by the user (such as mouse clicks or hitting the ‘Enter’ key). On the Mac, this is done using the *CGDisplayCreateImage* API [App16] in Objective C. Unfortunately, the main screen buffer does not include the mouse pointer cursor, since this is rendered in a separate overlay. To compensate for this, *macOSinstrumenter* also detects the cursor
position and re-inserts a cursor icon into captured screenshots. All screenshots (and corresponding instrumentation logs) are relayed to the backend analysis platform in close to real-time.

### 4.4.2 Screenshot analysis

The first requirement for analysis of the screenshots is to reproduce simple instrumentation from screenshots, such as active window detection. We make the simplifying assumption that users are working with three primary types of application: text editors for writing their ‘claims’, browsers for gathering ‘evidence’, and other applications unrelated to the primary sensemaking task (such as file browsers). In the case of the example in Figure 4.1, the editor application is *TextEdit*, and the browser is *Safari*.

For our initial analysis, we assume that users will be working with only one application window of each type. However, since users may have both editor and browser windows open simultaneously while creating their reports, a simple OCR of the screenshot will produce indistinguishable text merged from both the windows. Hence we propose the following approaches to detect both editor and browser window boundaries, and then to perform separate OCR on the appropriate regions.

1. **Detect active window**: the first step in recognizing the current activity of a user involves knowing which application is currently active. Different Operating Systems have unique characteristics that can be exploited for this purpose but typically, the application name (or a clue to it) is written on the screen in one of the few standard locations. In the case of *OSX* (now *macOS*), the active application name could simply be extracted from the top-left of the Apple Menu Bar, without sophisticated pattern matching on other parts of the window.

2. **Detect editor and browser window boundaries**: we assume that workers will be using a single text editor to produce a report on an assigned task, and a single browser to gather evidence. The sentences they write are the *claims* that they are making based on facts gathered from one or more data sources (most commonly a web browser). Accurately locating editor and browser window boundaries in a screenshot is essential to extract these claims. Our approach accomplishes this using a *template matching* algorithm [Lew95] implemented in *OpenCV* [Bra] to locate key template features of editor window corner edges, without the need to perform traditional boundary detection (such as *Canny* edge detection [Low99][Bay06]), which is notoriously messy.

3. **Extract text from windows**: we use *Tesseract* [Smi07], an open source OCR Engine for extracting the text from detected window boundaries. In automatic page segmentation mode, Tesseract outputs reasonably well segmented text even when there are multiple text segments in an image, such as in the example in Figure 4.1, which contains three columns of text in the browser.
4. **Filter for ‘main content’ in browser**: a typical webpage has several text blocks which include navigation menus, header, footer, etc. However, we assume that a user’s primary focus is the main content of the webpage and selecting only this content is a challenging task. After experimenting with several approaches based on contour and edge detection, text filtering after a complete OCR of the webpage produced the most accurate results. This text filtering approach relies on post-processing the segmented text output from the Tesseract OCR engine. We removed those lines that contained few words or only sentence fragments (except when immediately preceded by a longer fragment). Using this simple approach, we were able to remove most of the text from navigation menus and sidebars.

### 4.4.3 Algorithms for relating sensemaking artifacts

We assemble these ingredients into two algorithm variants: our *Stateless* algorithm (Algorithm 2) assumes that the evidence associated with a particular claim will be found in the same screenshot (as is the case in Figure 4.1); our *Stateful* algorithm (Algorithm 3) assumes that the evidence related to a claim can be found in any of the screenshots with an earlier timestamp than the one in which the claim was found.

Once text has been extracted from the relevant portions of the screen, we use a variety of different sentence similarity algorithms [Ach08], and threshold the scores (using $MIN, MAX$) to extract putative claim-evidence pairs.
Figure 4.1 A typical screenshot captured during an experimental run - this one is from Phase 3 (draft report writing) as described in Section 4.5.1. A participant is writing a report in TextEdit stating that they believe that Gary Johnson has the greatest chance of winning the 2016 election as a 3rd party candidate, and backing this up with evidence from a web page in Safari. Annotations on the screenshot show the active window indicator lights, the *macOSinstrumenter* (previously known as the *OSXInstrumenter*) icon in the menubar, the user’s claim from their report, and the associated evidence that we want to extract. Reproduced from [Jon17b] with permission. ©IEEE 2017.
Algorithm 2 Stateless Algorithm: extract claim-evidence pairs when they both occur in the same screenshot

1: function FINDSTATELESSPAIRS(Screenshot s)
2:   pairs ← empty set
3: if Editor is active in s then
4:   EditorBoundary ← extract boundary from s
5:   BrowserBoundary ← extract boundary from s
6:   claims ← OCR on EditorBoundary
7:   evidences ← OCR on BrowserBoundary
8: for each claim ∈ claims do
9:   for each evidence ∈ evidences do
10:      sim ← similarity(claim, evidence)
11:     if sim ∈ [MIN, MAX] then
12:        pairs.append(claim, evidence, sim)
13:     end if
14: end for
15: end if
16: return pairs
17: end function

4.5 Empirical Evaluation

4.5.1 Data collection

During 2014, an experiment was conducted with the NC State College of Humanities and Social Sciences (CHASS), with the purpose of studying sensemaking by constructing a carefully controlled experiment with university students and knowledge workers. The participants were given the task of choosing a candidate for the 2016 US Presidential Election that stood the highest chance of winning as a 3rd party candidate. This topic was deliberately chosen so that the majority of participants would have to search the web to find suitable candidates (most people know about Democratic and Republican candidates, but not so much about 3rd parties).

Each experimental run consisted of the following phases:

1. Initial test of reasoning abilities, using an online survey.
2. Practice task - searching the web to answer some simple questions about the 2012 election.
3. Main task - conducting research on the web to write a draft report that proposed a candidate and justified the choice using evidence from web pages.
Algorithm 3 Stateful Algorithm: extract claim-evidence pairs from different screenshots

1: function \textsc{FindStatefulPairs}(Screenshots $S$)
2:     \textit{pairs} $\leftarrow$ empty set
3:     $s_E$ $\leftarrow$ most recent editor screenshot
4:     $S_B$ $\leftarrow$ subset of $S$ preceding $s_E$
5:     \textit{Editor Boundary} $\leftarrow$ extract boundary from $s_E$
6:     \textit{claims} $\leftarrow$ OCR on \textit{Editor Boundary}
7:     \textbf{for each} $s \in S_B$ do
8:         \textit{Browser Boundary} $\leftarrow$ extract boundary from $s$
9:         \textit{evidences} $\leftarrow$ OCR on \textit{Browser Boundary}
10:     \textbf{for each} \textit{claim} $\in$ \textit{claims} do
11:         \textbf{for each} \textit{evidence} $\in$ \textit{evidences} do
12:             \textit{sim} $\leftarrow$ similarity(\textit{claim}, \textit{evidence})
13:             \textbf{if} \textit{sim} $\in$ [$\min$, $\max$] \textbf{then}
14:                 \textit{pairs}.append(\textit{claim}, \textit{evidence}, \textit{sim})
15:             \textbf{end if}
16:         \textbf{end for}
17:     \textbf{end for}
18:     \textbf{return} \textit{pairs}
19: \textbf{end function}

4. Group discussion to share thoughts and ideas.

5. Wrap-up task - editing the draft report as needed to create a final report.

6. Concluding survey.

Throughout each run of the experiment, \textit{macOSinstrumenter} was running in the background on the participants’ computers, recording all applications used, URLs accessed, and search terms entered. In addition, screenshots were automatically collected at 10-second intervals, and also on clicking a mouse button, or on hitting the ‘Return’ or ‘Enter’ key. The latter are called ‘Engagement-based’ screenshots and were intended to ensure that important moments (such as when a user switches windows, or enters a search term into the browser) were always captured in the screenshots. Data was recorded in this way for all phases, except for phase 4 (the group discussion), which was audio recorded.

In total 54 experimental groups were used, most with 3 participants (but a few with 2), making a total of 150 unique users. A total of 121,378 screenshots (totaling 399GB) were captured during the study - an average of 809 screenshots relating to each participant. After initial de-duping (to remove successive screenshots where nothing changed), the corpus\textsuperscript{3} was reduced to 85,551 screen-

\textsuperscript{3}the de-duped corpus is available on request from http://go.ncsu.edu/cic.
Each participant was issued with a MacBook laptop (late 2013 model) running the OS X Mavericks operating system. Each logged in using a fresh (anonymized) user account that was created for them. The desktop was not customized in any way, except we ran a script to remove many of the shortcuts in the task tray at the bottom of the screen. This was partly just to ‘de-clutter’ the screen but also to encourage them to use the ‘TextEdit’ editor for their reports, and the ‘Safari’ web browser for their research. A typical screenshot captured during Phase 3 of an experimental run is shown in Figure 4.1.

### 4.5.2 Data Characterization

A summary of metrics from the de-duped corpus of 85,551 screenshots is shown in Table 4.1. Just under two-thirds of these were obtained using the 10-second periodic trigger, with the remaining one-third produced as a result of mouse clicks or pressing the ‘Enter’ key. There were an average of 570 screenshots generated for each user. Our algorithms detected that ‘TextEdit’ was the active window in approximately 21% of screenshots, ‘Safari’ was active in 50% of screenshots and other applications (or just an empty desktop) in the remaining 29%. We found that about a third of active Safari windows corresponded to the two ‘survey’ phases of the study (i.e. phases 1 and 6 - these contained no useful ‘evidence’ for associating with user’s TextEdit reports, so were filtered out. Furthermore, a small number of active Safari windows (just under 4%) were partially obscured so it was difficult or impossible to confirm the window type - these were also removed. From the remaining active Safari windows, we were able to extract an average of 1,494 characters of text from each. We also looked for situations where a Safari window was in the background (typically behind a TextEdit window), such as the example in Figure 4.1. This situation occurred in about 21% of all screenshots - in fact, when a user had TextEdit open, they almost always also had a Safari window open in the background as well. From these backgrounded Safari windows, we were able to extract an average of 1,091 characters from each.

### 4.5.3 Results

We now present the results from evaluation of our algorithms - both for detection of active windows and URLs (as described in section 4.4.2), and for extraction of claims and evidence from different windows (as described in section 4.4.3).

#### 4.5.3.1 Active Window and URL Detection

We compared the active window events recorded by macOSinstrumenter with those detected by the screenshot analysis, across the entire corpus. Overall we obtained 98.15% agreement between the
Table 4.1 Summary of key metrics from the screenshot corpus [KJ17]

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of screenshots (after initial dedupe)</td>
<td>85,551</td>
</tr>
<tr>
<td>...captured using periodic trigger</td>
<td>54,324 (63.5%)</td>
</tr>
<tr>
<td>...captured using engagement trigger</td>
<td>31,227 (36.5%)</td>
</tr>
<tr>
<td>Total number of users</td>
<td>150</td>
</tr>
<tr>
<td>...average number of screenshots per user</td>
<td>570</td>
</tr>
<tr>
<td>Screenshots with TextEdit as the active window</td>
<td>18,014 (21.1%)</td>
</tr>
<tr>
<td>...average characters of user text from each</td>
<td>578</td>
</tr>
<tr>
<td>Screenshots with Safari as the active window</td>
<td>42,773 (50.0%)</td>
</tr>
<tr>
<td>...after filtering for 'survey' windows</td>
<td>28,785 (33.6%)</td>
</tr>
<tr>
<td>...after filtering for partially hidden windows</td>
<td>25,086 (29.3%)</td>
</tr>
<tr>
<td>...average characters of website text from each</td>
<td>1,494</td>
</tr>
<tr>
<td>Screenshots with Safari as the background window</td>
<td>18,300 (21.4%)</td>
</tr>
<tr>
<td>...average characters of website text from each</td>
<td>1,091</td>
</tr>
</tbody>
</table>

Table 4.2 Confusion matrix for active window detection, comparing our two approaches

<table>
<thead>
<tr>
<th>Screenshot \ Instrumenter</th>
<th>Finder</th>
<th>TextEdit</th>
<th>Safari</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finder</td>
<td>6,654</td>
<td>116</td>
<td>381</td>
</tr>
<tr>
<td>TextEdit</td>
<td>158</td>
<td>17,958</td>
<td>199</td>
</tr>
<tr>
<td>Safari</td>
<td>220</td>
<td>453</td>
<td>56,674</td>
</tr>
</tbody>
</table>

two instrumentation methods. A confusion matrix is shown in Table 4.2. There were also 1,552 active window events that related to other types of windows (such as Terminal) that were rarely used. The timestamp resolution we used was 1-second, which meant that it was possible for a window transition to occur in-between a reading from the macOSinstrumenter and a screenshot being taken during the same second, which we think accounts for most of the discrepancies. Also, we suspect it is possible for a particular window to be shown as being active (via the label in the top-left of the screen) for a fraction of a second after the system has already produced a window deactivated notification (e.g. example B in Figure 4.4).

We were also able to extract some URLs being browsed in Safari directly from the screenshots, although without the same level of completeness and accuracy as is possible from a direct call in software (macOSinstrumenter actually uses an AppleScript command for this). As is common with other modern browsers, Safari usually only displays the domain in the address bar, as opposed to the full URL; it is also possible to type search terms directly into the address bar, in which case the resulting URL to deliver the search results never gets shown. For these reasons, we recommend URL detection be carried out using ‘traditional’ instrumentation methods.
Table 4.3 Summary of results from Stateless and Stateful algorithms

<table>
<thead>
<tr>
<th>Metric</th>
<th>Stateless</th>
<th>Stateful</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextEdit sentences extracted ('claims')</td>
<td>109,465</td>
<td>1,450</td>
</tr>
<tr>
<td>Safari sentences extracted ('evidence')</td>
<td>89,669</td>
<td>100,124</td>
</tr>
<tr>
<td>Claim-evidence matches (0.3 threshold)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...from Difflib char sequence similarity</td>
<td>19,558</td>
<td>110,457</td>
</tr>
<tr>
<td>...from BLEU similarity</td>
<td>45,001</td>
<td>297,931</td>
</tr>
<tr>
<td>...from Jaccard (word) similarity</td>
<td>212</td>
<td>275</td>
</tr>
<tr>
<td>...from TF-IDF and cosine similarity</td>
<td>-</td>
<td>887</td>
</tr>
<tr>
<td>...from WordNet similarity</td>
<td>-</td>
<td>37,251</td>
</tr>
<tr>
<td>Claim-evidence matches (0.15 threshold)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...from Difflib char sequence matching</td>
<td>74,013</td>
<td>485,648</td>
</tr>
<tr>
<td>...from BLEU similarity</td>
<td>58,198</td>
<td>371,560</td>
</tr>
<tr>
<td>...from Jaccard (word) similarity</td>
<td>1,340</td>
<td>3,711</td>
</tr>
<tr>
<td>...from TF-IDF and cosine similarity</td>
<td>-</td>
<td>9,701</td>
</tr>
<tr>
<td>...from WordNet similarity</td>
<td>-</td>
<td>478,738</td>
</tr>
</tbody>
</table>

4.5.3.2 Claim and Evidence Extraction

We ran the stateless and stateful algorithms in section 4.4.3 using several different sentence similarity methods to match up claims to putative evidence. A summary of the number of claim/evidence pairs extracted using each variant can be seen in Table 4.3. It can be seen that many more TextEdit sentences appear to have been extracted in the Stateless case - this is simply due to the fact that only one copy of the TextEdit report was used in the Stateful algorithm (the one with the largest file size) whereas every TextEdit window was analyzed in the Stateless case. We also observe that more text was extracted from Safari windows in the Stateful case since this included screenshots where no TextEdit window was present; in the Stateless case, text was only extracted from background Safari windows.

The remainder of Table 4.3 shows the number of putative claim-evidence pairs extracted using five different sentence similarity algorithms. It can be seen that, for a given threshold, we always extract more putative matches using the Stateful algorithm variant. Furthermore, we can always extract arbitrarily more matches simply by lowering the similarity threshold. Two \textit{MIN} threshold values (0.15 and 0.3) are chosen for the table for illustrative purposes only. Whilst the scores obtained from each sentence similarity method cannot be directly compared, they do allow us to compare the stateless and stateful variants of each algorithm. In some cases, the stateful variant produced many more matches for a given threshold (such as for \textit{Difflib} [Bla04] character sequence analysis and \textit{BLEU} [Pap02], a common metric used for comparing machine translations). However, the quality of these additional matches was found to be quite noisy, with sometimes entirely irrelevant (or text fragment) sentences being matched (as can be the case with character-based methods). Conversely, Jaccard
Table 4.4 Likert scoring for quality of claim-evidence associations

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Poor - nonsense, no association between sentences.</td>
</tr>
<tr>
<td>2</td>
<td>Fair - some association but evidence does not match claim.</td>
</tr>
<tr>
<td>3</td>
<td>Good - associated sentences and weak claim/evidence.</td>
</tr>
<tr>
<td>4</td>
<td>Very good - looks likely that evidence matches claim.</td>
</tr>
<tr>
<td>5</td>
<td>Excellent - evidence clearly matches claim.</td>
</tr>
</tbody>
</table>

similarity (based on exact matching of complete words) produced less output for a given threshold but the output consisted of much ‘cleaner’ sentences. Finally, we tested the well-known approach of TF/IDF with cosine similarity, which is inherently stateful since it relies on a corpus of text, and also the synonym-based approach from WordNet [Mil95].

Next, we carried out a human evaluation on the extracted claims and evidence to try to assess the quality of the associations. We asked five staff volunteers at our university to manually assess the top 100 highest scoring claim/evidence matches for three of the approaches listed in Table 4.3 - we chose Stateless and Stateful Jaccard (i.e. exact word matching) in order to test our initial assumption that users are time constrained and are likely to directly re-use the same wording in their claims as in their evidence. To counter this, we also evaluated the output from Stateful WordNet, where words may be replaced by synonyms. For the human assessment, we used a 5-point Likert scale, using standard descriptions for assessing ‘quality’, as described in Table 4.4. We also provided an option for raters to ‘reject’ a claim-evidence pair if the text in both was identical or if it appeared that a user’s claim was shown in the evidence column, which may indicate a processing error. A summary of Likert results obtained from the human assessment are shown in Figure 4.2. Of the three methods evaluated, Stateless Jaccard provided the highest rating scores on average, followed by Stateful Jaccard and lastly Stateful WordNet. This confirmed our suspicion that most users were not substantially re-wording the evidence they gathered. However, it should be noted that Stateless Jaccard also contributed to the most rejected pairs, which indicated a higher instance of the ‘problematic image’ cases described in section 4.6.

In order to determine the level of inter-rater agreement, we ran a Fleiss’ Kappa [FC73] analysis, which produces a score from -1 to +1 to estimate the level of agreement between raters for each score; the results for Stateful Jaccard (the algorithm for which we had the most even distribution of scores) can be seen in Table 4.5. These indicate that there was slight agreement over which pairs were likely to indicate a processing error (score 0), and for scores 1 and 2. Inter-rater agreement was best on what constituted ‘excellent’ claim/evidence matches but very poor on what constituted scores of 3 and 4. This is perhaps unsurprising since there is inevitably a high degree of subjectivity on the strength of claim/evidence associations for the intermediate scores.

Finally, we wanted to measure the extent to which the human scores correlated with the com-
Figure 4.2 Results from human ratings of the quality of claim-evidence associations produced by three different variants of our algorithm. Scores of 1 and 2 in our 5-point scale (described in Table 4.4) are shown in dark and light red respectively; 3 is shown in gray; 4 and 5 are shown in light and dark blue respectively. The horizontal scale represents the percentage of respondents. Some of the pairs were rejected and flagged by the raters, usually because the claim and evidence were exact duplicates (which may indicate one of the problems in Figure 4.4 occurred); the total number of non-rejected ratings supplied by the human raters for each technique is listed on the right (out of a total of 500 ratings). ©IEEE 2017.

Table 4.5 Fleiss’ Kappa Statistics for Stateful Jaccard ratings

<table>
<thead>
<tr>
<th>Rating</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td>0.19</td>
<td>0.12</td>
<td>0.16</td>
<td>0.02</td>
<td>-0.03</td>
<td>0.28</td>
</tr>
</tbody>
</table>

puter scores produced by the sentence similarity measures. The results for Stateful Jaccard, which displayed the most uniform distribution across the likert scale, are shown in Figure 4.3. Overall we obtained a Pearson correlation coefficient of 0.66, which indicates a moderate to strong positive correlation, although it is clearly far from perfect agreement.

4.6 Discussion and Research Directions

These initial results show that, at least sometimes, it is possible to extract high quality claim-evidence pairs from text editor and web browser windows. Some examples of associations that were scored ‘5’ by the raters are shown in Table 4.6. However, it should be noted that our analysis was conducted in a controlled setting, and we did not take into account different operating systems or attempt to extract text from arbitrary application windows. Despite the controlled setting, we still encountered a number of problematic cases, in which active window detection and/or text extraction became more difficult, such as those illustrated in Figure 4.4. These kinds of difficulties will vary across operating systems.
Figure 4.3 Human vs Jaccard similarity score comparison for the Stateful Jaccard algorithm. Pairs are sorted by the average human rater score (which has been normalized to the range 0-1), and the corresponding Jaccard scores are shown in Blue.

Table 4.6 Examples of high quality claim-evidence associations extracted by our algorithms, which were given a score of 5 (Excellent) on our Likert scale by human raters.

<table>
<thead>
<tr>
<th>Claim</th>
<th>Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the 2012 election, Johnson received almost 1% of the popular vote, which was more than the other third party candidates combined.</td>
<td>The Johnson/Gray ticket received 0.99% of the popular to 1.27 million votes, more than all other minor candidates combined.</td>
</tr>
<tr>
<td>He supports energy exploration, gun rights, gay rights, and a free market.</td>
<td>He is fiscally conservative and socially libertarian, believing in energy exploration, gun rights, gay rights and a free market.</td>
</tr>
<tr>
<td>When I last voted, I felt that I was voting for the lesser of two evils.</td>
<td>How many people felt they were voting for the lesser of two evils?</td>
</tr>
</tbody>
</table>
Figure 4.4 A small selection of somewhat problematic screenshots: (A) Safari window is maximized, which means the characteristic ‘lights’ and header are not available for detection, (B) TextEdit window is partially maximized but the OS reports Safari as being on top, maybe because the user just clicked on the icon in the task bar, (C) the user has two TextEdit windows open simultaneously, which violates our assumption of a single editor window, (D) text highlighting and spell-checker underlining make it difficult to carry out accurate OCR on the TextEdit text.
Furthermore, we did not address the research question of how to identify ‘claims’ and ‘evidence’ in arbitrary unstructured text, relying instead on sentence similarity between text extracted from known applications. New research in this space, such as from argumentation structure analysis, would allow our algorithms to become useful in a much wider variety of settings. The challenge is also made much more difficult by the possibility that users have not just re-phrased a piece of evidence in order to make a claim but they have actually made an interpretation of the evidence. Such interpretations will likely score badly using our sentence similarity techniques but may actually be excellent examples of claims and evidence. For instance, one of the participants wrote ‘The current declared third party candidate list is bleak.’ after looking at the list of 3rd party and independent candidates on a website. He/she determined that the list looked bleak, even though this was not directly stated on the web page.

Overall though, we were able to demonstrate the potential of screenshot analysis techniques for extraction of simple claim-evidence pairs in a controlled setting. There are many potential next steps for building on this work - firstly, it would be interesting to determine if extracting more context for the ‘evidence’ would lead to better human ratings (such as the sentences preceding and following the one that matched). Secondly, it may be that other sentence similarity methods, such as ‘document vectors’ doc2vec [LM14] or fasttext [Jou16], could produce higher quality matches. Finally, image processing techniques are needed to handle screenshots pertaining to a wider range of different operating systems and application windows. It might be that new unsupervised approaches to identification of generic features in images (such as important blocks of text and invariant features of certain types of applications) could be applied here. Currently, Convolutional Neural Networks, would seem like the best bet.

To summarize, this chapter has presented a first-of-a-kind study to determine the feasibility of screenshot analysis as an alternative approach to instrumenting the working patterns of knowledge workers, and for associating claims they are making in written reports with evidence from web-based data sources.

Through a controlled experiment with 150 participants, we demonstrated that it is possible to gain insights into basic sensemaking processes while carrying out a complex task. We showed how a combination of image processing and natural language processing techniques can be used to automatically extract portions of text corresponding to claims made in reports, and to associate these with the likely evidence from web pages that were used to support the claims. This work opens up an exciting new research field that could eventually lead to algorithms that pro-actively help computer users overcome cognitive biases when analyzing complex information.
5.1 Motivation

From our initial attempts to analyze passive instrumentation data alone (section 3.6), it became apparent that there is a need to actively capture ‘why’ a user’s activity is taking place (i.e. to which tasks and goals it relates) and, critically, to be able to relate the activity to that of their collaborators. From activity data alone, it is possible to predict what activity may occur next with some accuracy, but not enough to provide recommendations that are accurate enough to not be annoying to users. Also, in a multi-tasking environment, there is often simply too much overlap between tasks to be able to reliably separate them (as we found in the analysis in section 2.4).

It is our desire to train effective machine learning models to be able to predict task labels given streams of multi-tasking activity data from multiple users (as per Chapter 2). To enable this, some proportion of labelled data is necessary. The interfaces we created for capturing this labelled data are referred to as our active Journaling interfaces and are the focus of this chapter. They are active in the sense that they require active participation from users.

This chapter addresses the core research question of how appropriate journaling interfaces can be designed to systematically capture tasks and goals from a user in a way that is acceptably unintrusive and that integrates with collaborative workflow. Although we knew what information we wanted to capture (i.e. what a user is doing and equally importantly, why), we did not know how to capture that data without disrupting a user’s workflow.
Responding to this concern, an investigation was carried out with LAS staff that employed an *ethnographic* approach [RC06] to design: studying the work patterns and note-taking habits of potential users. Using snowball sampling, we recruited seven analysts and researchers at the LAS. Each was observed for 2-3 hours, after which semi-structured interviews (30-40 minutes) were conducted with each participant. The observations were then summarized, and each was member-checked with the participant. This allowed a codebook of recurrent drives, concerns, work and annotation habits (inclusive of software preferences) to be created.

Based on these results, we identified three places where intelligent user interaction is most required: (1) to associate tasks and goals with documents and URLs *at the time of their usage*, (2) to reflect this information back to users and allow them to directly benefit from it, as well as to correct goal labels if needed, and (3) to visualize the data in aggregate for situational awareness. These requirements focussed our subsequent research on Journaling.

### 5.2 Related Work

Many academic and commercial projects have attempted to build systems that allow users to capture their thoughts and ideas - for brevity, we restrict our review specifically to those that are concerned with capturing task and goal labels from users, as opposed to other types of work-related information.

A well-known research tool that attempted to capture user tasks in a desktop context is *Task-Tracer* [Dra05][She09][Stu05], which provided an interface for logging user goals in Windows XP. This project had similar goals to ours but the interfaces they developed were found to be disruptive to a user’s flow and did not provide collaborative capabilities.

An ambitious program called *GlassBox* [Cow05] attempted to create an ‘instrumented infrastructure for supporting human interaction with information’, and they also trialled this on intelligence analysts. The team mentioned the difficulties of instrumenting specific applications at a low level, which is something we have deliberately avoided. They also described the core difficulty of gleaning information on the tasks and goals the analyst is attempting to achieve, which is essential for making sense of the passive instrumentation data. This is exactly the core problem that we are looking to solve with our interactive Journaling interfaces.

Task associations were also central to the *Mylar* study [KM05], which attempted to associate program elements in a software codebase with user tasks, in order to minimize time spent navigating files. However, this was limited to the *Eclipse* programming platform.

We anticipate that task association technologies will become critical to the new generation of ‘Smart Digital Assistant’ technologies that are coming into widespread use. These systems attempt to provide timely and relevant information to their users but only in limited (typically non-work-related contexts). A popular example is *Siri*, which uses an *Active Ontology* approach [Guz07] to ac-
Table 5.1 Comparison of features supported by similar journaling tools in industry and academia. SDAs refers to ‘Smart Digital Assistants’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>SDAs [Guz07]</th>
<th>Mylar [KM05]</th>
<th>TaskTracer [Dra05]</th>
<th>GlassBox [Cow05]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leverage Passive instrumentation</td>
<td>✓ limited</td>
<td>✓ limited</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Use Content-based features</td>
<td>✓ limited</td>
<td>✓ limited</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Active task labelling</td>
<td>× ✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Attempt to minimize disruption</td>
<td>✓ ×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Shared task taxonomy</td>
<td>× ✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Real-time user feedback</td>
<td>× minimal</td>
<td>not streaming</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

complish a limited range of tasks. Other similar tools include Google Now and Cortana (references easily found on the web). Typically these and other services lack broader contextual information about user’s working patterns and goals, and so are limited in the scope of advice they can provide; they cannot yet learn about new tasks automatically from many individuals. To enable this, it is critical to associate user activity with tasks that they are carrying out.

A brief comparison of some of the Journaling tools described here is shown in Table 5.1. To summarize, the principle limitations of these tools that we have sought to address are:

- they do not attempt to minimize user disruption while capturing task and goal labels - most try to gather as much information as possible, sometimes in the form of free-form text boxes.

- they lack a shared goal/task taxonomy, which means that these tools cannot support collaborative working where tasks are shared between more than one user.

- they provide minimal support for continuous feedback on who is working on what - this is essential to providing a useful collective understanding of knowledge worker’s information needs.

In summary, several research tools (most notably GlassBox [Cow05] and TaskTracer [Dra05] have been developed for capturing user tasks and goals but were found to be too disruptive to be viable on an ongoing basis and did not provide users with enough immediate benefits to offset this. Our aim is to iteratively build end-to-end prototypes that fully leverage rich context on the tasks that individuals or teams are trying to achieve, as well as the information and workflows needed to achieve them. We anticipate that this information will be vital to enabling next generation Smart Digital Assistant technologies that can begin to learn about tasks in any domain. This chapter describes some very preliminary steps towards this.
5.3 Major Contributions

This chapter describes the following major contributions:

- Design of a novel means for systematic capture of tasks and goals in a way that integrates with collaborative workflow. These tasks and goals can easily be related to information artifacts and to user actions logged through passive instrumentation (Chapter 3).

- Demonstration of initial benefits to users by reflecting their data back to them in real-time, and to managers and supervisors through aggregated visualizations, and through measuring aspects of team collaboration.

- Empirical studies with three different groups, consisting of graduate students and intelligence analysts, that demonstrated that this approach is suitable for use by diverse groups of knowledge workers over extended periods of time.

5.4 Creating a Taxonomy of Tasks

Before it is possible to label information artifacts with tasks and goals, it is necessary to tackle the question of how to create a taxonomy of users' tasks and goals in such a way that they can be shared with collaborating users, and accessed quickly and easily.

Some previous work has been carried out into the creation of taxonomies for human goals [Chu01] but the literature is sparse on how to create generalizable taxonomies for work-related projects and tasks. An alternative strategy is to adopt the approach of allowing a folksonomy [Hel11] to emerge but this risks introducing unnecessary inconsistency in labelling of collaborative tasks. We decided to initially adopt a hybrid approach, where some tasks are pre-defined in a static taxonomy, and others are user-definable. This ensures consistency of representation for some tasks but allows for custom representations of others to evolve.

For representation of a user's task hierarchy, we introduce the idea of a ‘task-tree’, where high-level tasks are successively broken down into lower level sub-tasks. This breakdown is inspired by the Pomodoro technique [Cir06], which suggests that humans can only focus optimally for periods of up to half an hour. From attempting to model my own work over the course of several weeks, I found that the tree generally required a minimum of four levels to break down to Pomodoro-sized tasks. These four levels roughly corresponded to: Strategic Goals, Tactical Goals, Tasks and Sub-tasks. Each tended to have different defining characteristics, as described in Table 5.2.

One of the main motivations behind this work is to develop tools for monitoring the progress of collaborative teams towards strategic and tactical goals at an organizational level. Strategic Goals in a workplace may involve the development of a new product, for which several separate projects...
are required. Each of these might map well to my ‘Tactical Goal’ level. In turn, each project is typically broken down into a number of tasks and sub-tasks. Typically, higher levels of the tree would be defined by leaders or managers and lower levels by individuals. Also, we expect higher levels to correspond to longer-lived tasks than lower-levels, and hence to be slower to change. Our solution makes all of this configurable (so a task-tree can equally be applied to individual and team tasks).

5.5 Labeling Information Artifacts

As we discovered from analysts in section 5.1, it is important to capture task labels for documents at the time of their usage - unsurprisingly, it is even more disruptive to users to have to go back and label documents after they have been used. However, documents are likely to be used more than once and therefore it is appropriate to provide a mechanism for task labels to be added or corrected on subsequent usage as well as on initial usage. Furthermore, if we are reflecting labelled documents back to users through the ‘recent-work dashboard’ (introduced in section 1.1), it makes sense to allow them to correct any labels, especially those that have been added by an algorithm. This section describes our interfaces for active journaling both during initial document use, and afterwards through the dashboard.

5.5.1 Requesting labels during initial document use

After studying previous task-labelling prototypes [Dra05; Cow05], which used drop-down menus/lists or auto-complete text boxes, we hypothesized that perhaps a visual representation of the task-tree itself would be a good initial approach, since this could provide an intuitive and easily-navigable representation. Several options for this were prototyped (implementation details beyond the scope of this report) but I eventually settled on a simple horizontally-expanding annotated tree design. Figure 5.1(A) is an example of the interface for user-selection of a task, which shows a small part of a user’s ‘task-tree’. Figure 5.1(B) shows a separate interface for editing the task-tree, either for a group or for an individual.

Whenever the system requires a task or goal label, it simply displays the task-tree interface and asks for a user selection. We refer to this as ‘nanoblogging’ since usually only one click is needed.
to select a goal. Zooming and scrolling in the tree is done using multi-touch gestures, as supported on most new desktop and mobile devices. However, the interface tries to minimize scrolling by remembering the user’s last task selection and tree layout - every non-leaf node can be expanded or contracted to show or hide child nodes. Crucially, on selecting a node, the interface immediately displays a list of other users working towards the same task, which fosters collaboration and prevents duplicated effort.

Ideally, we wanted to be able to associate *every* document and web page accessed by a user with a fully-specified task. However, we recognized that, in the absence of sophisticated task-document association algorithms, this was likely to impose an unacceptable burden on users.

For our first study (described in section 5.7.2), we prompted the users to select a goal from the tree for every new URL accessed. Based on feedback from this, we subsequently implemented three options for triggering a prompt to the user: (1) sampling - only prompt for one-in- every x new URLs, (2) prompt only after at least y seconds spent viewing a particular page, inspired by research on web page ‘dwell time’ [Liu10], (3) prompt only after scrolling down z% of the page contents. All values for
x, y, and z are user-configurable. In our second study (described in section 5.7.3), we evaluated the impact of these options on user experiences and on the number of labels provided. To the best of my knowledge these prompting options have not been used in any of the other prototype systems described in section 5.2.

When triggered, the task-tree interface can be displayed in a variety of ways. Initially it was integrated into the ChromeInstrumenter, a web browser extension described in section 3.4.2). In addition to displaying the task-tree, the extension also acts as an instrumentation agent. The interface can be displayed from other agents as well, such from our native MacOS application, MacOSInstrumenter, described in section 3.4.1.

In summary, although this interface is intended for active information gathering and, as such, could be distracting, it provides three immediate benefits to users that should help offset this: it serves to remind the user what they are working on, it shows them other tasks that are associated with the one they’re currently working on (thereby acting as a prompt for what they might want to do next), and finally it shows them who else is working towards the same goals, thus promoting collaboration.

5.5.2 Requesting labels through the dashboard

Recall that our ‘recent-work dashboard’ interface, as shown in Figure 1.2, provides each user with a constantly up-to-date page for keeping track of recently used information artifacts used both by themselves and/or by their colleagues. The dashboard receives a live feed of events from the instrumentation and journaling platform as documents and URLs are accessed by users, and organizes this information according to tasks. The currently active task is always shown at the top and highlighted.

Tags labels for any particular document or URL can be modified using the ‘Tag’ drop-down menu, which provides options for Adding or Removing tags. When selecting the ‘Add Tag’ option, the task-tree interface is displayed for selecting a new task. Note that each document or URL can be associated with any number of different tasks. On selecting the ‘Remove Tag’ option, the document is removed from its current task block. If a document has no task labels, it resides only in an ‘Untagged’ block at the bottom of the dashboard.

In the case where labels have been applied automatically (such as from the network fusion approach in Chapter 2), we automatically insert the document into the relevant task block, and we add an additional interface element to the dashboard, as shown in Figure 5.2. The ‘feedback icon’ allows feedback to be provided by the users if they wish. A confidence value for each association, $c_i$, such as that defined in section 2.1.1, is also communicated to users through the background color of the icon: green represents high confidence, amber represents medium confidence and red represents low confidence. All user feedback provided is sent back to the task-document association algorithm and may be used to add, delete or update the confidence on edges in a graph or other data structure.
Figure 5.2 Section of a user’s ‘recent-work dashboard’. The top document has been labelled by the user as belonging to the ‘Objective Setting’ task (the currently active task, highlighted in yellow). The next document has been automatically assigned to this task by the Network Fusion algorithm, and includes an icon with feedback buttons. Clicking the tickmark confirms the assignment (and the confidence for the association will be set to 1.0); clicking the cross will move the document down to the ‘Untagged’ group and the association will be broken. ©ACM 2017.

If the confidence of a given association falls below a threshold (which can be user-defined), a task label will not be automatically assigned and the document will simply remain in an ‘Untagged’ block at the bottom of the dashboard. In this way, precision and recall can be traded by the user according to their individual tolerance for correcting erroneous labels versus manually adding labels from the ‘Untagged’ block.

5.6 Benefits to Users and Managers

So far in this chapter, we have described two interfaces to allow task labels to be systematically captured from users - the journaling task-tree interface, and the recent-work dashboard. While both interfaces are designed to integrate with user’s workflow and minimize disruption to them, providing task labels is inevitably a disruptive activity to some extent. Even with sophisticated algorithms to ‘guess’ the majority of labels, some labels will still be required from users in order to train these algorithms. In order to offset the disruption this will cause, the interfaces themselves must provide benefits to the users; otherwise they will only ever be of use as research tools, a fate that befell several of the previous systems described in section 5.2.

Such benefits may be intended for individual users, or for their managers (or instructors in a classroom setting). For users, we hypothesized that the benefits may be threefold:

1. Information recall when task switching - the recent-work dashboard is designed to quickly
assist in ‘restoring state’ when switching between tasks (for instance, through the ‘Open All’ button that allows all documents in use for a particular task to be re-opened at once). Furthermore, the journaling task-tree interface is designed to ‘remind’ people about other tasks they might want to switch to, by displaying related tasks in the tree in close proximity.

2. Self-awareness - the data gathered from instrumentation and journaling can be aggregated and visualized in various ways that may provide insights into how people are accomplishing their tasks, and on which sub-tasks they are spending the most time.

3. Fostering Collaboration - both the task-tree interface and the dashboard include ‘user icons’ that provide a list of other users working towards particular tasks, as can be seen in Figures 1.2 and 5.1. It is hoped that these will be appreciated by users as mechanisms to foster collaboration opportunities on particular tasks.

For managers, we hypothesized that variants of the second and third benefits may particularly apply. In the case of self-awareness, managers (or class instructors) may be more interested in gaining awareness of the overall breakdown of time spent on tasks across their team (or class). Similarly, in addition to fostering collaboration opportunities across their teams, managers may also be interested in measuring the levels of collaboration across the individuals in their teams. We hypothesized that the data we are collecting could be utilized to gain an approximate idea which users are working together.

We set out to evaluate these benefits with three real-world groups of knowledge workers - two classes at NC State University and one group of Intelligence Analysts at the LAS. Of course, an ulterior motive for these trials was also to gather labelled document corpora for training machine learning classifiers, but we wanted to take the opportunity to evaluate the interfaces at the same time, to determine whether or not they would be suitable for continuous use, rather than just for short-term research.

5.7 Empirical Evaluation

5.7.1 Approach - Iterative Evaluation Studies

We carried out three experimental deployments of the Instrumentation and Journaling platforms to evaluate them and gather datasets for further research. For each study, we sought out users of varying technical savvy and disciplinary backgrounds: the first involved a graduate computer science class; the second a graduate political science class; and the third a group of real-life intelligence analysts. Respecting the educational contexts of our (first two) deployments, we worked closely with instructors to modify and situate the prototype in a manner that the instructors felt would be educationally beneficial. Towards this end, we employed the Design-Based Research (DBR) approach.
Figure 5.3 (A) ‘Icicle plot’ showing breakdown of a subset of events for four users. Each row corresponds to a level in the task hierarchy. Each column corresponds to a task or sub-task, where the width of each column (or ‘icicle’) corresponds to the number of events related to that activity. (B) ‘Task-tree graph’ showing activity breakdown for User 1 in the class. Larger, darker, circles indicate more activity for a particular task, and totals are carried cumulatively upwards through the tree. Users or teams can see very quickly from this visualization where they have spent their time. Note that plot (B) is a more recent snapshot of User 1’s activities and contains more sub-tasks. ©IEEE 2016.

[BS04], predicated upon: rapid iteration of the prototype based on student/instructor feedback; documentation of problems and consequent interventions; observation of students as they work with the evolving prototype, allowing participants to suggest modifications that would improve usability. The goal of DBR is twofold: the development of an educational product and to provide a process for the product’s deployment in classroom settings. DBR has drawbacks though: the constant interventions and changing nature of the prototype complicate the variables at play, limiting generalizability of findings or isolation of causes; however, we feel that its benefits (facilitating prototype iteration, development of instruction, and articulation of use-cases) justify its use at this stage of development. The sections below document how we iterated the prototype in order to bring it closer to production in a socially responsive way.
5.7.2 Computer Science Class

Our focus for this study was to evaluate the usability of the interfaces and to gather an initial labelled document corpus for text mining. The 21 graduate CS students used standardized Ubuntu Virtual Machines, each running the ChromeInstrumenter. To provide passive instrumentation, we wrote a bash script that logged application and document/file usage in Ubuntu, then sent events to our back-end. We configured the ChromeInstrumenter to prompt users to select a goal on every new URL. Users selected goals from a tree, structured around the hierarchy of homework and project assignments. As per the levels defined in Table 5.2, the ‘strategic goal’ was to pass the class; ‘tactical goals’ related to types of assignments; ‘tasks’ to specific assignments and ‘sub-tasks’ to activities within those assignments (e.g. Review Papers, Implement Code, Test Code etc).

A summary of user participation can be seen in Table 5.3 (refer to column for Study A). 17 of the 21 students provided at least one labelled event per week, and were deemed as ‘active’; data from the others was dropped. The passive instrumentation recorded usage of 1085 unique documents (including URLs), most of which were opened repeatedly as the class progressed. Using the task-tree journaling interface, each of the users labelled an average of 77 documents each, which were related to 75 different tasks (of the 134 available in the tree). The average number of users per task (our ‘Collaboration Index’) was only 1.9, although this rose to 2.7 and 4.9 respectively with tasks expanded to only 3 or 2 levels respectively (as per Table 5.2). A useful feature of this scheme is that users can broaden their pool of potential collaborators by reducing the specificity of their tasks.

Derived from this data, we provided the students with visualizations of their work activities (for visual reference, see Figure 5.3; plot-styles described in section 5.6); the instructor was given access to all visualizations. Individual visualizations allowed students to see where they focused their time and, by extension, where their attention was lacking (e.g. did they forget to test their code for Project P2?). Collectively, these visualizations allowed the class instructor to see how and when students were working on a given assignment - information that helped her to decide whether to extend deadlines or modify assignment requirements. Furthermore, the captured data showed significant diversity between users working towards similar goals, especially in terms of how they chose to spread the workload for assignments.

At the end of the study, we conducted in-person surveys/interviews with volunteer participants (n=6), consisting of likert-scale questions, short response questions, and a visualization prompt. The bulk of the questions focused on the utility/usability of the prototype; others solicited feedback about problems, solutions, and possible directions the project could take. Short response questions were read aloud, and the answers were transcribed (in brief) by the interviewer. The summarized responses were then checked by the participant; if any were deemed inaccurate, the participant was able to correct them. The results from selected questions, from which we learned the most, are shown in Figure 5.4).
Table 5.3 Participation and Engagement Metrics obtained from the 3 studies

<table>
<thead>
<tr>
<th>Metric</th>
<th>Study A: CS Class</th>
<th>Study B: PS Class</th>
<th>Study C: Analysts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of participants (Active/Total)</td>
<td>17 / 21</td>
<td>10 / 10</td>
<td>6 / 7</td>
</tr>
<tr>
<td>Passive Instrumentation Metrics:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total events generated by the passive instrumentation (Average per user)</td>
<td>141261 (8309)</td>
<td>5035 (504)</td>
<td>74090 (12348)</td>
</tr>
<tr>
<td>Total unique information artifacts logged</td>
<td>1085</td>
<td>916</td>
<td>2536</td>
</tr>
<tr>
<td>Average application/file events per user</td>
<td>8035</td>
<td>-</td>
<td>10402</td>
</tr>
<tr>
<td>(not enabled for Study B)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average URL events per user (enabled for all studies)</td>
<td>275</td>
<td>504</td>
<td>1946</td>
</tr>
<tr>
<td>Active Journaling Metrics:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total task labels submitted (Average per user)</td>
<td>1308 (77)</td>
<td>4096 (410)</td>
<td>3780 (630)</td>
</tr>
<tr>
<td>Unique task labels: submitted by users / total available in task-tree</td>
<td>75 / 134</td>
<td>20 / 20</td>
<td>33 / 52</td>
</tr>
<tr>
<td>‘Collaboration indices’: mean number of users per 4, 3 and 2-level goal</td>
<td>1.9, 2.7, 4.9</td>
<td>-, 3.5, 6.0</td>
<td>1.3, 1.8, 3.5</td>
</tr>
</tbody>
</table>

The surveys indicated that our participants generally: understood how to use the different features of the application and would consider using refined versions of it; however, a number of them found it to be disruptive to the work flow because of compatibility issues, absent features, or frequent “goal” promptings – leading some students to bypass elements of the application. With regards to the promptings, one student noted that, though disruptive, the prompts imposed “a systematic way of doing work”, that he found to be beneficial. With regards to the visualized reports, some students said that they preferred to “think in lists”; others found them refreshingly different and actively sought out new ways to visualize their goals and activities. Based on this feedback, we revised the prototype by: creating a cleaner set of options for ChromeInstrumenter, adding more file types to the recent-work dashboard, moving away from the Ubuntu VM (the cause of many problems), and adding several new triggering options for the task-tree interface (reducing interruption from prompts). Our interfaces are now flexible enough to run on any platform (e.g. Linux/Ubuntu, Mac OSX and Windows desktops), although varying levels of passive instrumentation are available for each. Having revised our instructions (now a combination of written & video), our preliminary results suggest that users can develop fluency with the three core interfaces in less than an hour.

The CS experiment has provided a corpus that we intend to make publicly available for analysis by researchers. We are now analyzing the full corpus collected from this class and developing techniques for: automatic task classification based on text content, document prediction / recommendation, and for the creation of task-centric search capabilities. Furthermore, this experiment drew our attention to an interesting question - can this application, by encouraging systematic working in
students, lead to positive educational outcomes? Furthermore, how might this system be refined to better help an instructor understand what is happening in her class?

5.7.3 Political Science Class

Our second study involved a graduate political science class of 10 students, who were required to perform a web-based security threat analysis (i.e. review and evaluate dozens of references, before synthesizing them into a coherent report). In this study, our primary interests were threefold: testing the utility of our interfaces among humanities/social-science students, refining the end-to-end application, and expanding our data corpus. However, these goals remained tempered by the principles of socially responsive design [Bar05]: discovering the instructor’s and student’s needs, then working to modify our application to better serve the requirements of the class.

To evaluate our interfaces for this type of analysis task, we conducted the study using a task-tree that reflected the structure of typical information analysis tasks, as validated by recent research [DC15b]. We based it on the six key steps in an ordinal structure for a generic analysis workflow, breaking each step into more specific tasks only when necessary:

1. Understand requirements: identify customers and their priorities.
3. Obtain data: construct queries and retrieve output.
4. Process data: extract meaning from the data and investigate any anomalies.
5. Interpret data: construct logical arguments that can be used to explain findings.
6. Communicate conclusions: record findings and express levels of confidence.

Figure 5.4 Results from a small selection of questions that were asked in surveys of experiment participants after the three studies. Blue indicates positive responses to the questions (Agree, and Strongly Agree); Red indicates negative responses (Disagree and Strongly Disagree); Gray is Neutral. The horizontal scale represents the percentage of respondents. Note that slightly different experimental setups were used for each experiment, so not all questions were asked to all participants; counts are shown on the right hand side. The red responses were particularly helpful for improving the task-tree and dashboard interfaces. ©IEEE 2016.
The students used our platform to record the URLs associated with each of the stages in this analysis workflow; since the workflow was almost exclusively web-based, no application or file-based instrumentation was included. Participation metrics from this study can be seen in Table 5.3 (column for Study B). This time around, all students actively participated. Over a two-month period, each one tagged an average of 410 URLs. This compares to 77 for the CS class, representing over 5 times more active engagement with the Journaling interface. We attribute this increase to the three new ‘prompting options’ (described in section 5.5) that were made available to students for this study. By more carefully deciding when to prompt (i.e. using dwell time on page and proportion of page scrolled), this seemed to dramatically reduce the chances that users got frustrated and switched off the tooling. In addition, the trigger based on scrolling 50% of the way down a web page (in addition to a 70s dwell time) increased the average number of gathered events per user by over 25% (from 444 to 563 events per user). For this study, we also used a simplified task-tree (with only 3 levels and 20 possible labels), which both sped up the ‘nanoblogging’ task and increased the collaboration indices (average number of users contributing to each goal).

Given this data, we could then compare analytic approaches between groups of students. In particular, some deficiencies in approaches (e.g. lack of systematic planning in stage 2) showed up strongly in the data. In addition, we were able to estimate the amount of collaborative activity occurring within each team throughout the course of the trial. An example of the data collected is shown in Figure 5.5, in which we computed a time-windowed version of our collaboration index, defined as the number of team members working on the same task in successive 2-hour windows. The instructor was able to use this data to monitor team progress throughout the study. We are now evaluating the final reports and seeking to establish correlations between the quality of reports, the analytic workflows that were used to create them, and the collaboration patterns used by each team.

5.7.4 Intelligence Analysts

The third evaluation study we conducted consisted of a group of seven intelligence analysts, working in collaboration with researchers at our university to study two present-day security topics. They used a similar task-tree as the political science class but with an additional level of sub-tasks (making a total of 52 possible tasks, rather than 20).

For this deployment of the prototype, we systematically captured and assessed different types of information artifacts used by the professional analysts at each stage of the analysis process, including URLs, documents and applications. This was to provide a comparison with the political science students and also to create a more comprehensive dataset for testing task-centric document curation algorithms.

Results from the trial can be seen in Table 5.3 (column for Study C). Each user submitted an average of 630 labelled events over two months (a further 54% increase over Study B), covering 33 goals.
Figure 5.5 Collaboration patterns for three teams of students in the Political Science class. The x-axis is time, measured in hours; the y-axis is a 'collaboration index', defined as synchronous activity (on the same task) within 2-hour windows. The mean collaboration index, $\mu$, varied considerably between teams. This is an example of how the frequency of collaborative activity and number of participating collaborators could be continuously estimated throughout the study. ©IEEE 2016.

We repeated the evaluation survey with this group ($n=7$), and added the results to those in Figure 5.4. This group had a lower tolerance to working with Beta software than the students, which lead to more criticism of aspects that did not integrate well. Overall though, most agreed that the interfaces were easy to navigate and had clear purpose. A few users had specific issues with versions of Chrome that we are working to resolve. All users were keen to continue using the prototype after the trial period.

### 5.8 Discussion and Future Work

The data from our three evaluation studies in section 5.7 demonstrated that our interfaces are generally considered to be suitable for continuous use over extended time periods as-is, and that sufficient benefits are provided to users (e.g. through provision of collaboration opportunities) to offset the disruption caused. However, we wanted to refine the Journaling interfaces to further reduce disruption to users - hence the focus on algorithms such as that described in Chapter 2, which automatically assign task labels to information artifacts based on analysis of the user's instrumentation data and,
crucially, that from their collaborators.

However, as research in this field develops, it would be possible to provide new services to collaborating knowledge workers that build on the benefits derived from our journaling interfaces. These include task-driven recommendations and task-centric search capabilities, where the results are tailored to the subject matter of the user’s work. These services could be delivered through our ‘recent-work dashboard’ interface. Another natural extension to this work, as identified in [Ack13], is to move beyond knowledge sharing to ‘expertise-sharing’, where entire workflows are shared between collaborating users.

In summary, we have demonstrated effective journaling interfaces and an instrumentation platform to systematically capture the goals and tasks that teams of knowledge workers are striving to achieve, and to associate them with specific activities and information artifacts on an ongoing basis. Through studies with two classes of students and with intelligence analysts, we have demonstrated that the interfaces have a footprint that is considered acceptable for continuous use in a work environment. Through our ‘recent-work dashboard’ that is continuously updated with the gathered data, the tools are both facilitating collaboration and allowing the extent of collaboration to be measured in a task-centric manner. Furthermore, the large amount of data being gathered by our platform is opening a new field of research in our group that is enabling the creation of task-centric machine learning algorithms that will be essential to next-generation smart digital assistants.

5.9 Human Factor Considerations

Following the three trials described in this chapter, a series of interviews were conducted\(^1\) with analysts at the LAS, in order to gather their thoughts on the perceived utility of the Journaling Interfaces and Instrumentation Platform, as well as their concerns and ideas for future directions. A brief summary of that work is included here in order to further inform future research directions in this field. This complements another recent study on the sociological implications of self-tracking technologies [Lup14], which provides a more general study of the field, but without particular reference to knowledge workers.

Firstly, in terms of perceived benefits, it transpired that most of the benefits communicated by the interview participants were related to the logging functions of the application more so than its potential coupling with recommender systems in the future. The benefits described by these users could be broken down into the following categories:

- **Source management** - many users wanted to keep track of recently used documents, which formed the motivation for our ‘recent-work dashboard’ prototype. Further work is required in order to quantify the time savings from this prototype.

\(^1\)these interviews were expertly conducted by Gwendolynne Reid and Chris Kampe in August 2016.
- **Collaboration** - users wanted to gain insights from team members, as well as to share documents and to assist with training new staff. On the whole, collaboration was seen as a primary benefit of the Journaling Application in analytic work.

- **Cognition** - this refers to the application's use as a memory aid, particularly when restarting tasks in a multi-tasking environment, but also for 'synthesis' (adding 'memory-jogging' tags for future use). Some people also thought the system would be useful for managing attention, and helping people to keep focussed on what they're supposed to be doing. Finally, there were perceived benefits for self-reflection, particularly in two specific circumstances: (1) where timesheets or task billing is required, (2) where someone has a particular issue they're looking to gather evidence for. However, as we found early in this work, occasional self-reflection is not enough to keep motivating people to use the system - continuous, just-in-time, and pro-active benefits are preferable.

- **Activity Documentation** - users wanted automatic generation of evidence for performance reports and for capturing workflows, as well as an ability to ‘mine’ for behavioral anomalies and correlations. They also wanted to understanding actual resources that went into a task (for estimation), and to produce summarizes for management. The system could also be applied to the 'Build, Measure, Learn' cycle of innovative research [Rie11] - i.e. use to measure build resources in the 'Build' process, benefits/utility of tools in the 'Measure' stage, and as a reflective tool for the 'Learn' stage.

- **Resource allocation** - users wanted to use the system in conjunction with predictive analytics to determine when to assign more resources to a system or project.

- **Training** - as a way to capture workflows and best practices for passing onto new staff (thereby enhancing ‘institutional memory’).

- **Archives/history** - finally, users wanted to use the tool to maintain a record (or ‘provenance’) for work products and decisions that were made.

The concerns about the system and associated research directions that were expressed by the LAS staff during interviews could be broken down as follows:

- **Classification/privacy** - analysts expressed concern about the Journaling application inadvertently allowing individuals access to documents or parts of documents that they do not have the clearance to view. Also any provenance for deleted data also needs deleting. Overall, analysts were much less concerned about their own privacy as individuals, with most analysts pointing out that they had given up a certain right to privacy when taking their job, and were more concerned about being in compliance with laws and regulations pertaining to the privacy of the persons whose data they might access as part of their work.
• **Cognition** - users mentioned the potentially damaging cognitive effect of interruptions, but another interesting concern was over the potentially adverse encouragement of ‘groupthink’, where bad practices could spread instead of best practices - people could become lazy in their thinking.

• **Timing** - concerns were expressed over interruptions at bad times. Users requested a ‘surge’ mode, in which no prompting would occur. They also wanted the system to learn when people are triaging their work and are therefore potentially more interruptible. Some analysts wanted to work with an idea privately and then gradually expose the idea to team members as it became more mature, and expressed a need for control over data sharing.

• **Collaboration** - sometimes people didn't want to collaborate, often due to conflicted goals, which were typically mission vs personal; personal goals (usually a desire for personal advancement and recognition) often created conditions more conducive to competition and territoriality, rather than collaboration. Also the potential exists for data being used by managers against their employees.

• **Analytic mistakes** - one potential danger of increased collaboration is in sharing ‘half-baked’ documents that could lead to false assumptions and conclusions. A need was expressed for appropriate caveating of shared documents.

• **Managerial mistakes** - much like the concerns heard about Journaling data being misinterpreted and leading to mistaken conclusions (i.e. if taken out of context), analysts expressed concerns about their activity data leading to mistaken managerial conclusions and decisions. Either accidental or deliberate mis-use, mis-interpretation or abuse of the data. The system doesn't currently capture the difficulty of the task, or the context/environmental factors at the time.

Finally, the following additional features or capabilities were described, and could be used to inspire possible future directions for this research:

• **Controlling disruption** - users wanted a variety of new controls over prompting, particular to allow for ‘surge’ and ‘draft’ (or ‘private’) modes, as well as prompting based on the user’s affective state. Users also wanted certain families of sites to always be blacklisted (e.g. HR/personal docs). Detecting the user’s affective state, and also certain types of privacy-critical documents are interesting avenues for future research.

• **Organization** - users wanted an ability to quickly resume tasks, as provided by the dashboard, but also expressed a need for alternatives to the simple task-tree used here, which may not be sufficient for all tasks. Some users want a less structured approach (i.e. just a collection of ‘tags’ or ‘bins’).
• **Exploration** - users were drawn to the idea of sharing their resources and being able to see resources tagged by their colleagues. Suggested additional features for tagging were: a field-dependent credibility score, including users & dates (i.e. for attributability), and automatic recommendation of associated tags (e.g. from co-occurrence). Users wanted to distinguish between qualitative (e.g. good, reliable) and subject-matter (topic-based) tags.

• **Reflection** - several ideas for visualizations were communicated, both for summarizing types of activities (writing reports, email) and the projects that activities relate to.

• **Control/sharing** - users wanted levels for personal, team, organization / community, and maybe public sharing, and to maintain direct control of these. Concerns were expressed over management abuse of the data (and also concerns of ‘non-reciprocating’ sharers).

• **Other ideas** - auto-summary generation for tasks, ‘gamification’ (rewarding sharers without punishing others), graphical tags (particularly to denote trustworthiness of documents), and an ability to provide ‘status-updates’ for tasks before suspending them (in the form of free-form text or voice messages!).

These desired features were analyzed according to the likely level of labor required to realize them, and according to the priority level suggested by the students and analysts. We found that some relatively simple features, such as better prompting and sharing controls, were also the highest priority. This is good news as it suggests that several worthwhile additional benefits may be realizable relatively easily. It is also clear that there is plenty of scope for the development of new algorithms to assist users, particularly for better activity summarization, and for improved visualization of self-tracking data.
This dissertation has presented several instrumentation agents for self-tracking of knowledge workers, as well as two state-of-the-art examples of analytics that can help to enable more effective knowledge creation. We started by motivating this work from a survey of students and intelligence analysts, eliciting the problems they face when tackling knowledge-intensive tasks. We found that, especially in multi-tasking and collaborative environments, they needed assistance keeping track of information artifacts that were being used to infer (or to create) new knowledge.

To address this need, in Chapter 2, we presented a variety of approaches to the challenge of curating documents according to the tasks they correspond to, and doing this in an environment where several team members are working together, and where several tasks may be ongoing at any given time. In order to tackle this in a scalable and generalizable manner, we created a new mathematical framework for modeling analytical workflows based on representations of users, tasks and documents in a common vector space. We showed how to construct such embeddings from a multi-layered graph, and how to optimize classification accuracy by incorporating additional context (such as from document topics, team characteristics, and from temporal information). We also showed how our algorithm can operate on continuous event streams, and how it can be used to drive real-world user interfaces, such as our ‘recent-work dashboard’. Finally, we showed how users could configure the algorithm according to their preferences (for instance, to trade off precision and recall),
and how feedback could be obtained from them to help improve algorithm performance over time.

To provide self-tracking data for our analytics, we described several instrumentation agents and a processing platform in Chapter 3, capable of continuously generating event data and screenshots from knowledge worker’s computer desktops in a way that is consensual but minimally disruptive. We described an agent for MacBook computers that can capture holistic information on document usage across all applications running on the desktop and that can provide a rich set of metadata to enable task-centric document curation. We also described a more constrained, but platform-independent, agent for Chrome web browsers that can capture information on documents accessed online. To demonstrate the future potential of such self-tracking data, we then briefly described a number of other use-cases for the passive instrumentation data, which included user-action prediction (for pro-active resource allocation, or for recommender systems), instrumentation for enterprise intelligence, and analysis of online search behaviors. All of these are rich areas for future research.

In Chapter 4, we presented an alternative approach to self-tracking analytics - this time using the desktop screenshots collected by one of our agents. We first showed how simple image analysis techniques could be used to reproduce aspects of agent-based instrumentation, such as application and document detection, before moving on to demonstrate how screenshot analysis can be used to provide finer-grained insight into information artifacts. Through a controlled analysis study that generated over 120k screenshots, we were able to show how it is possible to systematically and accurately extract ‘claims’ being made in written reports, and to associate these with ‘evidence’ from web browsing. These artifacts could be used as the basis for new automated provenance generation capabilities in a subset of knowledge creation tasks, particularly those where the user is searching for information on a subject they aren’t familiar with, and under time pressure. Such provenance information could potentially be used to provide an ‘evidence trail’ for written reports, both to increase accountability/credibility of reports and also for educational benefit. This work could be taken much further - one exciting direction would be to build on current research in the field of veracity [Li15], perhaps by creating algorithms to automatically ‘fact-check’ evidence statements, or by attempting to find other documents that either corroborate or contradict a particular piece of evidence. These additional documents could make good material for recommendations to knowledge workers that might be used to help them overcome a variety of cognitive biases, particularly confirmation bias.

Finally, in Chapter 5, we described a set of ‘Journaling’ interfaces that could be used by knowledge workers to associate task labels with information artifacts and with instrumentation data. These labels were needed to enable the training of supervised machine learning algorithms, such as those presented in Chapter 2. Using our interfaces, we gathered labelled data from three real-world user studies (with students and intelligence analysts), each conducted over a period of at least two months. We researched ways to minimize disruption to users from the need to provide labels, such as through smart triggering mechanisms for the interfaces. We also demonstrated capabilities to provide imme-
mediate benefits to them that helped offset the impact of any disruption, such as by enabling better collaboration and by assisting with information recall. Much more research could be conducted in this space to further reduce the impact of such label requests and to increase the benefits from these kinds of systems. One promising direction is in the area of ‘active learning’ algorithms that could be used to carefully limit feedback requests to only those that could maximally benefit the underlying algorithmic models.

After the conclusion of our three user trials, we conducted a series of user interviews to gather feedback on perceived benefits of our self-tracking technology, as well as concerns relating to it. This provided a rich set of future research directions, both to mitigate concerns and to maximize benefits. We also gathered feedback on new capabilities that could be added to our prototypes and found that some simple features, such as a better ability to control prompting, and new sharing features, are likely to provide the most benefit initially. Longer-term, new techniques to infer the affective state of users (perhaps using wearable sensors) could be used to create a new generation of more ‘emotionally intelligent’ algorithms that can better respond to the user’s current information needs (as well as to their current mood or stress levels).

The benefits, concerns and ideas that we elicited from our user studies corroborate well with a recent analysis of the wider field of self-tracking [GD16]. In that analysis, the authors point out that, although self-tracking can sometimes create a perceived burden on users to respond to their data in some way, they have seen many examples of people using their data simply to ‘quiet their worries’. In a similar way, we think that, in many cases, exposing knowledge workers to their self-tracking data will help them feel less anxious and more productive, not the reverse.

All this provides reason to be optimistic that there is an exciting future ahead for the field of self-tracking for knowledge workers. However, there are still many more research questions to be addressed before use of these technologies is likely to become generally accepted, and before we can begin to unlock the full potential of instrumentation-enabled human-machine collaborative intelligence in the knowledge-creation workplace.
BIBLIOGRAPHY


[Bra] Bradski, G. Dr. Dobb’s Journal of Software Tools ()


