

Automatic Generation of Issue Maps: Structured, Interactive Outputs for Complex Information Needs

Dafna Shahaf
CMU-CS-12-140

September 2012

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Carlos Guestrin, Chair
Manuel Blum
Christos Faloutsos
Eric Horvitz, Microsoft Research
Jon Kleinberg, Cornell

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy*

Copyright © 2012 Dafna Shahaf

This research was sponsored by the National Science Foundation under grant number IIS-0644225, the Department of Defense/Army Research Office under grant number W911NF0710287, the Office of Naval Research under grant number N000014-07-1-0747, the Army Research Office under grant numbers DAAD190210389 and W91F0810242, Microsoft Research Graduate Women's Scholarship and Microsoft Research PhD Fellowship.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Data mining, Mapping information, Structure of information, Coherence, Mapping science, Mapping news

For David.

Abstract

When information is abundant, it becomes increasingly difficult to fit nuggets of knowledge into a single coherent picture. Complex stories spaghetti into branches, side stories, and intertwining narratives; search engines, our most popular navigational tools, are limited in their capacity to explore such complex stories.

We propose a methodology for creating structured summaries of information, which we call *metro maps*. Our proposed algorithm generates a concise structured set of documents that maximizes coverage of salient pieces of information. Most importantly, metro maps explicitly show the relations among retrieved pieces in a way that captures story development.

The overarching theme of this work is formalizing characteristics of good maps, and providing efficient algorithms (with theoretical guarantees) to optimize them. Moreover, as information needs vary from person to person, we integrate user interaction into our framework, allowing users to alter the maps to better reflect their interests. Pilot user studies with real-world datasets demonstrate that the method is able to produce maps which help users acquire knowledge efficiently. We believe that metro maps could be powerful tools for any Web user, scientist, or intelligence analyst trying to process large amounts of data.

Acknowledgments

On my first day in Pittsburgh, I stopped a stranger on the street and asked him where I could get breakfast. By the end of the conversation, I had his business card and a warm invitation to have dinner with his family. This trend seemed to continue throughout my entire time at CMU: everywhere I went, kindness awaited. I would like to take this opportunity to thank those who made my grad school experience memorable.

First and foremost, I'd like to thank my advisor, Carlos Guestrin, for challenging me and encouraging me to find my way in the research landscape; for knowing when to push and when to back off (and when to bring white chocolate); for insisting on sitting through practice talks and reading drafts, even when things were hectic (Once more, with feeling: Overview slides are the enemy! This paper needs more oomph!).

They say that a single conversation with a wise man is better than ten years of study; from a quick back-of-the-envelope calculation, I figure that I owe my committee several thousands of years. Many thanks go to Manuel Blum, for conversations about anything and everything, from astrophysics to knights in medieval times – all of them inspiring; To Eric Horvitz, for some of the best times I've had during my PhD, and for always looking after me; To Christos Faloutsos, for ever-interesting pointers (also, apologies – uplifting news are hard to find); To Jon Kleinberg, for always finding new ways to look at a problem.

I've had enjoyable and rewarding discussions with many other professors during my PhD. I'd especially like to thank Geoff Gordon (for knowing everything), Anupam Gupta (for being my local theory oracle), Yossi Azar (for being my external theory oracle), and Niki Kittur (for putting my research in a broader context). I could always count on Nachum Dershowitz, Gal Kaminka, Aryeh Kontorovich and Oren Kurland for words of wisdom.

In addition, a big thanks to Michelle Martin, Deb Cavlovich, and the unsung heroes of the department offices and the office of international education, who made my graduate school experience smooth, even (and especially) when bureaucracy was against us.

I have learned a lot from being a member of the SELECT lab; every presentation with them is like a mini-defense. Thanks to Danny Bickson, Byron Boots, Joseph Bradley, Lucia Castellanos, Anton Chechetka, Miroslav Dudík, Khalid El-Arini, Stanislav Funiak, Joseph Gonzalez, Arthur Gretton, Sue Ann Hong, Jon Huang, Andreas Krause, Aapo Kyrola, Yucheng Low, Austin McDonald, Sajid Siddiqi, Ajit Singh, Le Song, Gaurav Veda, Yisong Yue, Brian Ziebart, and summer intern Doris Xin. Special thanks to Yisong (for screaming at the taxi driver and for a very enjoyable concert), Khalid (ditiě! No pay!) and Gaurav (for stopping political debates before they turned violent).

My friends deserve special thanks for being there when needed and forgiving my absence at (many) other times. I am thankful to Eyal Ron (I knew you'd fly over) and Tal and Deb Galili (for forgiving my minimal online presence). I'd like to thank Kate Taralova, for the most positive attitude I have ever seen (even when the car keys fell into the river during rafting), and for being a great friend (with a perfect sense of timing); Ali Kemal Sinop, for spring carnivals; Brendan Meedar, for unfinished movies; Or Sheffet, for crosswords, earworms (Gaaaaadi!), dreams about stakes (and some stranger dreams), and putting up with me as an officemate; Guy Zinman, for snails; Greg Kesden, for play-doh perfume and seahorse surgeries; Tal and Sagi Perel, Sharon and Ziv Baum (and Leo), Uriel Haran, Tzur and Galit Frenkel, Yair and Dava Movshovitz-

Attias, Ariel and Vera Procaccia (and Charlie), Harel and Bimla Schwarz (and Lucy), Rani and Julia Bear, for being my community here and for countless mangals (no, this does not translate to 'barbecue').

I'd like to thank Eyal and Anat Lubetzky, Suchi Saria and Aria Haghighi for a great summer, from looking for Japanese karaoke clubs in Seattle to coloring light bulbs at 4am. Thanks goes to Bill Harris for a completely-different yet equally-awesome summer, full of 'splosions, Cheerwine, and Wisconsin stories.

My final thanks are reserved for my family, who have been a continual source of support. Special thanks to my parents, for backing me up and encouraging through the years of research (and leaving some travel agents very confused in the process); To David's parents, for a strategically-planned dive into the orange-juice aisle. And of course, to David, for making rummikub sets with me against all odds; for a trip to Cedar Point by a roller-coaster hater; for Pogo dwarfs; and for so much more than I can fit in this page (or the entire thesis, for that matter).

Thank you.

Contents

- 1 Introduction 1**
 - 1.1 Why Maps? 3
 - 1.2 Thesis Statement and Our Approach 5
 - 1.2.1 Approach Overview 6

- 2 Connecting the Dots: Constructing a Single Line 13**
 - 2.1 What makes a story good? 14
 - 2.1.1 Formalizing story coherence 15
 - 2.2 Finding a Good Chain 20
 - 2.2.1 The Algorithm 21
 - 2.2.2 The effect of $kTotal, kTrans$ 24
 - 2.3 Interaction 25
 - 2.4 Summary 26

- 3 Constructing a Map 27**
 - 3.1 Properties of a Good Map 28
 - 3.1.1 Coherence 29
 - 3.1.2 Coverage 29
 - 3.1.3 Connectivity 31
 - 3.1.4 Objective function: Tying it all together 31
 - 3.2 Finding a Good Map 33
 - 3.2.1 Representing all coherent chains 33
 - 3.2.2 Finding a high-coverage map 35
 - 3.2.3 Increasing connectivity 36
 - 3.3 Interaction 38
 - 3.3.1 Interaction with a Map 38
 - 3.3.2 Interaction with a Single Chain 39
 - 3.4 Summary 41

- 4 Case Study 1: News 43**
 - 4.1 The Need for Maps of News 44
 - 4.2 Objective for the News Domain 44
 - 4.2.1 Coherence: Measuring influence without links 44
 - 4.2.2 Coverage 47

4.2.3	Connectivity	48
4.2.4	Examples of Maps and Chains	49
4.3	User study: News	50
4.3.1	Connect the Dots: User Study	51
4.3.2	Metro Maps: User Study	55
4.3.3	Interaction	58
4.4	Summary	62
5	Case Study 2: Science	63
5.1	The Need for Maps of Science	64
5.2	Objective for Scientific Papers	64
5.2.1	Coherence for Scientific Papers	64
5.2.2	Coverage	66
5.2.3	Connectivity	71
5.2.4	Example Maps	72
5.3	User study: Science	72
5.3.1	Results and Discussion	74
5.3.2	User Comments	75
5.4	Summary	75
6	Implementation and Analysis	79
6.1	Algorithm Overview	80
6.2	Speeding Up the Computation	81
6.2.1	Complexity of Map Construction	82
6.2.2	Scaling Up the Coherence Graph	82
6.2.3	Scaling Up Coverage and Connectivity	87
6.3	Running Times	87
7	Related Work	89
7.1	Visual Metaphors	91
7.2	Connecting the Dots	92
7.3	Summarization and News	94
7.4	Notions of Coverage	95
7.5	Mapping Science	96
7.6	Summary	98
8	Open Questions and Criticism	99
8.1	Scaling Up	100
8.2	Query Mechanisms	101
8.3	Navigation and Personalization	102
8.4	Quantifying Surprise	103
8.5	Learning an Objective	104
8.6	Further Directions	104

9	Conclusion and Discussion	107
9.1	Connecting the Dots	108
9.2	Metro Maps	109
9.3	Future Directions	110
9.4	Usage of Maps	111
9.5	Epilogue	114

List of Figures

1.1	Incomprehensible text in a map form.	3
1.2	Left: A concept map explaining why we have seasons (adapted from Novak and Cañas [2006]). Right: a mind map about personal health (adapted from Jane Genovese). . . .	4
1.3	Issue map: “Can Computers Think?”. Right: a detail from the map. The argument “Computers can’t have free will” is disputed by “Humans also lack free will” (which is in turn supported by two more arguments). Adapted from http://www.macrovu.com/CCTGeneralInfo.html	5
1.4	Two examples of stories connecting the same endpoints. Left: chain created by considering local interactions. Right: a more coherent chain. Activation patterns for each chain are shown at the bottom; the bars indicate appearance of words in the article above them.	7
1.5	A fragment of the coherence graph \mathbb{G} for $m = 3$. Note overlap between vertices.	9
1.6	An example of our results (condensed to fit space). This map was computed for the query ‘Gree* debt’. The main storylines discuss the austerity plans, the riots, and the role of Germany and the IMF in the crisis.	10
2.1	Two examples of stories connecting the same endpoints. Left: chain created by shortest-path (dashed lines indicate similarities between consecutive articles). Right: a more coherent chain. Activation patterns for each chain are shown at the bottom; the bars indicate appearance of words in the article above them.	15
2.2	Word influence from an article about the OJ Simpson trial to two other documents, one about football and another about DNA evidence.	17
2.3	An illustration of the results of the linear program, showing initialization and activation levels along a chain for three words. Activation level is the height of the bars. Initialization level is the difference in activation levels between two consecutive transactions, if the activation level has increased.	18
2.4	Scoring a chain.	19
2.5	Activation patterns found by our algorithm for a chain connecting 9/11 to Daniel Pearl’s murder. (a): Activation levels. (b): Activation levels weighted by the influence (rescaled). For illustrative purposes, we show the result of the integer program (IP) we get replacing constraint (7) of the LP by its binary equivalent.	20
2.6	Activation levels for the best $s-t$ chain for different values of $kTotal$ and $kTrans$	25
2.7	Chain refinement. The starred article was added in order to strengthen the last link. . . .	26
3.1	Greek debt crisis: a simplified metro map. Metro lines are coherent storylines (stops correspond to articles).	28

3.2	Encoding chains as a graph: each vertex of the graph corresponds to a short chain. A path in the graph corresponds to the concatenated chain.	33
3.3	A fragment of the coherence graph \mathbb{G} for $m = 3$. Note overlap between vertices.	34
3.4	An example of our results (condensed to fit space). This map was computed for the query ‘Gree* debt’. The main storylines discuss the austerity plans, the riots, and the role of Germany and the IMF in the crisis.	37
3.5	Tag cloud of documents about Guantanamo Bay detention camp. Word size is proportional to its frequency.	41
4.1	A bipartite graph used to calculate influence.	45
4.2	A map about the legal process leading to Clinton’s acquittal.	49
4.3	A map about the racial aspects of OJ Simpson’s trial.	49
4.4	Map about the earthquake in Haiti, before interaction.	50
4.5	Example output chains for Connect-Dots and Google News Timeline. Users were given access to the full articles. The GNT chain is a lot less coherent, and includes several insignificant articles, e.g., an article about a domain name that once belonged to bin Laden’s family.	51
4.6	Top: Evaluating effectiveness. The average (over users) of the fraction of familiarity gap which was closed after reading a chain. Numbers at the bottom indicate the average familiarity with each story (on a scale of 1 to 5) before reading any chain. Bottom: Relevance, coherence, and non-redundancy (broken down by simple vs. complex stories). The y axis is the fraction of times each method was preferred, compared to another chain. Users could mark chains as ‘equally good’, and therefore the numbers do not sum to 1. Our algorithm outperformed the competitors almost everywhere, especially for complex stories.	54
4.7	User study results: average number of correct answers amongst users vs. time. As the task gets more complex, metro maps become more useful.	57
4.8	Tag clouds representing descriptions of Google News (left) and Map (right) paragraphs. Note maps receive more positive adjectives.	59
4.9	Map about the earthquake in Haiti, after decreasing the importance of the word ‘abduct’.	61
4.10	Map about the earthquake in Haiti, after increasing the importance of ‘aid’ and ‘rescue’.	62
5.1	Direct (left) vs. ancestral influence (right).	65
5.2	Tag clouds for p_1 and p_2 . The size of a word is proportional to its frequency. (a-b) p_1 and p_2 ’s content, respectively. (c-d) Venues and authors of papers affected by p_1 and p_2 , respectively. Note that (a) and (b) are very similar, but (c) and (d) are not.	66
5.3	A simple citation graph. Edges traverse in the direction of impact, from cited to citing paper. (a) Coverage of document A. Gradient indicates different degrees of coverage. (b-c) The effect of adding papers B and C (respectively) to paper A. Since B’s descendants are already covered to some extent by A, we prefer C.	68
5.4	Two branches in the citation graph. The left branch is coherent; the right one is not.	69
5.5	Coherence graph. Nodes represent papers (names appear inside). Paths represent coherent chains. Each paper may have multiple corresponding vertices: the highlighted vertices are all copies of paper p	69

5.6	Two coherent chains (theory of SVMs, application of SVM to vision). The chains do not intersect, yet are related: the application chain uses tools from the theory chain. Dashed gray lines indicate impact.	72
5.7	Part of the map computed for the query ‘Reinforcement Learning’. The map depicts multiple lines of research (see legend at the bottom). Interactions between the lines are depicted as dashed gray lines, and relevant citation text appears near them.	77
5.8	A segment of a map computed for the query SVM/ Support vector machine, showing the intersection of two lines: multi-class SVMs and large-scale SVM. In the interest of space, we condensed the time line.	78
6.1	Behaviour of the top-9 features for Chain A (left), and A’ (see in text, right). x axis represents document position in the chain, y axis represents PCA feature value. Circles correspond to the actual feature values in the chain, and dashed lines are the lowest-degree polynomial which fits these points within a specified error. The degree of each polynomial appears above it.	84
7.1	Related Work. The light-blue line details the evolution of this work. The red line revolves around visual representations for science, while the pink line focuses on map representations. The green and orange lines concentrate on methods for summarizing collections of documents, especially in the news domain. Variations of “Connecting the Dots” problem appear along the yellow line. Finally, notions of coverage are covered by the purple line.	90
7.2	Helping intelligence analysts make connections via chains of cliques [Hossain et al., 2011].	92
7.3	A connection graph between Alan Turing and Sharon Stone [Faloutsos et al., 2004].	93
7.4	Newsjunkie story interface [Gabrilovich et al., 2004].	95
7.5	Mapping Chemistry [Boyack et al., 2009]. Map of the 14 disciplines, fractions of papers by field for each discipline, and knowledge flows between disciplines for 1974.	96
7.6	Mapping Chemistry [Boyack et al., 2009]. Each node is a cluster of journals, and is sized to show numbers of papers in the journal cluster.	97
7.7	Mapping Geography abstracts to visualize the domain [Skupin, 2004].	97
7.8	iOpener Workbench (Action Science Explorer) tool for summarizing research domains [Chen, 2004; Dunne et al., 2010].	98
8.1	An overview of the metro map system. The algorithm (middle) receives data (top left) and a query (top right) and computes a map.	100
8.2	MemeTracker Leskovec et al. [2009]: Top 50 threads in the news with highest volume for the period Aug. 1 Oct. 31, 2008. Each thread consists of all news articles and blog posts containing a textual variant of a particular quoted phrase. (Phrase variants for the two largest threads in each week are shown as labels pointing to the corresponding thread.) The data is drawn as a stacked plot in which the thickness of the strand corresponding to each thread indicates its volume over time.	101

9.1	Bloom’s cognitive categories [Bloom et al., 1956], lowest order processes to the highest: Knowledge (Remember), Comprehension, Application, Analysis, Synthesis (Create), Evaluation.	112
9.2	Our proposed information-needs taxonomy of map users.	113
9.3	Use Case (Illustration). Top: Adding a map sidebar to a news site allows a user to understand the broader context of the article, or navigate to related articles. Bottom: Adding a Connect-the-Dots sidebar to a news site allows the user to connect the article they are reading (top right) and other articles.	115

List of Tables

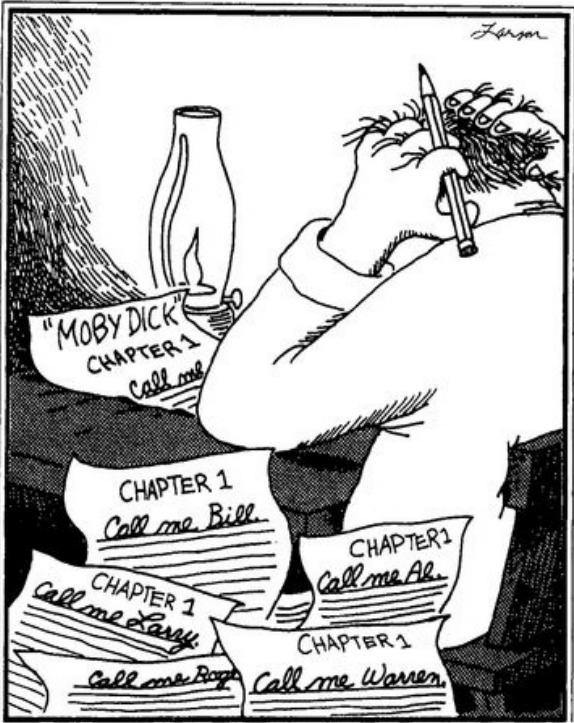
- 4.1 Top 10 influential words for different values of ϵ 47
- 4.2 subtopic recall (fraction of the important events that are successfully retrieved) per map size. 56
- 4.3 Ease of navigation: number of articles that users visited per correct answer. 58

- 5.1 Information collection patterns. 74
- 5.2 Precision scores: paper score and label score. 74

- 6.1 Running times for MetroMaps. 87

Chapter 1

Introduction



“Distringit librorum multitudo” (the abundance of books is a distraction), said Lucius Annaeus Seneca; he lived in the first century.

A lot has changed since the first century, but Seneca’s problem has only become worse. The surge of the Web brought down the barriers of distribution, and users find themselves overwhelmed by the increasing amounts of data; relevant information is often buried in an avalanche of data, and locating it is difficult. The problem spans entire sectors, from intelligence analysts trying to discover pertinent and actionable information, to scientists and web users.

Extracting useful knowledge from large data sets requires appropriate means. In recent years, search engines have been relied upon for accessing information, and investments have even been made to create specialized search and retrieval tools (e.g., academic search and news search). However, the search and browsing experience might be best characterized as providing keyhole views onto the data: while search engines are highly effective in retrieving nuggets of knowledge, the task of fitting those nuggets into a coherent picture remains difficult.

We identify three main issues with current, keyword-based search engines:

- * **Expressing information needs** Often, users know precisely what they want to find, but it is not easy for them to distill their ideas down into a few *keywords*. For this reason, building models of users’ information needs from keywords alone is difficult.
- * **Structured output** The output of most search engines today consists of a list of relevant documents. Yet, richer forms of output can be conceived. Often, visually exploring search engine results can reveal new and interesting phenomena, which are harder to see in a textual list.
- * **Interaction** Interaction with most search engines today is fairly restricted. Users who are not satisfied with the search results are often limited to submitting a new, refined query. Refining queries is a non-trivial task, and users often make several attempts at it before finding the desired result (or giving up). Richer models of interaction can help users better express their information needs. Furthermore, adding an interactive component to our system can allow users to take a full advantage of the structured output (for example, by refining parts of the output, or zooming into components).

Several tools exist today for summarizing complex topics. Text Summarization methods can be classified into extractive and abstractive summarization [Radev et al., 2002]: Extractive summarization methods select important sentences from the original document, while abstractive summarization methods attempt to rephrase the information in the text. Other summarization systems [Gabrilovich et al., 2004; Swan and Jensen, 2000; Yan et al., 2011; Allan et al., 2001] have focused on *timeline* generation.

However, these styles of summarization only work for simple stories, which are linear in nature. In contrast, complex stories exhibit a very non-linear structure: stories spaghetti into branches, side stories, dead ends, and intertwining narratives. To explore these stories, users needs a *map* to guide them through unfamiliar territory.

1.1 Why Maps?

Our interest in map representations stems from a desire to understand the associations and inter-relations within a topic. A map can be an ideal tool for this task. In the real world, cartographic maps have been relied upon for centuries to help us understand our surroundings and the relationships between neighbouring objects.

Most importantly, maps take advantage of the human mind's ability to scan an entire page in a non-linear manner and quickly extract information and patterns. This ability goes unexploited when sifting through paragraphs of text or a linear timeline. Consider, for example, the following block of text:

“ Aliquam fringilla tortor sapien sociosqu mauris per pulvinar ante habitasse euismod elit interdum primis elementum auctor aptent malesuada. ”

This text is meaningless, since we cannot understand the words. However, when we see the same text organized in a map form, we immediately form a mental model of the information:

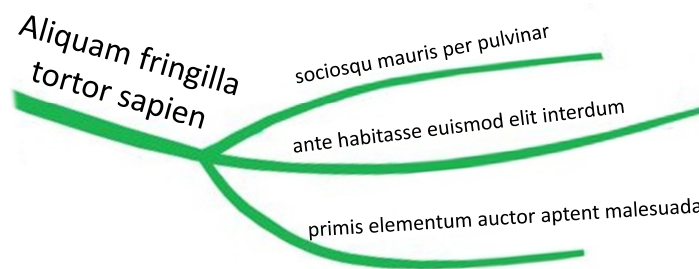


Figure 1.1: Incomprehensible text in a map form.

While still unable to interpret the text, we can see that there is one main idea and three smaller ones. These smaller ideas seem to have similar relationship to their parent, so we may guess that they have some characteristics in common with each other: They could be three properties of the main idea, or perhaps demonstrative examples. Our mental model of this data is primed with expectations for patterns for us to evaluate and expand.

The important point is that **there is information in the lines of a map** – in this case, more than in the words. In a way, the map acts as the body language of its words¹.

Recently, several map-like representations have been gaining popularity: A *concept map* [Novak, 1990] is a graphical representation where nodes represent concepts, and edges represent the relationships between concepts (Figure 1.2, left). *Knowledge maps* [O'Donnell et al., 2002] are similar to concept maps, but restrict the set of edge labels. *Mind maps* [Buzan and Buzan, 1995] are usually radial, with emphasized color and pictures (Figure 1.2, right).

Finally, *Issue maps* (or argument maps) display the structure of an argument, showing inferential connections between propositions and contentions. Figure 1.3 shows an example of an

¹Presentation inspired by Nick Duffill.

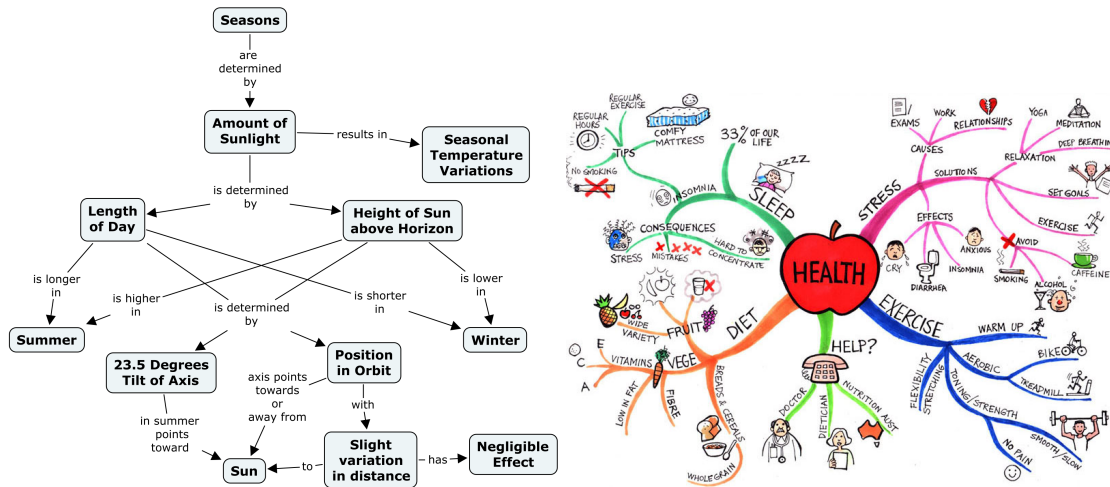


Figure 1.2: Left: A concept map explaining why we have seasons (adapted from Novak and Cañas [2006]). Right: a mind map about personal health (adapted from Jane Genovese).

issue map dealing with perhaps the oldest AI question: “can computers think?” Figure 1.3 (right) focuses on a detail of the map discussing free will.

Importantly, there is strong empirical support for map representations in enhancing, retaining and improving knowledge [Nesbit and Adesope, 2006]. Mind maps were shown to increase memory recall for undergraduate students when compared to traditional note taking [Farrand et al., 2002], and also to reduce cognitive load [Sarker et al., 2008]. Rewey et al. [1991] showed that students recall more central ideas when they learn from a knowledge map than when they learn from (informationally isomorphic) text. The benefits of maps are often higher for students with low prior knowledge.

In addition to recall, the use of knowledge maps enhances students’ subjective reactions to studying and testing. In [Hall and O’Donnell, 1996], map users reported significantly greater motivation and concentration than text users. Similarly, Reynolds et al. [1991] found that students using a hypermap expressed less frustration than those using hypertext.

For all these reasons, we believe that **maps can be a new, exciting way to cope with information overload**. Out of all the map representations discussed above, we consider issue maps to be the most suitable for our needs: issue maps provide a structured, easy way to navigate within a new topic and to grasp the big picture quickly. For example, they can be of extreme usefulness to news readers trying to understand a complex topic, or to researchers who want to understand how their work fits in with previous work and what can be done to advance their field.

Unfortunately, generating good maps is a *manual*, time-consuming process. The map in Figure 1.3 took several man-years to finalize. Furthermore, the maps cannot be easily updated when new data becomes available. In order to overcome these issues, we would like to be able to *automate* the creation of issue maps.

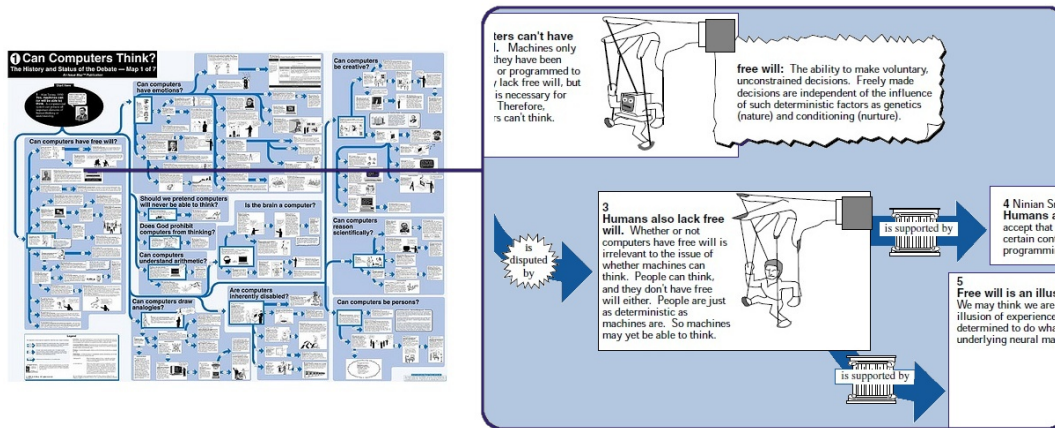


Figure 1.3: Issue map: “Can Computers Think?”. Right: a detail from the map. The argument “Computers can’t have free will” is disputed by “Humans also lack free will” (which is in turn supported by two more arguments). Adapted from <http://www.macrovu.com/CCTGeneralInfo.html>.

1.2 Thesis Statement and Our Approach

This thesis research revolves around the following statement:

“ We can effectively manage information overload by producing personalized issue maps in response to the user’s expressed needs. ”

In the next sections, we propose two systems which serve as stepping stones on the way to automated issue maps. The systems incrementally expand on the set of possibilities determined by **expressing information needs**, **structured output** and **interaction**. The work is performed in two main tasks:

Task 1: Connecting the Dots
Given two articles, our system automatically finds a coherent chain of articles linking them together.

Task 2: Metro Maps
Given a set of documents \mathcal{D} , our system automatically creates a structured summary of \mathcal{D} , which we call a metro map. Metro maps consist of a concise structured set of documents which maximizes coverage of salient pieces of information. Most importantly, metro maps explicitly show the relations among different pieces in a way that captures story development.

The systems are built to handle large amounts of daily-changing data, such as the blogosphere, ACM Digital Library of research papers, and Reuters world news corpus. We believe that the outcome of this work could be a set of powerful tools for any Web user, scientist, or intelligence analyst trying to process large amounts of data.

1.2.1 Approach Overview

We now provide an overview of our approach for automatically generating issue maps. Note that we are only interested in ways to compute the *content* of the map. Visualization techniques, despite their important role, are outside the scope of this work.

Task 1: Connecting the Dots

Maps are complex, and creating one can be daunting. Before tackling an entire map, we focus on a simpler task: a map containing only a single line. To further simplify the task, we start by assuming that the endpoints of the line are known. In other words, we are interested in investigating methods for automatically *connecting the dots*. Given two articles, s and t , our system automatically finds the most coherent chain of articles linking them together. For example, in the news domain, the system can recover the chain of events starting with the decline of home prices (January 2007), and ending with the ongoing health-care debate.

Problem 1.2.1. *Given a set of documents \mathcal{D} , two special documents $s, t \in \mathcal{D}$ and an integer K , find a chain of documents of length $K - 2$ that maximizes $\text{Coherence}(s, d_1, \dots, d_{K-2}, t)$.*

The main challenge of connecting the dots is formalizing a notion of coherence. In this work, we argue that coherence is a *global* property of the chain, and cannot be captured by local interactions between neighbouring articles in the chain.

To demonstrate our point, consider Figure 1.4. The figure shows word occurrence along two chains of articles. Both chains share the same endpoints: Clinton’s alleged affair and the 2000 election Florida recount. The chain on the left is an associative, stream-of-consciousness chain. It touches upon the Microsoft trial, Palestinians, and European markets before returning to Clinton and American politics. Note that each transition, when examined out of context, is reasonable: for example, the first and the second articles are court-related.

In contrast, the chain on the right is more coherent. It tells the story of Clinton’s impeachment and acquittal, the effect on Al Gore’s campaign, and finally the elections and recount.

In order to understand what makes one chain more coherent than the other, we look at word appearances (Figure 1.4, bottom). Bars correspond to the appearance of a word in the articles depicted above them; for example, the word ‘Clinton’ appeared throughout the entire right chain, but only at the beginning and the last two articles on the left.

It is easy to spot the associative flow of the left chain in Figure 1.4. Words appear for very short stretches, often only in two neighbouring articles. Some words appear, disappear and re-appear. Contrast this with the chain on the right, where stretches are longer and transitions between documents are smoother. This observation motivates our definition of coherence:

$$\text{Coherence}(d_1, \dots, d_n) = \max_{\text{activations}} \min_{i=1 \dots n-1} \sum_w \text{Influence}(d_i, d_{i+1} \mid w) \mathbb{1}(w \text{ active in } d_i, d_{i+1}) \quad (*)$$

We formulate coherence as an optimization problem. We look for a small set of words w that capture the essence of the story (*active words*). Possible activations are constrained to imitate

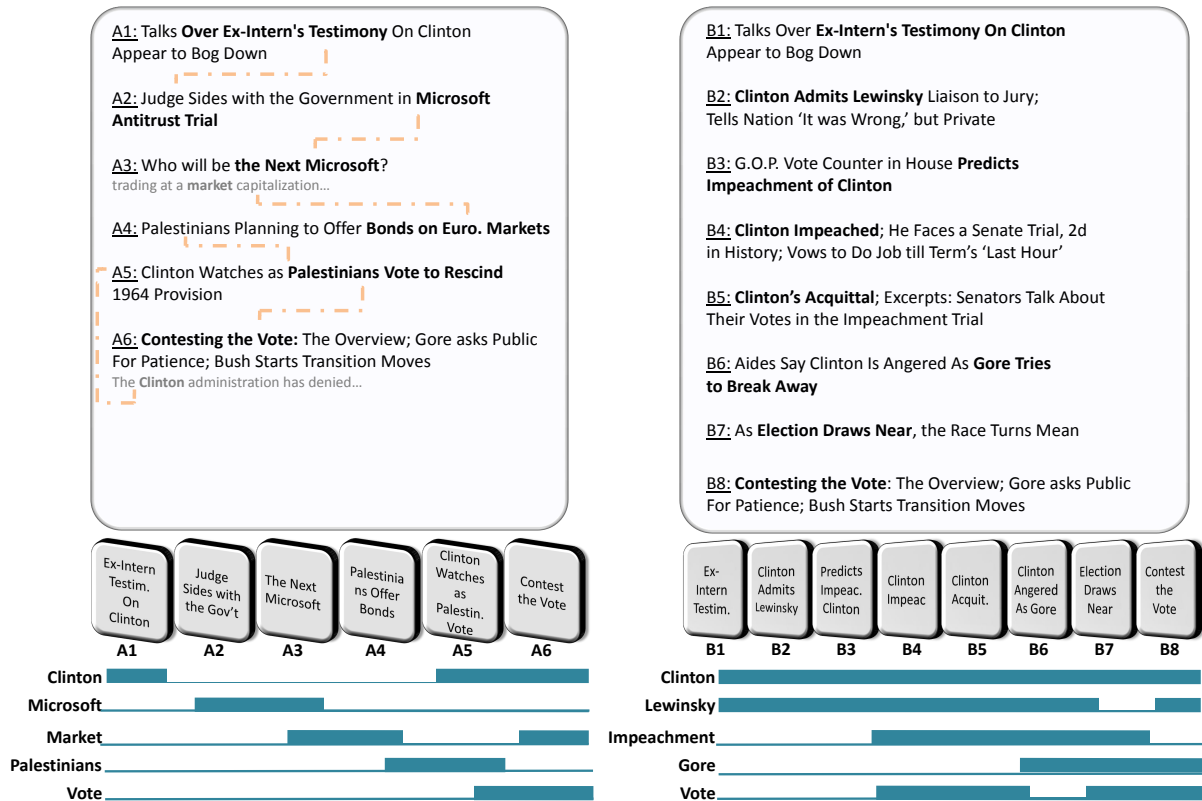


Figure 1.4: Two examples of stories connecting the same endpoints. Left: chain created by considering local interactions. Right: a more coherent chain. Activation patterns for each chain are shown at the bottom; the bars indicate appearance of words in the article above them.

the behaviour of words in the coherent chain of Figure 1.4 (right). The choice of active words determines the score of each transition $d_i \rightarrow d_{i+1}$. The score of the chain is then determined by the score of its weakest link. Refer to Chapter 2 for further details.

We use a Linear Programming solver to find the optimal set of words and evaluate coherence of a chain. We apply a generalized best-first search strategy to find the best chain and solve problem 1.2.1. The general philosophy of generalized best-first search is to use heuristic information to assess the merit latent in every candidate search avenue found during the search. The algorithm continues the exploration along the direction of highest merit.

Task 2: Metro Maps

Now that we know how to construct single lines, we can tackle complete maps. As before, the main challenge lies in formalizing the characteristics of a good map. First, we need to formally define metro maps.

Definition 1.2.2 (Metro Map). A metro map \mathcal{M} is a pair (G, Π) , where $G = (V, E)$ is a directed graph and Π is a set of paths in G . We refer to paths as metro lines. Each $(u, v) \in E$ must belong to at least one metro line.

Vertices of the map V correspond to news articles, and are denoted by $docs(\mathcal{M})$. The lines of Π correspond to aspects of the story. A key requirement is that each line tells a coherent story: Following the articles along a line should give the user a clear understanding of evolution of a story. We re-use our notion of coherence from connecting the dots.

Coherence is crucial for good maps, but it is not sufficient. Coherent lines are not necessarily interesting; in addition to being coherent, lines should also **cover** topics which are important to the user.

To model coverage, we first define a set of elements \mathcal{E} we wish to cover (for example, \mathcal{E} can be words, or topics from a topic model) . We first model the amount a map covers an element e , $cover_{\mathcal{M}}(e)$. We require that $cover_{\mathcal{M}}(e)$ has a *diminishing-returns* property. In other words, if the map already includes documents that cover e well, $cover_{\mathcal{M}}(e)$ should be nearly maximized, and adding another document that covers e well should provide very little extra coverage of e . Therefore, we prefer to pick articles that cover other features, promoting diversity.

Next, we model the amount a map \mathcal{M} covers the corpus as the weighted sum of the amount it covers each element:

$$Cover(\mathcal{M}) = \sum_e \lambda_e cover_{\mathcal{M}}(e)$$

The weights λ_e bias the objective towards maps which cover important features of the corpus. In Chapter 3 we discuss learning a personalized notion of coverage.

After formulating the first two desired properties of a map, we note that a map is more than just a set of lines; there is information in its *structure* as well. Therefore, our last property is **connectivity**. The map’s **connectivity** should convey the underlying structure of the story, and how different aspects of the story interact with each other.

We conducted preliminary experiments exploring different notions of connectivity. These results suggest that the most glaring usability issue arises when maps do not show connections that the user knows about. We came to the conclusion that the connectivity objective needs to be a pairwise function of the metro lines:

$$Conn(M) = \sum_{i < j} Conn(\pi_i, \pi_j)$$

In the simplest case, we could score a point for every two metro lines that intersect.

We now have our three properties. Next, we need to combine them into an objective function. We must consider tradeoffs among these properties: for example, maximizing coherence often results in repetitive, low-coverage chains.

First, we noticed that maximizing coherence does not necessarily lead to good maps. Rather, we treat coherence as a constraint: a chain is either coherent enough to be included in the map, or it is not.

We are left with coverage and connectivity. We believe that *coverage* is the more crucial property out of these two. A map that covers interesting topics but does not show the connections among them can still be useful to the user; a well-connected map about topics that are not important for the user has little value. Therefore, coverage is our primary objective, and connectivity is our secondary.

We can now formulate our problem. Since coverage is our primary objective, we first find κ , the maximal coverage across maps with coherence $\geq \tau$. Then, we look for a map maximizing connectivity among all coherent maps that achieve the maximal coverage κ . One may think of the formulation as coordinate ascent, or lexicographic optimization.

Problem 1.2.3. *Given a set of candidate documents \mathcal{D} , find a map $\mathcal{M} = (G, \Pi)$ over \mathcal{D} which maximizes $\text{Conn}(\mathcal{M})$ s.t. $\text{Coherence}(\mathcal{M}) \geq \tau$ and $\text{Cover}(\mathcal{M}) = \kappa$.*

We also restrict the size of the map to K lines of length at most l .

In order to find a map optimizing this objective, we first list all candidate chains. We compactly represent all coherent chains as a graph, which we call a *coherence graph*. Each vertex of the graph corresponds to a short coherent chain. Edges indicate chains which can be concatenated and still maintain coherence. This property is transitive, so every path in the graph corresponds to a coherent chain.

For example, Figure 1.5 shows a fragment of a coherence graph for $m = 3$. The figure depicts multiple ways to extend the story about the trapped Chilean miners: one can either focus on the rescue, or skip directly to the post-rescue celebrations.

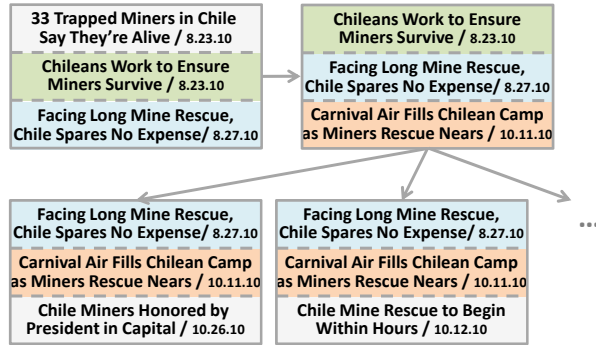


Figure 1.5: A fragment of the coherence graph \mathbb{G} for $m = 3$. Note overlap between vertices.

After constructing the coherence graph, we use it to extract a set of K chains. Since the chains came from the coherence graph, they are guaranteed to be coherent.

The problem of finding a set of chains that maximize coverage is hard. Fortunately, we can exploit the submodularity of our coverage notion. A set function f is submodular if for all $X \subseteq Y, x \notin Y$ we have

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y).$$

Intuitively, submodular functions have *diminishing returns*.

Taking advantage of submodularity enables us to apply well-studied algorithms to the problem. We use submodular orienteering algorithms with approximation guarantees to extract high-coverage chains from the graph. In the last step of the algorithm, we increase connectivity without sacrificing coverage by applying local search in the neighbourhood of the map.

Figure 1.6 displays a sample map generated by the methodology. This map was computed for the query ‘Gree* debt’. The main storylines discuss the austerity plans, the riots, and the role of Germany and the IMF in the crisis.

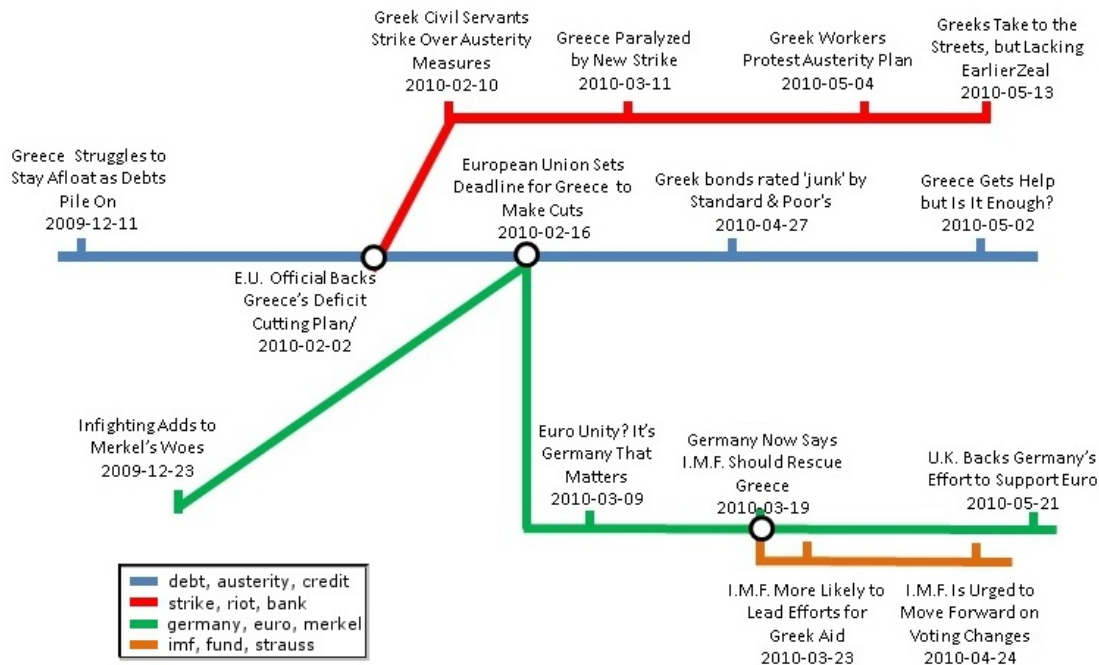


Figure 1.6: An example of our results (condensed to fit space). This map was computed for the query ‘Gree* debt’. The main storylines discuss the austerity plans, the riots, and the role of Germany and the IMF in the crisis.

After devising an algorithm for computing an initial map, we integrated models of interaction with metro maps, letting the users indicate their preferences. We rely on feedback to adjust the coverage function and bias the map towards features that are important to the user.

We apply our map algorithm to two domains: news articles and scientific papers, adapting the map objective to different domain characteristics. For example, in the news domain we have only the article’s content to rely upon; in the scientific domain we can take advantage of the side information provided by the citation graph.

Contributions

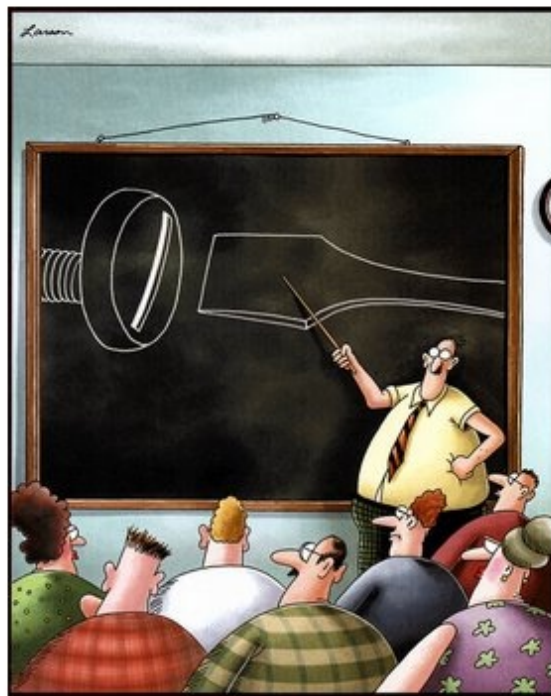
Our main contributions are as follows:

- Formalizing characteristics of a good story and the notion of *coherence* (Chapter 2).
- Providing an efficient algorithm for connecting two fixed endpoints while maximizing chain coherence.
- Proposing the notion of a metro map and formalizing metrics characterizing good metro maps: *coherence*, *coverage* and *connectivity* (Chapter 3).
- Adapting metro maps to the news domain, and formalizing *influence* with no link structure (Chapter 4).
- Adapting metro maps to the scientific domain, taking advantage of the additional structure encoded in the citation graph (Chapter 5):

- Characterizing the probability that ideas in two papers stem from a common source, then using this notion to define coherence of research lines.
- Quantifying the impact of one paper on the corpus.
- Proposing a notion of connectivity that captures how different lines of research can still interact with each other, despite not intersecting.
- Providing efficient methods with theoretical guarantees to compute these metrics and find a diverse set of high-impact, coherent storylines and their interactions (Chapter 3).
- Integrating user preferences into our framework by providing appropriate user-interaction models (Chapters 2,3).
- Performing validation studies with users that highlight the promise of the methodology, as our method outperforms popular competitors. In Chapter 4 we show how our algorithms effectively help users understand the news, especially for stories without a single dominant storyline. In Chapter 5 we show that map users find better papers and cover more important areas than users of competitor systems.

Chapter 2

Connecting the Dots: Constructing a Single Line



School for the mechanically declined

This chapter is based on [Shahaf and Guestrin, 2010] and [Shahaf and Guestrin, 2012].

Maps are complex structures. Before tackling them, we focus on a simpler task: a map containing only a single line. To further simplify the task, we assume the endpoints of the line are known. In other words, we are interested in investigating methods for automatically *connecting the dots*. Given two articles, our system automatically finds a coherent chain linking them together. For example, in the news domain, the system can recover the chain of events starting with the decline of home prices (January 2007), and ending with the ongoing health-care debate.

In this chapter, we formalize the characteristics of a good chain and provide an algorithm to connect two fixed endpoints. In Chapter 4 we evaluate our algorithm over real news data. Our user studies demonstrate the algorithm’s effectiveness in helping users understanding the news. In Sections 2.3 and 3.3 we incorporate user feedback into our framework, allowing the stories to be refined and personalized.

In terms of our three characteristics,

-
- * **Expressing information need** The input is a pair of articles, s and t .
 - * **Structured output** The output is a *chain* of articles connecting s to t .
 - * **Interaction** Users interact with the chain in various ways, including refining it, and tailoring it to their interests by increasing/decreasing the importance of specific words.
-

2.1 What makes a story good?

Our goal is to find a good path between two articles, s and t . A natural thing to do would be to construct a graph over the articles and find the shortest s - t path. In case there are no edges between articles, we will have to add them ourselves, e.g., by linking similar articles together.

However, this simple method does not necessarily yield a good chain. Suppose we try to find a coherent chain of events between Clinton’s alleged affair and the 2000 election Florida recount. We pick two representative documents,

s: *Talks Over Ex-Intern’s Testimony On Clinton Appear to Bog Down* (Jan 1998)

t: *Contesting the Vote: The Overview; Gore asks Public For Patience* (Nov 2000)

and find a shortest path between them. The result is shown on Figure 2.1 (left). This chain of stories is rather erratic, passing through the Microsoft trial, Palestinians, and European markets before returning to Clinton and American politics. Note that each transition, when examined out of context, is reasonable: for example, the first and the second articles are court-related. Those correlations are marked by dashed lines in Figure 2.1.

The problem seems to lie with the locality of shortest-path. Every two consecutive articles are related, but there is no *global*, coherent theme to the chain as a whole. Rather, shortest-path may exhibit *stream-of-consciousness* behaviour, linking s and t by a chain of free associations. A better chain is in Figure 2.1 (right). This chain tells the story of Clinton’s impeachment and acquittal, the effect on Al Gore’s campaign, and finally the elections and recount. In the following, we identify the properties which make this chain better.

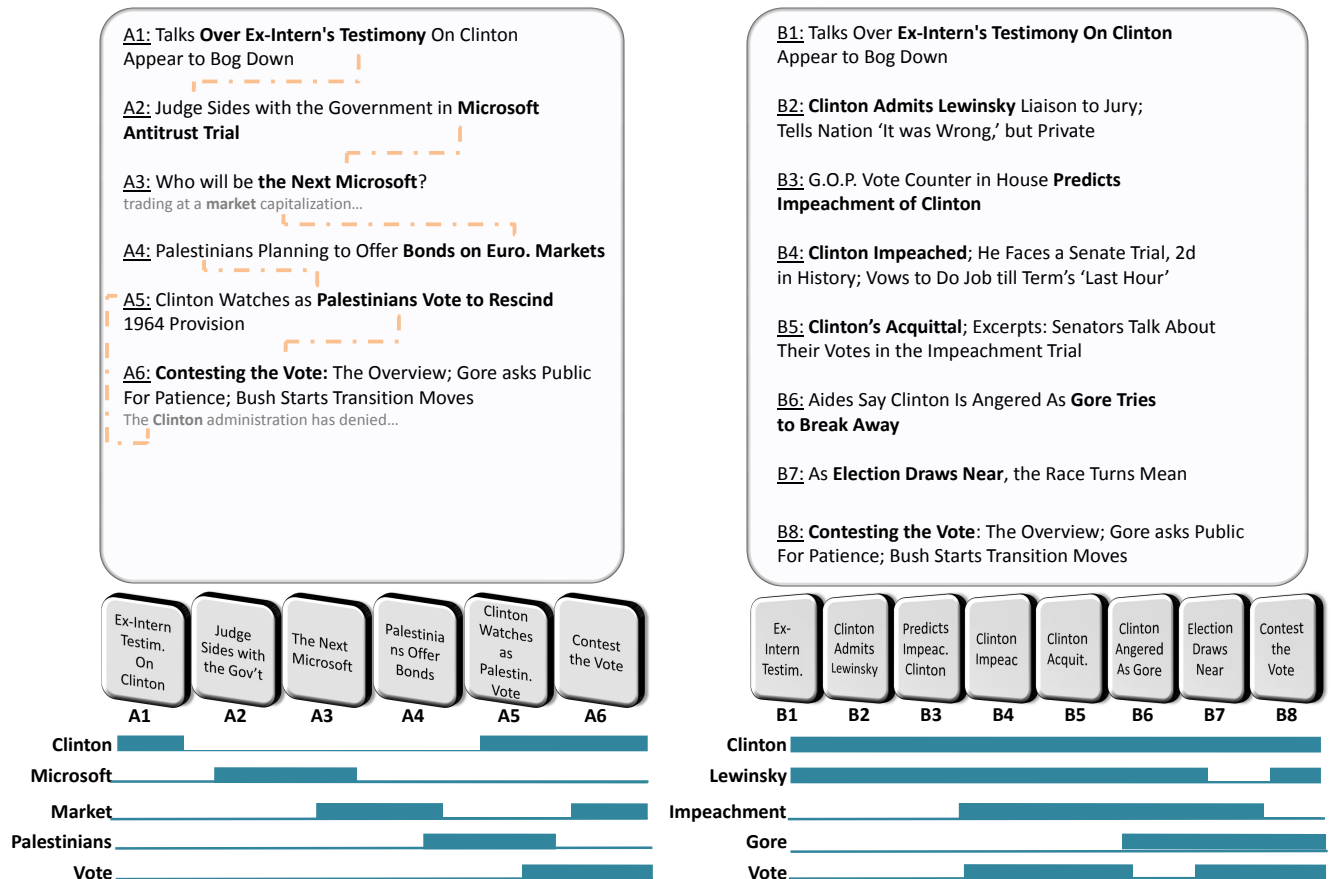


Figure 2.1: Two examples of stories connecting the same endpoints. Left: chain created by shortest-path (dashed lines indicate similarities between consecutive articles). Right: a more coherent chain. Activation patterns for each chain are shown at the bottom; the bars indicate appearance of words in the article above them.

Let us take a closer look at these two chains. Figure 2.1 (bottom) shows word activation patterns along both chains. Bars correspond to the appearance of a word in the articles depicted above them. For example, the word ‘Clinton’ appeared throughout the entire right chain, but only at the beginning and the last two articles on the left. It is easy to spot the associative flow of the left chain in Figure 2.1. Words appear for very short stretches, often only in two neighbouring articles. Some words appear, then disappear for a long period and re-appear. Contrast this with the chain on the right, where the stretches are longer (some words, like Clinton and Lewinsky, appear almost everywhere), and transitions between documents are smoother. This observation motivates our definition of coherence in the next section.

2.1.1 Formalizing story coherence

Let \mathcal{D} be a set of articles, and \mathcal{W} a set of features. For the sake of presentation, assume elements of \mathcal{W} are words or phrases. Each article is a multiset of elements of \mathcal{W} . Given a chain (d_1, \dots, d_n) of articles from \mathcal{D} , we can estimate its coherence from its word activation patterns. One natural

definition of coherence is

$$Coherence(d_1, \dots, d_n) = \sum_{i=1}^{n-1} \sum_w \mathbb{1}(w \in d_i \cap d_{i+1}).$$

Every time a word appears in two consecutive articles, we score a point. This objective has several attractive properties; it encourages positioning similar documents next to each other and rewards long stretches of words. It is also very simple to compute. However, this objective suffers from serious drawbacks:

Weak links They say that a chain is only as strong as its weakest link; this applies to our chains as well. Summing over the transitions can lead to ‘broken’ chains (having weak links), since a chain with many strong links and few weak ones may still score very high. For example, a chain in which all articles but the last one are about the Lewinsky scandal will receive a good score, while not connecting the endpoints in any way.

A more reasonable objective would consider the *minimal* transition score instead of the sum.

$$Coherence(d_1, \dots, d_n) = \min_{i=1 \dots n-1} \sum_w \mathbb{1}(w \in d_i \cap d_{i+1})$$

However, other drawbacks still exist.

Missing words Due to our noisy features, some words do not appear in an article, although they should have. For example, if a document contains ‘lawyer’ and ‘court’ but not ‘prosecution’, chances are ‘prosecution’ is still a highly-relevant word. Considering only words from the article can be misleading in such cases.

Moreover, even if our features were not noisy, an indicator function is not informative enough for our needs.

Importance Some words are more important than others, both on a corpus level and on a document level. For example, in the shortest-path chain, the first two articles shared several words, among them ‘judge’ and ‘page’. Clearly, ‘judge’ is more significant, and should affect the objective function more.

Combining **Importance** and **Missing words**, it becomes clear that we need more than a simple feature indicator. Rather, we need to take into consideration the *influence* of d_i on d_{i+1} through the word w . We defer the formal definition of influence to later (See Sections 4.2.1, 5.2.1 for domain-specific formulations). Intuitively, $Influence(d_i, d_j \mid w)$ is high if (1) the two documents are highly connected, and (2) w is important for the connectivity. Note that w does not have to appear in either of the documents. See Figure 2.2 for an example: the source document d_0 is

d_0 :*Judge Lance Ito lifted his ban on live television coverage of the O.J. Simpson trial*

We calculated word-influence from d_0 to two other documents, using methods explained in Section 4.2.1. The blue bars (in the back) represent word influence for document

d_1 :*O.J. Simpson’s defense lawyers told the judge they would not object to the introduction of DNA evidence*

and the red bars (front) represent word influence for

Word Influence

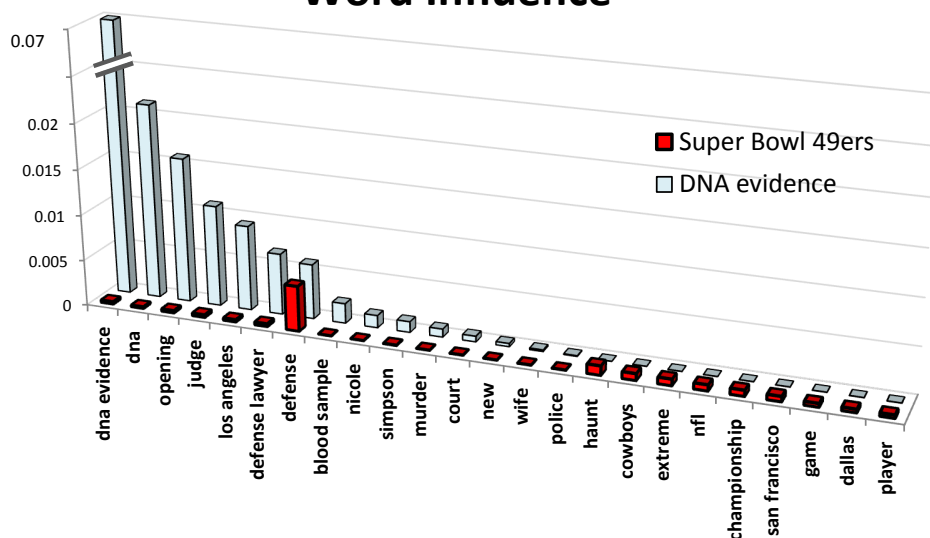


Figure 2.2: Word influence from an article about the OJ Simpson trial to two other documents, one about football and another about DNA evidence.

d_2 : *Winning three consecutive Super Bowls would be a historic accomplishment for San Francisco 49ers*

First, note that the blue bars are generally higher. This means that d_1 is more relevant to the source article d_0 . The influential words for d_1 are mostly court-related, while d_2 's are sport-related (interestingly, the word 'Defense' is strong in both documents, for completely different reasons). Note that many of the influential words do not appear in either of the three articles, thereby solving the **Missing words** problem. With the new *Influence* notion, our objective can be re-defined as

$$Coherence(d_1, \dots, d_n) = \min_{i=1 \dots n-1} \sum_w Influence(d_i, d_{i+1} | w)$$

This new objective, while better, still suffers from the problem of **Jitteriness**.

Jitteriness: the objective does not prevent jittery activation patterns, i.e., topics that appear and disappear throughout the chain.

One way to cope with jitteriness is to only consider the longest continuous stretch of each word. This way, going back-and-forth between two topics provides no utility after the first topic switch. Remember, this stretch is not determined by the actual appearance of the word along the chain; words may have a high influence in some transition even if they are missing from one (or both) of the articles. Rather, we define an activation pattern arbitrarily for each word, and compute our objective based on it. The coherence is then defined as the score under the best

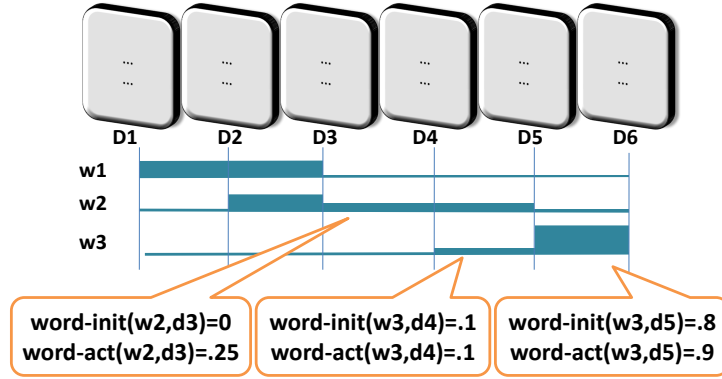


Figure 2.3: An illustration of the results of the linear program, showing initialization and activation levels along a chain for three words. Activation level is the height of the bars. Initialization level is the difference in activation levels between two consecutive transactions, if the activation level has increased.

activation pattern:

$$Coherence(d_1, \dots, d_n) = \max_{activations} \min_{i=1 \dots n-1} \sum_w Influence(d_i, d_{i+1} | w) \mathbb{1}(w \text{ active in } d_i, d_{i+1}) \quad (*)$$

Since influence is non-negative, the best solution is to activate all words everywhere. In order to emulate the behaviour of the activation patterns in Figure 2.1, we constrain the possible activation patterns we consider: we limit the total number of active words and the number of words that are active per transition. In order to avoid multiple stretches, we allow each word to be activated at most once.

Instead of using binary activations (words are either active or inactive), we propose a softer notion of continuous activations. A word's activation is in the range $[0, 1]$, signifying the degree to which it is active. This leads, quite naturally, to a formalization of the problem as a linear program.

Linear Program Formulation

The objective function (*) we defined in the previous section can be readily formalized as a linear program (LP). The LP is specified in Figure 2.4 and illustrated in Figure 2.3.

We are given a chain of n chronologically-ordered documents, d_1, \dots, d_n . First, we define variables describing word activation levels. We define a variable $word-active_{w,i}$ for each document $i = \{1, \dots, n-1\}$ and word w . Variable $word-active_{w,i}$ measures the activation level of w during the transition from d_i to d_{i+1} . In Figure 2.3, those variables are represented by the height of the bars. When a word's activation level increases between two consecutive transactions ($d_{i-1} - d_i - d_{i+1}$), we say it was *initialized* in d_i . We define another variable $word-init_{w,i}$ indicating the initialization level of w in d_i . In the 0-1 case of Figure 2.1, $word-init_{w,i} = 1$ means that w is first activated in d_i . In the continuous case of Figure 2.3, $word-init_{w,i}$ corresponds to the

$$\begin{aligned}
& \max \text{minedge} \\
& \quad \underline{\text{Smoothness}} \\
& \quad // \text{word } w \text{ initialized at most once} \\
& \quad \forall_w \sum_i \text{word-init}_{w,i} \leq 1 \tag{2.1.1} \\
& \quad // \text{if } w \text{ is active in the } i\text{th transition,} \\
& \quad // \text{either it was active before or just initialized} \\
& \quad \forall_{w,i} \text{word-active}_{w,i} \leq \text{word-active}_{w,i-1} + \text{word-init}_{w,i} \tag{2.1.2} \\
& \quad // \text{no words are active before the chain begins} \\
& \quad \forall_w \text{word-active}_{w,0} = 0 \tag{2.1.3} \\
& \quad \underline{\text{Activation Restrictions}} \\
& \quad // \text{no more than } kTotal \text{ words activated} \\
& \quad \sum_{w,i} \text{word-init}_{w,i} \leq kTotal \tag{2.1.4} \\
& \quad // \text{no more than } kTrans \text{ words active per transition} \\
& \quad \forall_i \sum_w \text{word-active}_{w,i} \leq kTrans \tag{2.1.5} \\
& \quad \underline{\text{Objective}} \\
& \quad // \text{minedge holds the minimum score over edges} \\
& \quad \forall_i \text{minedge} \leq \\
& \quad \quad \sum_w \text{word-active}_{w,i} \cdot \text{influence}(d_i, d_{i+1} \mid w) \tag{2.1.6} \\
& \quad \forall_{w,i} \text{word-active}_{w,i}, \text{word-init}_{w,i} \in [0, 1] \tag{2.1.7}
\end{aligned}$$

Figure 2.4: Scoring a chain.

increase of height between two consecutive transitions. Note that in the continuous case, a word can be initialized several times.

The LP has three main parts. In **Smoothness**, we require that the activation patterns are smooth: First, constraint (1) requires that each word is activated at most once. Constraint (2) links the initialization and activation variables together. It ensures that an active word w implies that either w was active in the previous transition, or it just got activated. We also set $\text{word-active}_{w,0} = 0$ (3). Intuitively, it means that no words were active before the beginning of the chain.

In **Activation Restrictions**, we limit the total number of active words (4) and the number of words that can be active during a single transition (5). We use parameters $kTotal$ and $kTrans$ to control the number of active words. The interplay between those two parameters controls the length of activation segments. For example, if $kTotal \sim kTrans \cdot (K - 1)$, the LP might pick different words for every transition, and segments will be short. See Section 2.2.2 for further discussion.

Finally, we get to the **Objective Function**. For every edge i , we calculate its influence. Based

on Equation (*), edge influence is the weighted influence of the active words:

$$\sum_w \text{word-active}_{w,i} \cdot \text{influence}(d_i, d_{i+1} \mid w)$$

Our goal is to maximize the influence of the weakest link: to do this, we define a variable *minedge*, which takes the minimum influence across all edges (6). Our objective is to maximize this variable.

As a sanity check, we tried the LP on real chains. Figure 2.5 (left) shows the best activation pattern found for a chain connecting 9/11 and Daniel Pearl’s murder (top five words). This pattern demonstrates some of the desired properties from Section 2.1: the word ‘Terror’ is present throughout the whole chain, and there is a noticeable change of focus from Bin Laden to Pakistan and the kidnapped journalist. Figure 2.5 (right) shows activation \times influence (rescaled). Notice that words with the same activation levels can have different levels of influence, and thus different effect on the score.

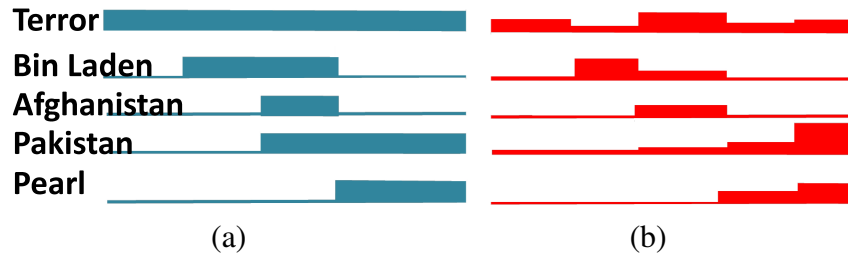


Figure 2.5: Activation patterns found by our algorithm for a chain connecting 9/11 to Daniel Pearl’s murder. (a): Activation levels. (b): Activation levels weighted by the influence (rescaled). For illustrative purposes, we show the result of the integer program (IP) we get replacing constraint (7) of the LP by its binary equivalent.

2.2 Finding a Good Chain

In the previous sections we discussed a method to score a fixed chain. However, we are still left with the problem of *finding* a chain:

Problem 2.2.1. *Given a set of documents \mathcal{D} , two special documents $s, t \in \mathcal{D}$ and an integer K , find a chain of documents of length $K - 2$ that maximizes $\text{Coherence}(s, d_1, \dots, d_{K-2}, t)$.*

One natural way is to use *local search*. In local search, we start from a candidate chain and iteratively move to a neighbour chain, chosen to maximize our scoring function. Local search is easy to understand and to implement; however, it suffers from some known drawbacks, in particular a tendency get stuck in a local optimum. Our weakest-link objective creates a plethora of local optima, thus aggravating the problem.

Another common technique to tackle hard problems is *approximation algorithms*. Approximation algorithms are heuristics that have provably good guarantees on the quality of their solutions. However, in our case, the LP objective poses difficulties along this path.

Consider two coherent chains, (d_1, \dots, d_k) and (d_k, \dots, d_{2k-1}) . Intuitively, concatenating the chains can result in a much-weaker chain; despite the fact that both chains share an article, they may focus on completely different aspects. However, the LP objective implies that

$$\text{Coherence}(d_1, \dots, d_{2k-1}) \geq \frac{1}{2} \min\{\text{Coherence}(d_1, \dots, d_k), \text{Coherence}(d_k, \dots, d_{2k-1})\}.$$

since we can scale down the LP variables of both chains by a factor of two, and the result is a valid solution for the concatenated chain. In other words, concatenating two chains will cause us to lose at most a factor of two. A similar principle holds for concatenating more than two chains. In fact, a greedy algorithm which considers single edges, completely out of context, will result in a $\frac{1}{K}$ approximation ratio.

For this reason, we abandon high approximation ratios. Instead, we will now explore practical ways to speed up the process of finding an optimal chain.

2.2.1 The Algorithm

Of all search strategies used in problem solving, one of the most popular methods for exploiting heuristic information to cut down search time is the *informed best-first* strategy. The general philosophy of this strategy is to use the heuristic information to assess the merit latent in every candidate direction exposed during the search and explore the direction of highest merit first.

Refer to Algorithm 1 for our main algorithm, `findOptimalChain`. The input to the algorithm is a directed graph $G = (V, E)$ over articles, two designated vertices s and t , and an integer K . The output is the most coherent chain between s and t of length K that uses only edges from E . In other words, the graph G encodes the transitions that are valid to use in a chain; for example, we could use it to encode chronological constraints.

The outline of the algorithm is based on the *generalized best-first* strategy [Dechter and Pearl, 1985]. We keep a priority queue of selected chains (line 6). We initialize the queue with the chain consisting only of vertex s (line 7). At each iteration, we expand the chain which features the highest merit (line 9), generating all of its valid extensions (lines 17-18). If the current chain if of length l and ends with vertex u , `validExtensions(u, l)` returns the set of neighbours of u in G which can reach t in exactly $K - l - 1$ steps.

We would like to terminate the search as soon as the first s - t chain of length K is selected for expansion (line 16), but without compromising optimality. In order to do this, it is sufficient to require that the evaluation function used to sort the queue is *admissible* – always provides optimistic estimates of the final costs. We will use the following lemma:

Lemma 2.2.2. *Let $\pi_k = (d_1, d_2, \dots, d_k)$ a chain, and $\pi_{k+1} = (d_1, d_2, \dots, d_k, d_{k+1})$ a chain extending π_k by a single article d_{k+1} . Then $\text{Coherence}(\pi_k) \geq \text{Coherence}(\pi_{k+1})$.*

Proof. Consider the LPs for π_k, π_{k+1} , denoted LP_k and LP_{k+1} . We now show that each feasible solution to LP_{k+1} maps to a feasible solution of the same value for LP_k .

The variables of LP_k are a subset of the variables of LP_{k+1} . We use the simplest mapping: given a solution to LP_{k+1} , we assign the same value to all shared variables of LP_k . This is a feasible solution: Constraints (2),(3),(5),(6) and the relaxed version of (7) are all satisfied

Algorithm 1: findOptimalChain(G, s, t, K)

input : $G = (V, E)$ graph, $s, t \in V$, K integer.
output: A most coherent s - t chain of length K using only edges from E .

- 1 **foreach** $v \in V$ compute possible distances from s to v and from v to t ;
- 2 **if** no s - t chain of length K exists **then**
- 3 **return** \emptyset ;
- // Calculate an upper limit for edge coherence
- 4 **foreach** $(u, v) \in E$ **do**
- 5 $val_{(u,v)} = \text{evalEdge}(u, v)$;
- 6 $Q = \text{New priority queue}$;
- 7 Insert $\langle (s), 0 \rangle_{exact}$ into Q ;
- 8 **while** $Q \neq \emptyset$ **do**
- 9 Extract $p = \langle \pi, val_{\pi} \rangle$ from the top of Q ;
- 10 **if** p is marked ‘approx’ **then** // Replace with a tighter estimate: evaluate with LP of Section 2.1
- 11 $newval_{\pi} = \text{evalLP}(\pi)$;
- 12 Insert $\langle \pi, newval_{\pi} \rangle_{exact}$ into Q ;
- 13 **else** // p is marked as ‘exact’
- 14 // Is it a solution? If not, find valid chain extensions
- 15 Let u be the last node of π ;
- 16 **if** $u = t$ **then** // Found the best s - t chain of length K
- 17 **return** π ;
- 18 // Find all neighbours of u which can reach t in $K - |\pi| - 1$ steps, and insert extended chain into the queue
- 19 **foreach** $v \in \text{validExtensions}(u, |\pi|)$ **do**
- 20 Insert $\langle (\pi, v), \min\{val_{\pi}, val_{(u,v)}\} \rangle_{approx}$ into Q ;

directly by LP_{k+1} . Constraints (1) and (4) provide upper-bounds on summations of non-negative numbers. Since the bound holds for LP_{k+1} , it must hold when we sum over a subset of the numbers. Therefore, constraints (1) and (4) are satisfied in LP_k as well.

Finally, *minedge*, the objective variable, is also a shared variable. Thus, there exists a feasible solution to LP_k of the same value. Since this holds for every feasible solution to LP_{k+1} , we have shown that $Coherence(LP_k) \geq Coherence(LP_{k+1})$. \square

As a direct consequence, any chain extending π is always at most as coherent as π . Therefore, $Coherence(\pi)$ is an optimistic evaluation for each of its extensions, and we can use it without compromising optimality. In lines 11-12 we call `evalLP` to evaluate chains using the LP from Section 2.1.1.

Using the LP to evaluate chains has many benefits; in particular, solving the LP is very quick, usually taking a fraction of a second. In fact, it is so quick that we can use an Integer Program (IP) solver and solve the problem with binary activations. Still, the lemma hints at another shortcut we may take: when we extend chain π with edge e , the value of the extended chain cannot exceed

$$\min\{Coherence(\pi), Coherence(e)\}. \quad (2.2.1)$$

If e happens to be a weak edge, we do not need to solve the LP for the extended chain in order to know it is also weak.


We avoid unnecessary LP computations by caching the coherence of all single edges in advance (line 5). Note that we do not have to solve the LP for single edges. Instead, `evalLP(u, v)` computes the sum of influences of the top $kTrans$ influential words for edge (u, v) . The result is precisely the coherence of the edge, and is very efficient to compute. When we come across a chain for the first time, we evaluate it using equation 2.2.1 (line 18). Only when the chain is chosen for expansion, we replace this approximation with the tighter bound derived from the LP (lines 10-12).

Theorem 2.2.3. *Algorithm `findOptimalChain` always terminates with an optimal solution, or returns \emptyset if no solution exists.*

Proof. In line 1, the algorithm computes all possible distances from s and to t . The algorithm exits on line 3 and returns \emptyset iff s cannot reach t in $K - 1$ steps. By definition, this is exactly the case when no solution exists.

The optimality part of the proof follows from the admissibility of the evaluation function. When the algorithm terminates its search, it has found a chain whose actual coherence is higher than the estimated coherence of any chain which extends any chain in the queue. But since those estimates are admissible (and thus optimistic), we can safely ignore the chains in the queue. \square

In the worst case, the number of nodes expanded is exponential in K . In practice, however, the search is very quick (See Chapter 6).

 **Example 2.2.4.** *Figure 4.2 shows a chain connecting Clinton's first testimony to his acquittal. The chain focuses on the legal process, including subpoenas and witness choices. We reproduce it here for the reader:*

- *Clinton, in First for a President, Testifies in Sex Harassment Suit*
- *SUBPOENAS SENT AS CLINTON DENIES REPORTS OF AN AFFAIR WITH AIDE AT WHITE HOUSE*
- *An Alternative to Impeachment*
- *Obstruction Charge Drove Managers' Witness Choices*
- *A Dispirited Hyde Opposes Indicting Clinton*
- *Road to Reconciliation Appears Long and Hard After Acquittal*

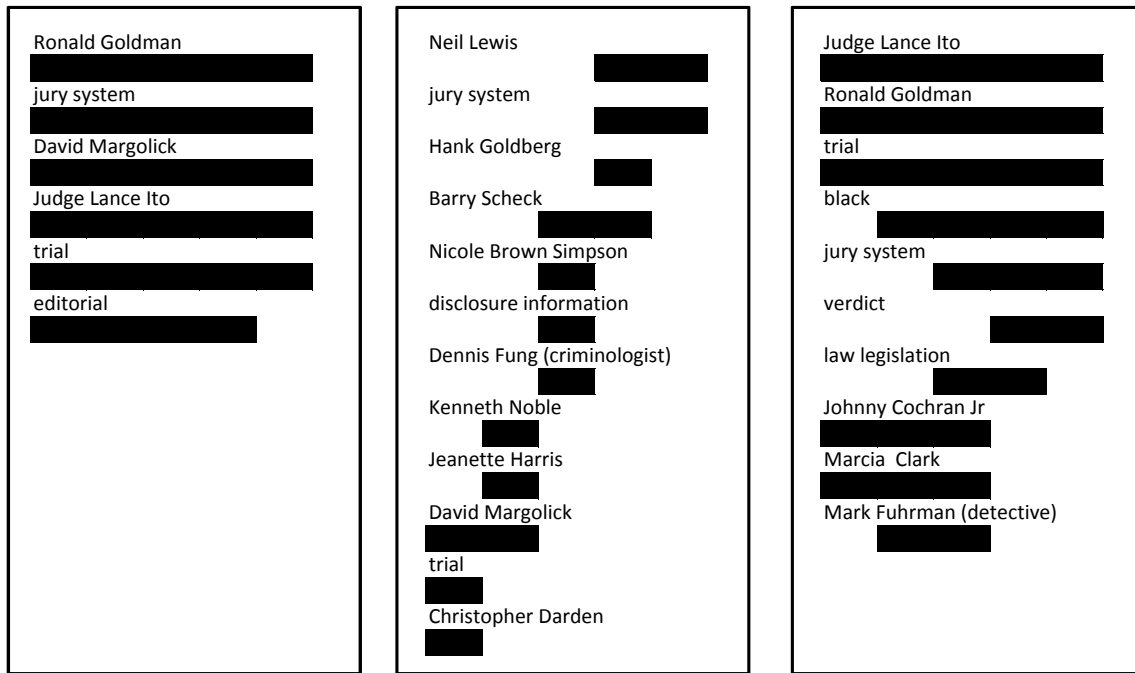
2.2.2 The effect of $kTotal, kTrans$

The LP of Section 2.1 has introduced two parameters, $kTotal$ and $kTrans$. $kTotal$ restricts the total number of active words, and $kTrans$ restricts the number of active words per transition. The choice of parameters determines valid activation patterns, and thus has a significant effect on the value of chains. To demonstrate the effect of these parameters, we have chosen two articles about OJ Simpson's trial. We have then computed the best chain between the articles for different values of $kTotal$ and $kTrans$.

Without loss of generality, we only consider cases where $kTotal \geq kTrans$. If $kTotal \gg kTrans$ (and in particular, when $kTotal$ approaches $kTrans \cdot (K - 1)$), the LP (or IP) can afford to pick $kTrans$ different words for each transition. This results in a behaviour similar to the shortest-path chains of Section 2.1. Figure 2.6(b) shows activation levels of the optimal chain. Note the short segments; as in Section 2.1, they indicate the lack of a global theme.

Setting $kTotal = kTrans$ is equivalent to picking $kTotal$ words to be active throughout the chain. In other words, we are looking for $kTotal$ segments of length K . Similar to the way short segments encourage rare words, long segments push towards common words. See Figure 2.6(a): active words include names of key figures in the trial, and generic words such as 'editorial'. Note that the last segment is not of full length; this implies that activating the word in the final transition would have no effect on our weakest-link objective.

Medium-length segments (Figure 2.6(c)) seem to combine the best of both worlds. Transitions are forced to share words between them, but these words do not have to be common across the entire chain. In our experiments, we have found that $\frac{kTotal}{4} \leq kTrans \leq \frac{kTotal}{2}$ often produces good results for $K = 7$. When $kTotal$ is small, the words tend to capture the essence of the story nicely, but the stories themselves tend to be rather simple. As we increase $kTotal$, the stories become more and more complicated. As before, increasing $kTotal$ too much can result in behaviour similar to the shortest-path chains.



(a) $kTotal = kTrans$

(b) $kTotal \gg kTrans$

(c) $kTotal > kTrans$

Figure 2.6: Activation levels for the best $s-t$ chain for different values of $kTotal$ and $kTrans$.

2.3 Interaction

Thus far, we have defined a way to find chains connecting two endpoints. However, the user may not find the resulting chain satisfactory. In information retrieval systems, the solution is often to let the users revise their queries; for a complex information need, users may need to modify their query many times. In this section, we propose to take advantage of the structured nature of the chains, and explore more expressive forms of interaction. We explore two different types of user feedback: *refinement* of a chain, and *tailoring* to user interests.

Refinement: When presenting a chain to a user, some of the links in the chain may not be obvious. Moreover, the user might be especially interested in a specific part of the chain. For example, a user not familiar with the details of the Lewinsky story might want to further expand the first link of Figure 2.1 (right). We provide the user with a mechanism to indicate areas in the chain which should be further refined; a refinement may consist of adding a new article, or replacing an article which seems out of place.

Since evaluating a single chain is quick, the refinement process is very efficient. Starting from the original chain, we try all possible replacement/insertion actions. We evaluate each chain (see Section 2.1), and return the best one.

In Figure 2.7, the starred article is the result of an insertion request. Adding the article strengthened the end of the chain, while maintaining the global theme.

Incorporate user interests: There can be many coherent ways to connect s and t , especially when they are about similar topics. For example, consider the OJ Simpson trial story.

- Simpson Defense Drops DNA Challenge
- Issue of Racism Erupts in Simpson Trial
- Ex-Detective’s Tapes Fan Racial Tensions in Los Angeles
- Many Black Officers Say Bias Is Rampant in LA Police Force
- With Tale of Racism and Error, Lawyers Seek Acquittal
- ★ **In the Joy Of Victory, Defense Team Is in Discord** ★
 ★ (Defense lawyers argue about playing the race card) ★
- The Simpson Verdict

Figure 2.7: Chain refinement. The starred article was added in order to strengthen the last link.

Suppose the user is interested in the racial aspects of the case, but our algorithm finds a chain focusing on the verdict. We propose a mechanism for the user to focus the chains around concepts they find important. In our case, the user may increase the importance of ‘racial’ or ‘black’, and perhaps decrease the importance of ‘verdict’. We defer the details of this mechanism to Section 3.3.

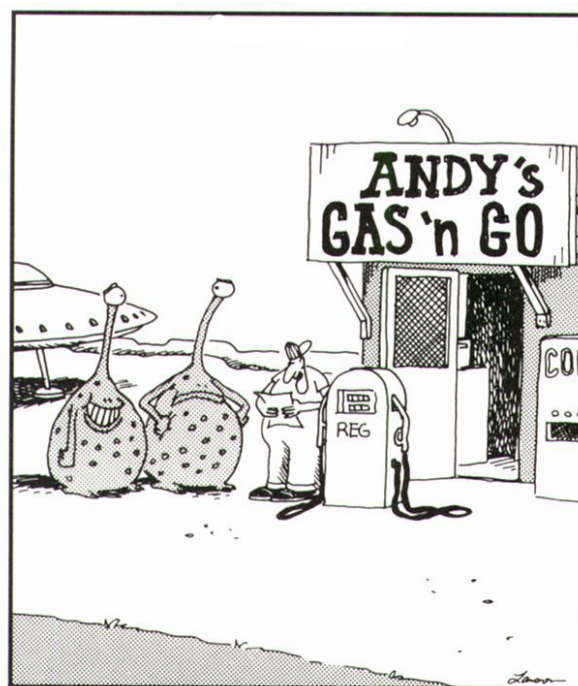
The idea of personal word preferences might also be useful for the **refinement** task. Suppose the user asked to replace an article d_i ; if there are many articles similar to d_i , local search is likely to return one of them. We can implement a mechanism similar to our work in [El-Arini et al., 2009] to find words which might be attributed for the user’s dislike of d_i , and decrease their importance. This way, the replacement article will not be similar.

2.4 Summary

In this chapter we proposed a mechanism for connecting two input articles by a coherent chain. Importantly, we claim that coherence is a *global* property of a chain: coherent chains have a theme running through them. We formalized this intuition as an optimization problem, and used a linear programming solver to score a chain. Next, we discussed mechanisms that allow users to refine and refocus chains according to their interests. In later chapters, we apply these ideas to real-world datasets.

Chapter 3

Constructing a Map



“Shoot! You not only got the wrong planet, you got the wrong *solar* system. ... I mean, a wrong planet I can understand—but a whole solar system?”

This chapter is based on [Shahaf et al., 2012b].

Now that we know how to construct single lines, we can tackle complete maps. In this chapter, we formalize the characteristics of a good map and provide an algorithm to construct one. Our description of the characteristics and the algorithm is intentionally abstract; in chapters 4 and 5 we demonstrate how to adapt these abstract notions to different domains.

In terms of our three characteristics,

-
- * **Expressing information need** The input is a set of articles, \mathcal{D} .
 - * **Structured output** The output is a *map* of articles summarizing and visualizing \mathcal{D} .
 - * **Interaction** Users tailor the map to their interests by increasing/decreasing the importance of specific words.
-

3.1 Properties of a Good Map

What are desired properties of a metro map? In the following, we motivate and formalize several (sometimes conflicting) criteria. In Section 3.2, we present a principled approach to constructing maps that optimize tradeoffs among these criteria. First, we need to formally define metro maps.

Definition 3.1.1 (Metro Map). A metro map \mathcal{M} is a pair (G, Π) , where $G = (V, E)$ is a directed graph and Π is a set of paths in G . We refer to paths as metro lines. Each $(u, v) \in E$ must belong to at least one metro line.

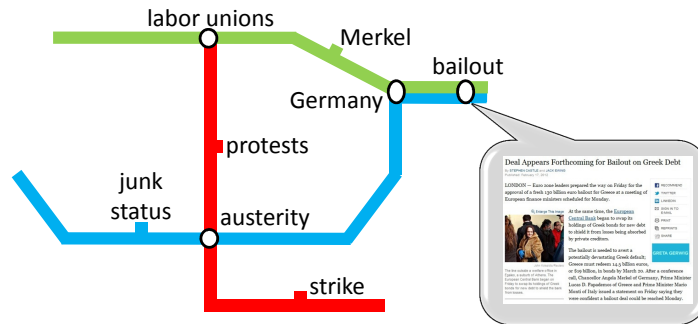


Figure 3.1: Greek debt crisis: a simplified metro map. Metro lines are coherent storylines (stops correspond to articles).

As an example, the map in Figure 3.1 includes three metro lines. Vertices V correspond to news articles, and are denoted by $docs(\mathcal{M})$. The lines of Π correspond to aspects of the story. A key requirement is that each line tells a coherent story: Following the articles along a line should give the user a clear understanding of evolution of a story.

Coherence is crucial for good maps, but is it sufficient as well? In order to put this matter to a test, we found maximally-coherent lines for the query ‘Bill Clinton’ (using methods of Chapter 2). The results were discouraging. While the lines we found were indeed coherent, they were not *important*. Many of the lines revolved around narrow topics, such as Clinton’s visit to Belfast,

or his relationship with his religious leader. Furthermore, as there was no notion of diversity, the lines were very repetitive.

This example suggests that selecting the most coherent lines does not guarantee a good map. Instead, the key challenge is balancing coherence and **coverage**: in addition to being coherent, lines should also cover topics which are important to the user.

Finally, a map is more than just a set of lines; there is information in its *structure* as well. Therefore, our last property is **connectivity**. The map’s **connectivity** should convey the underlying structure of the story, and how different aspects of the story interact with each other.

In Sections 3.1.1-3.1.3, we formalize **coherence**, **coverage** and **connectivity**. In Section 3.1.4, we explore their trade-offs and combine them into one objective function.

3.1.1 Coherence

Coherence was defined in Chapter 2. The coherence of a line is defined as:

$$Coherence(d_1, \dots, d_n) = \max_{\text{activations}} \min_{i=1 \dots n-1} \sum_w Influence(d_i, d_{i+1} | w) \mathbb{1}(w \text{ active in } d_i, d_{i+1}) \quad (*)$$

In other words, coherence is an optimization problem, where the goal is to choose a small set of features (called ‘active’), and score the chain based only on these features. The score of the chain is based on the strength of its weakest link. Constraints on possible activations enforce a small number of words and smooth transitions, imitating the behaviour of Figure 2.1.

Note that in order to compute coherence, we assume we can compute the influence function $Influence(d_i, d_{i+1} | w)$, measuring how important w is in the transition from d_i to d_{i+1} . In Chapters 4 and 5 we give concrete examples of such functions.

Finally, we want all the lines of the map to be coherent. Therefore, the coherence of a *map* is defined as the minimal coherence across its lines Π .

3.1.2 Coverage

In addition to coherence, we need to ensure that the map covers topics which are important to the user. The goal of coverage is twofold: we want to both cover **important** aspects of the story, but also encourage **diversity**.

Before we talk about coverage, we need to define the elements we wish to cover. Let \mathcal{E} be a set of elements. The elements could be low level, such as words (“Obama”, “China”), or high level, such as topics from a topic model. In Section 5.2.2 we propose elements which are not based on article content at all.

In addition to elements \mathcal{E} , we assume we are given a function $cover_{d_i}(e) : \mathcal{E} \rightarrow [0, 1]$, quantifying the amount that document d_i covers element e . In the simplest case, $cover(\cdot)$ is a binary indicator function, effectively turning documents into subsets of features. As another example, if \mathcal{E} is a set of words, we can define $cover(\cdot)$ as tf-idf values.

Our goal is extend the given function $cover(\cdot)$ to maps. Since in our model coverage does not depend on map structure, it is enough to extend $cover(\cdot)$ to a function over *sets* of documents.

A natural candidate for $cover_{\mathcal{M}}(e)$ is to view set-coverage as an additive process:

$$cover_{\mathcal{M}}(e) = \sum_{d_i \in docs(\mathcal{M})} cover_{d_i}(e)$$

Additive coverage is natural and easily computable. However, it suffers from one major drawback: since the coverage each document provides is independent of the rest of the map, additive coverage does not encourage diversity, and redundant maps can achieve high coverage.

In order to encourage diversity, we require that set coverage be a diminishing-returns procedure. In other words, if the map already includes documents which cover e well, $cover_{\mathcal{M}}(e)$ should be nearly maximized, and adding another document which covers e well should provide very little extra coverage of e . This encourages us to pick articles which cover other features, promoting diversity.

Formally, $cover_{\mathcal{M}}(e)$ should satisfy two requirements:

1. **Relation to Single-Document Coverage:** $cover_{\mathcal{M}}(e)$ should be a function of single-document coverage:

$$cover_{\mathcal{M}}(e) \equiv g(\{cover_{d_i}(e) \mid d_i \in docs(\mathcal{M})\})$$


2. **Submodularity:** A set function f is submodular if for all $X \subseteq Y, x \notin Y$ we have

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$$

By requiring that

$$cover_{\mathcal{M}}(e) \equiv f(docs(\mathcal{M}))$$

for some submodular function f , we achieve the diminishing-returns behaviour.

 **Example 3.1.2** (Coverage for the News Domain). *In Section 4.2.2 we define coverage as a sampling procedure:*

$$cover_{\mathcal{M}}(e) = 1 - \prod_{d_i \in docs(\mathcal{M})} (1 - cover_{d_i}(e))$$

Each document in the map tries to cover feature e with probability $cover_{d_i}(e)$. The coverage of e is the probability at least one of the documents succeeded; refer to Section 4.2.2 for further discussion and proof of submodularity.

We now have a way to measure how much a map covers a single feature. Finally, we want to measure how much a map covers the entire *corpus*. The submodularity of $cover_{\mathcal{M}}(e)$ already encourages diversity, but we still need to ensure that the map touches upon aspects of the corpus that are **important**.

In order to know which aspects of the corpus are important to the user, we assume we are given non-negative weights λ_e for each element e , signifying the importance of the element. For

example, if elements are words (and stopwords have been removed), the weights can correspond to word frequency in the dataset.

Finally, we model the amount \mathcal{M} covers the corpus as the weighted sum of the amount it covers each element:

$$Cover(\mathcal{M}) = \sum_e \lambda_e cover_{\mathcal{M}}(e)$$


The weights bias $Cover$ towards maps which cover important features of the corpus. In Section 3.3 we discuss learning a personalized notion of coverage.

3.1.3 Connectivity

Our final property is connectivity. There are many ways to measure connectivity of a map: one can count the number of connected components, or perhaps the number of vertices that belong to more than one line.

We conducted preliminary experiments exploring different notions of connectivity. These results suggest that the most glaring usability issue arises when maps do not show connections that the user knows about. For example, in a map about Israel, a line about legislative elections was not connected to a line about the chosen government’s actions. We came to the conclusion that the connectivity objective needs to be a pairwise function of the metro lines:

$$Conn(\mathcal{M}) = \sum_{i < j} Conn(\pi_i, \pi_j)$$

 **Example 3.1.3** (Connectivity for the News Domain). *In the news domain (Chapter 4), our user studies indicated that the type of connection (one article, multiple articles, position along the line) was not as important as its mere existence. Therefore, we simply defined connectivity as the number of lines of Π that intersect:*

$$Conn(\mathcal{M}) = \sum_{i < j} \mathbb{1}(\pi_i \cap \pi_j \neq \emptyset)$$

3.1.4 Objective function: Tying it all together

Now that we have formally defined our three properties, we can combine them into one objective function. We need to consider tradeoffs among these properties: for example, maximizing coherence often results in repetitive, low-coverage chains. Maximizing connectivity encourages choosing similar chains, resulting in low coverage as well. Maximizing coverage leads to low connectivity, as there is no reason to re-use an article for more than one line.

Let us start with coherence. As mentioned in Section 3.1, we are not interested in *maximizing* coherence. Instead, we treat coherence as a *constraint*: only consider lines above a certain coherence threshold τ , whether absolute or relative. In order to pick τ , one can try different values for multiple queries, taking care not to re-use these queries in user studies. In our experiments, we often used the top 10% of the chains. In the following, we assume that τ is fixed, and denote a chain coherent if its coherence is above τ .

We are left with coverage and connectivity for our objective. Suppose we pick connectivity as our primary objective. Our biggest obstacle is that coherent lines tend to come in groups: a coherent line is often accompanied by multiple similar lines. Those lines all intersect with each other, so choosing them maximizes connectivity. However, the resulting map will be highly redundant. For this reason, we choose coverage as our primary objective.

Let κ be the maximal coverage across maps with coherence $\geq \tau$. We can now formulate our problem:

Problem 3.1.4. *Given a set of candidate documents \mathcal{D} , find a map $\mathcal{M} = (G, \Pi)$ over \mathcal{D} which maximizes $\text{Conn}(\mathcal{M})$ s.t. $\text{Coherence}(\mathcal{M}) \geq \tau$ and $\text{Cover}(\mathcal{M}) = \kappa$.*

In other words, we first maximize coverage; then we maximize connectivity over maps that exhibit maximal coverage.

There is one problem left with our objective: consider two metro lines that intersect at article d . Our coverage function is a set function, therefore d is accounted for only once. In other words, replacing d in one of the lines can only increase coverage. Since there usually exists a similar article d' which can replace d , the max-coverage map is often *disconnected*. Worse yet, it is often *unique*. In order to mitigate this problem, we introduce slack into our objective:

Problem 3.1.5. *Given a set of candidate documents \mathcal{D} , find a map $\mathcal{M} = (G, \Pi)$ over \mathcal{D} which maximizes $\text{Conn}(\mathcal{M})$ s.t. $\text{Coherence}(\mathcal{M}) \geq \tau$ and $\text{Cover}(\mathcal{M}) \geq (1 - \epsilon)\kappa$.*

for a given ϵ . Tuning ϵ on a separate set of queries, we found that values between 0.05-0.15 often give reasonable results.

Finally, we need to restrict the size of \mathcal{M} ; we chose to restrict \mathcal{M} to K lines of length at most l . Alternatively, since some stories are more complex than others, one may prefer to add lines until coverage gains fall below a threshold.

Summary: The map objective can be thought of as:

“ Find a diverse set of important storylines, and show how they interact. ”

Note that maps can be applied to a variety of domains: in essence, maps are useful whenever one can come up with a useful notion of a *storyline*. Storylines do not have to be chronological. Rather, there has to be a dependency structure: reading an article is more beneficial after reading the previous articles in the storyline.

3.2 Finding a Good Map

In this section, we outline our approach for solving Problem 3.1.5. In Section 3.2.1 we represent all coherent chains as a graph. In Section 3.2.2 we use this graph to find a set of K chains that maximize coverage; in Section 3.2.3, we increase connectivity without sacrificing coverage.

3.2.1 Representing all coherent chains

In order to pick good chains, we first wish to list all possible candidates. However, representing all chains whose coherence is at least τ is a non-trivial task. The number of possible chains may be exponential, and therefore it is infeasible to enumerate them all, let alone evaluate them.

Instead we propose a divide-and-conquer approach, constructing long chains from shorter ones. This approach allows us to compactly encode many candidate chains as a graph. See Figure 3.2 for an illustration: each vertex of the graph corresponds to a short chain. Edges indicate chains which can be concatenated and still maintain coherence. A path in the graph corresponds to the concatenated chain.

It is tempting to concatenate any two chains that share an endpoint. That is, concatenate (d_1, \dots, d_k) and (d_k, \dots, d_{2k-1}) to form (d_1, \dots, d_{2k-1}) . However, caution is needed, as combining two strong chains may result in a much weaker chain. For example, Chain B ended with an article about protests in Greece. If we concatenate it with a (coherent) chain about protests across the globe, the concatenated chain will change its focus mid-way, weakening coherence.

The problem appears to lie with the *point of discontinuity*: when we concatenate (d_1, \dots, d_k) with (d_k, \dots, d_{2k-1}) , there is no evidence that both chains belong to the same storyline, despite having a shared article d_k . From the user's point of view, the first k articles are coherent, but (d_2, \dots, d_{k+1}) may not be. This observation motivates our next definition:

Definition 3.2.1 (*m*-Coherence). A chain (d_1, \dots, d_k) has *m*-coherence τ if each sub-chain of length m (d_i, \dots, d_{i+m-1}) , $i = 1, \dots, k - m + 1$ has coherence at least τ .

The idea behind *m*-coherence is to control the discontinuity points. Choosing m is a tradeoff: Increasing m results in more-coherent chains, as the user's 'history window' is wider. However, it is also more computationally expensive. In particular, if $m = l$ the user remembers the entire chain, thus *l*-coherence is equivalent to the regular notion of coherence. If $m = 2$, the user only

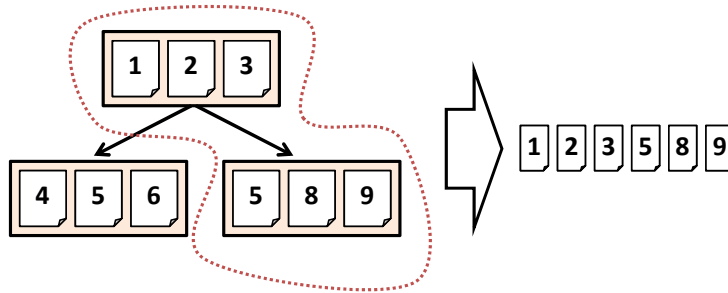


Figure 3.2: Encoding chains as a graph: each vertex of the graph corresponds to a short chain. A path in the graph corresponds to the concatenated chain.

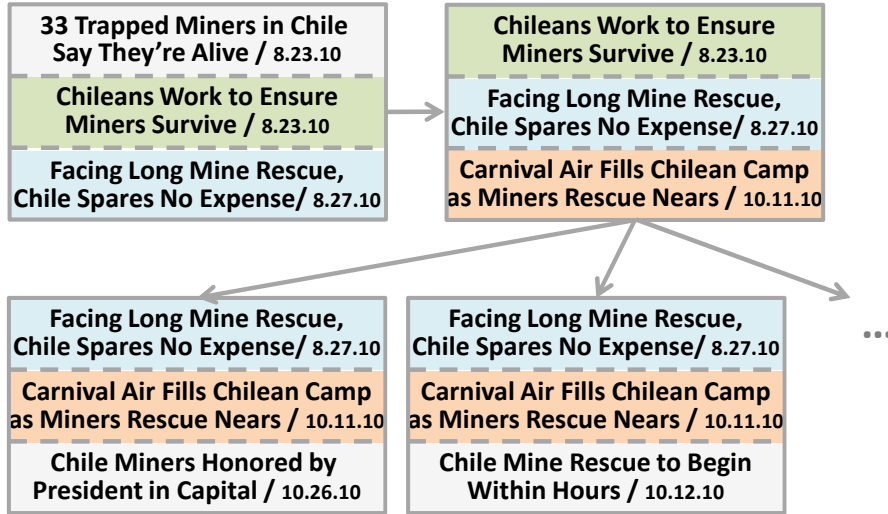


Figure 3.3: A fragment of the coherence graph \mathbb{G} for $m = 3$. Note overlap between vertices.

remembers the last edge; therefore, 2-coherent chains optimize transitions without context, and can result in associative chains like Chain A.

In practice, we chose the highest m we could afford computationally (our website should handle queries in real time). After choosing an appropriate m , we rephrase Problem 3.1.5:

Problem 3.2.2. *Given a set of candidate documents \mathcal{D} , find a map $\mathcal{M} = (G, \Pi)$ over \mathcal{D} which maximizes $\text{Conn}(\mathcal{M})$ s.t. $m\text{-Coherence}(\mathcal{M}) \geq \tau$ and $\text{Cover}(\mathcal{M}) \geq (1 - \epsilon)\kappa$.*

Representing all chains whose m -coherence is at least τ is a less daunting task. Observe that m -coherent chains can be *combined* to form other m -coherent chains if their overlap is large enough. Specifically, we require overlap of at least $(m - 1)$ articles:

Observation 3.2.3. *If chains $c = (d_1, \dots, d_k)$ and $c' = (d_{k-(m-2)}, \dots, d_k, \dots, d_r)$ are both m -coherent for $k \geq m > 1$, then the conjoined chain $(d_1, \dots, d_k, \dots, d_r)$ is also m -coherent.*

The proof follows directly from Definition 3.2.1.

We can now construct a graph \mathbb{G} encoding all m -coherent chains. We call \mathbb{G} a *coherence graph*. Vertices of \mathbb{G} correspond to coherent chains of length m . There is a directed edge between each pair of vertices which can be conjoined ($m - 1$ overlap). It follows from observation 3.2.3 that all paths of \mathbb{G} correspond to m -coherent chains.

We are still left with the task of finding short coherent chains to serve as vertices of \mathbb{G} . These chains can be generated by a *generalized best-first* search strategy, similar to Section 2.2.1. In a nutshell, we keep a priority queue of sub-chains. We initialize the queue with all chains consisting of a single transition. At each iteration, we expand the chain which features the highest coherence, generating all of its extensions. When we reach a chain of length m , we make it into a new vertex and remove it from the queue. We continue until we reach our threshold. Since the evaluation function used to sort the queue is admissible (as a subchain is always at least as coherent a chain which extends it), optimality is guaranteed. In Chapter 6, we present faster methods to find good short chains.

☞ **Example 3.2.4** (Coherence Graphs). *Figure 3.3 shows a fragment of a coherence graph for $m = 3$. The figure depicts multiple ways to extend the story about the trapped Chilean miners: one can either focus on the rescue, or skip directly to the post-rescue celebrations.*

3.2.2 Finding a high-coverage map

In the previous section, we constructed a coherence graph \mathbb{G} representing all coherent chains. Next, we seek to use this graph to find a set of chains which maximize coverage, subject to map size constraints.

Problem 3.2.5. *Given \mathbb{G} coherence graph, find paths $p_1 \dots p_K$ s.t. $Cover(docs(\bigcup_i p_i))$ is maximized, and $|docs(p_i)| \leq l$.*

This problem is NP-hard, which necessitates resorting to approximation methods. First, let us pretend that we can enumerate all paths of \mathbb{G} that contain up to l documents. Then, we can take advantage of the properties of $Cover(\cdot)$, as defined in Section 3.1.2. In particular, since $Cover(\cdot)$ is a linear combination of submodular functions with non-negative coefficients, $Cover(\cdot)$ itself is a submodular function.

Although maximizing submodular functions is still NP-hard, we can exploit the classic result of Nemhauser et al. [1978], which shows that the greedy algorithm achieves a $(1 - \frac{1}{e})$ approximation. In other words, we run K iterations of the greedy algorithm. In each iteration, we evaluate the *incremental* coverage of each candidate path p , given the paths which have been chosen in previous iterations:

$$IncCover(p|\mathcal{M}) = Cover(p \cup \mathcal{M}) - Cover(\mathcal{M})$$

That is, the additional cover gained from p if we already have articles of \mathcal{M} . We pick the best path and add it to \mathcal{M} .

Let us revisit our assumption: unfortunately, enumerating all candidate paths is generally infeasible. Instead, we propose a different approach: suppose we knew the max-coverage path for each pair of fixed endpoints, documents d_i and d_j . Then, we could modify the greedy algorithm to greedily pick a path amongst these paths only. Since there are only $O(|\mathcal{D}|^2)$ such pairs, greedy is feasible.

Computing the max-coverage path between two endpoints is still a hard problem. In order to solve it, we formulate our problem in terms of *orienteering*. Orienteering problems are motivated by maximizing some function of the nodes visited during a tour, subject to a budget on the tour length.

Problem 3.2.6 (Orienteering). *Given an edge-weighted directed graph $G = (V, E, len)$ and a pair of nodes s, t , find an s - t walk of length at most B that maximizes a given function $f : 2^V \rightarrow \mathbb{R}^+$ of the set of nodes visited by the walk.*

We set all edge lengths to 1. We want a path containing at most l articles; since each vertex of \mathbb{G} corresponds to m articles, and the overlap is $m - 1$, we set the budget B to be $l - m$. In addition, we want f to reflect the incremental coverage of path p given the current map, so we define

$$f(p) = IncCover(p|\mathcal{M})$$

We adapt the submodular orienteering algorithms of Chekuri and Pal [2005] to our problem. This is a recursive greedy algorithm (See Algorithm 2). Since we look for paths of a constant size, its running time is polynomial ($O(2nk)^{\log k}$ for paths of length k). Most importantly, it yields an $\alpha = O(\log OPT)$ approximation. We combine the greedy algorithm with submodular orienteering. At each round, we compute approximate best-paths between every two documents (given the chains which have been selected in previous iterations) using submodular orienteering. We then greedily pick the best one amongst them for the map. The algorithm achieves a $1 - \frac{1}{e^\alpha}$ approximation.

Algorithm 2: RecursiveGreedy(s, t, B, X, i) Chekuri and Pal [2005]

input : s, t vertices, B indicates that we seek an s - t walk of length at most B . X is the set we wish to augment, and i indicates allowed depth of the recursion.

- 1 **if** *there is no s - t walk of length at most B* **then**
- 2 | **return** *infeasible*;
- 3 **if** *there is an edge between s and t* **then**
- 4 | $P = s, t$;
- 5 **else**
- 6 | $P = \emptyset$;
- 7 Base case: $i = 0$. **return** P ;
- 8 $m = \text{IncCover}(P \mid X)$;
- 9 **foreach** $v \in V$ **do**
- 10 | **for** $b = 1 \dots B$ **do**
- 11 | $P_1 = \text{RecursiveGreedy}(s, v, b, X, i - 1)$;
- 12 | $P_2 = \text{RecursiveGreedy}(v, t, B - b, X \cup P_1, i - 1)$;
- 13 | **if** $\text{IncCover}(P_1 \cdot P_2 \mid X) > m$ **then**
- 14 | $P = P_1 \cdot P_2$;
- 15 | $m = \text{IncCover}(P_1 \cdot P_2 \mid X)$;
- 16 **return** P ;

The main bottleneck in our algorithm is the need to re-evaluate a large number of candidates. However, many of those re-evaluations are unnecessary, since the incremental coverage of a chain can only decrease as our map grows larger. Therefore, we use CELF Leskovec et al. [2007], which provides the *same* approximation guarantees, but uses lazy evaluations, often leading to dramatic speedups.

3.2.3 Increasing connectivity

We now know how to find a high-coverage, coherent map \mathcal{M}_0 . Our final step is to increase connectivity without sacrificing (more than an ϵ -fraction of) coverage.

In order to increase connectivity, we apply a local-search technique. At iteration i , we consider each path $p \in \Pi_{i-1}$. We hold the rest of the map fixed, and try to replace p by p' that

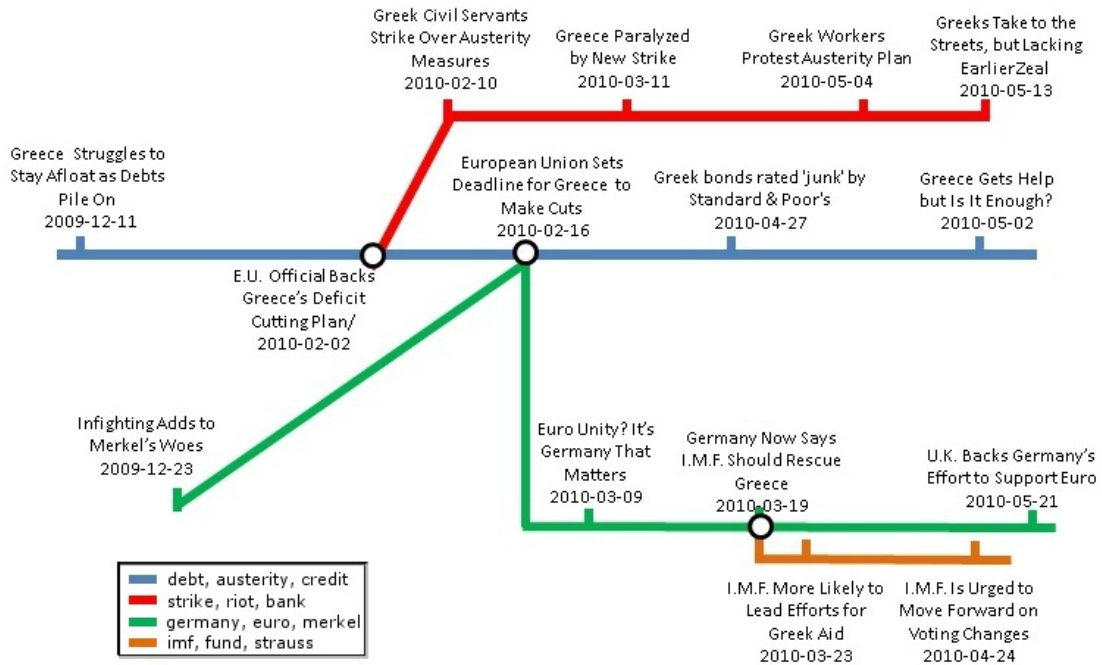


Figure 3.4: An example of our results (condensed to fit space). This map was computed for the query ‘Gree* debt’. The main storylines discuss the austerity plans, the riots, and the role of Germany and the IMF in the crisis.

increases connectivity and does not decrease coverage. At the end of the iteration, we pick the best move and apply it, resulting in \mathcal{M}_i .

In order to find good candidates to replace a path p , we consider the map without p , $\mathcal{M}_{i-1} \setminus p$. We re-use the technique of submodular orienteering and compute approximate max-coverage paths between every two documents. In order to guide the process, we can bias the orienteering algorithm into preferring vertices that already appear in $\mathcal{M}_{i-1} \setminus p$. We consider all chains which do not decrease map coverage, and pick the one which maximizes connectivity. We stop when the solution value has not changed for T iterations. We can apply a variety of local-search heuristics to improve the results, such as random restarts, plateau search, constraint weighting, taboo search, and randomized tie breaking.

Example 3.2.7 (Map). *Figure 3.4 displays a sample map generated by the methodology. This map was computed for the query ‘Gree* debt’. The main storylines discuss the austerity plans, the riots, and the role of Germany and the IMF in the crisis. In order to facilitate navigation in the map, we have added a legend feature. We assign a few characteristic words to each line. The words chosen to describe each line carry the highest incremental coverage for that line.*

3.3 Interaction

3.3.1 Interaction with a Map

Models of interaction can be naturally integrated with metro maps. In order to be useful, the model must be capable of representing users’ interests. We rely on user feedback in order to learn preferences and adjust the maps accordingly. In the following, we illustrate the potential of *learning a personalized coverage function* with an example.

Since our coverage objective is a set function, the most natural notion of feedback from a machine learning perspective would be for users to provide a single label for the map, indicating whether they like or dislike it. However, this approach is not practical. Since there are exponentially many such maps, we are likely to need an extensive amount of user feedback before we could learn the function.

Even more importantly, this approach makes it hard for users to form expressive queries. Ideally, we would like to support queries of the form ‘I want to know more about the involvement of Germany in the debt crisis’, or ‘I do not want to know about Wyclef Jean’s Haitian Presidential bid’. However, labeling entire maps – or even single documents – is just not rich enough to support this query model. Indeed, the user could indicate that they dislike Wyclef Jean articles shown to them, but there is no way for them to specify that they would like to see something which is not on the map.

We propose to let the user provide *feature-based feedback* instead. Feature-based feedback provides a very natural way for supporting the queries mentioned above. For example, the user could increase the importance of the word ‘Germany’ and decrease the importance of ‘Wyclef Jean’ to achieve the desired effect.

There has been growing recent interest in feature-based feedback. Druck et al. [2008] proposed a discriminative semi-supervised learning method that incorporates into training affinities between features and classes. For example, in a baseball vs. hockey text classification problem, the presence of the word “puck” can be considered as a strong indicator of hockey. We refer to this type of input as a *labeled feature*.

Unlike previous approaches that use labeled features to create labeled pseudo-instances, Druck et al. [2008] uses labeled features directly to constrain the model’s predictions on unlabeled instances. They express these soft constraints using generalized expectation (GE) criteria – terms in a parameter estimation objective function that express preferences on values of a model expectation.

We apply the idea of labeled features to metro maps. We aim at creating two classes of documents, roughly meant to represent ‘interesting’ and ‘non-interesting’. Initially, we have no labels; we compute a metro map (as discussed in previous sections) and display it to the user. In addition, we show the user a *tag cloud*. A tag cloud is a visual depiction of words, where size represents frequency. The cloud describes the documents of the map. We let users adjust word importance. For example, importance of 0.9 implies that 90% of the documents in which the word appears are interesting to the user. The relative transparency of the model allows users to make sense of feature weights.

When the user is done adjusting word importance, we train a MaxEnt classifier on \mathcal{D} using those constraints. The classifier then assigns a score μ_i to each document $d_i \in \mathcal{D}$. μ_i represents

the probability that d_i is interesting to the user. Given μ_i , we define a personalized notion of coverage:

$$\text{per-cover}_i(j) = \mu_i \cdot \text{cover}_i(j)$$

Weighting the coverage by μ_i causes non-interesting articles to contribute very little coverage. Note that non-interesting articles may still be picked for the map, e.g. if they are a coherent bridge between two areas of high interest. See examples in Section 4.3.3.

3.3.2 Interaction with a Single Chain

We note that the notions developed in this chapter can help us enhance interactions with a *single* chain as well. In the following, we describe two new ways to connect the dots.

Feature-based Feedback

In Section 2.3 we were interested in ways to let users indicate the concepts they find important. The feature-based feedback of the previous section can work for a single chain as well. As is the case with maps, we are now looking for a maximum-coverage coherent chain (instead of a maximally-coherent chain). In order to find such a chain, we restrict the coherence graph to represent only s - t chains of length at most K , and restrict the number of lines to one.

Connecting One (or Less) Dots

In the previous chapter, we presented a system that could connect two fixed endpoints. However, this form of input is often unrealistic, as the user may not know both the starting and ending points of the story. In this section, we consider the problem of forming a coherent chain when the endpoints are only *partially specified*.

Is this problem harder than our original problem? On one hand, the number of chains to consider is much larger. On the other hand, our algorithms seem easy to extend for the case of partially specified endpoints: if the start (end) point is unknown, we can add a virtual source (sink) node, connect it to all other nodes, and proceed as if the new node was a part of the input.

Let us demonstrate what happens when we extend our algorithm using virtual sources and sinks. Consider again the example from Figure 2.1. Instead of specifying both endpoints (Clinton's alleged affair and the 2000 election Florida recount), our input consists of the Clinton-Lewinsky article alone. The most coherent chain starting with this article is

- Talks over Ex-Intern's Testimony on Clinton Appear to Bog Down
- Is the Ex-Intern Getting Hostility or Sympathy?
- Lewinsky Would Take Lie Test in Exchange for Immunity Deal
- Calls to Intern from Clinton are the Envy of the Capital
- In America: The Clinton M.O.
- Lewinsky Ordered before Grand Jury on Ties to Clinton

As can be seen, the entire chain revolves around the Clinton-Lewinsky story. Furthermore, it is restricted to the very beginning of the story: in fact, the articles barely span more than a month's time.

Note that **having two fixed endpoints served a purpose**: it indicated possible directions which the user cares about. For example, specifying the Florida recount article forced the story to advance along the contemporaneous political events. Furthermore, when only a single endpoint is given, there is no incentive to ever change the topic, and the best chains would likely revolve around their starting point. Worse yet, when *no* endpoints are specified, this starting point might be uninteresting to the user.

In order to ensure that the chain is interesting to the user, we let the user guide us as we build the chain. In the following we define an interactive variant of Connect-the-Dots, called `Connect-A-Dot`. In `Connect-A-Dot`, the user fixes a single article d , and *incrementally* builds a chain around it.

We focus on the case where d is either the first or the last article in the chain (**Where Do We Go From Here?** and **How Did We Get Here?**, respectively). However, our techniques apply to the case d is located at other positions as well.

Algorithm Overview Intuitively, the algorithm is an interactive version of our search algorithm from Section 2.2, where the user guides the search. Our system starts from an input article and incrementally adds articles to the chain. At each step, the system computes the set of valid extensions to the current chain which reach a certain coherence threshold.

The valid extensions represent the many ways in which the chain can progress; we rely on user feedback to choose the next article. Unfortunately, the set of candidate articles is often too big to display. Therefore, our main challenge is to pick a *small* set of *diverse* candidate articles which covers all major aspects of the story.

For example, consider a user who is interested in the story leading to Guantanamo Bay closing. The user picks an input article and feeds it to the system. Figure 3.5 shows a tag cloud representing valid extensions of the input article; the size of a word is proportional to its frequency. Some major aspects of the story (which we should cover) are Obama’s promise, legal aspects, NGO reports, and suicide attempts.

Our algorithm’s outline is described in Algorithm 3. At each iteration, our goal is to pick k candidate articles and show them to the user. The articles should cover the different ways in which the chain may progress. As a first step, we calculate the set of documents $Cands$ which can serve as valid extensions to the chain we have built so far (Line 3). `findExtensionsAboveThreshold` evaluates all $O(|\mathcal{D}|)$ possible documents, and keeps the ones which are above a given threshold (either absolute or relative). Evaluating a single chain is done via the LP of Section 2.1.1, and takes very little time.

Next, we wish to pick a subset $\mathcal{A} \subseteq Cands$ of size b which maximizes coverage of $Cands$ (Line 4), display them to the user, use their choice to extend the chain (Lines 5-6) and re-iterate. Line 4 lies at the heart of the algorithm: in it, we pick a small set of documents that represents the important stories of $Cands$. Luckily, these were our exact requirements from the coverage notion of Section 3.1.2. Furthermore, the submodularity of our coverage notion allows us to exploit the classic result of Nemhauser again [Nemhauser et al., 1978] to obtain a $(1 - \frac{1}{e})$ approximation.

Now that we know how to pick a small set of documents which (approximately) maximizes coverage, we note that we can use the exact same method when no endpoints are specified (`Connect-No-Dots`). In this case, the user first selects a set of candidate documents (e.g.,

Algorithm 3: *interactiveConnectDot*($G, d, isForward, b$)

input : $G = (V, E)$ graph, $d \in V$, *isForward* boolean indicating direction of time, b number of alternatives to show the user.

// Initialize the chain

1 $curChain = (d)$;

2 **while true do**

 // Find candidate articles

3 $Cands = findExtensionsAboveThreshold(G, curChain, isForward)$;

 // Find a small subset which covers $Cands$ well

4 $\mathcal{A} = maximizeCoverage(Cands, b)$;

 // Display subset to user, extend the chain by user's choice

5 $v = getUserChoice(\mathcal{A})$;

6 $curChain = extendChain(curChain, v, isForward)$;

Chapter 4

Case Study 1: News



This chapter is based on [Shahaf et al., 2012b].

In the previous chapter, we defined abstract characteristics of a good map. In this chapter, we demonstrate how to apply these characteristics to the news domain.

4.1 The Need for Maps of News

“Can’t Grasp Credit Crisis? Join the Club”, stated David Leonhardt’s article in the New York Times. Credit crisis had been going on for seven months by that time, and had been extensively covered by every major media outlet throughout the world. Yet many people felt as if they did not understand what it was about.

Paradoxically, the extensive media coverage might have been a part of the problem. This is another instance of the *information overload* problem, long recognized in the computing industry. Users are constantly struggling to keep up with the larger and larger amounts of content that is being published every day; with this much data, it is often easy to miss the big picture.

The consequences of missing the big picture are numerous and substantial. News has fundamental impact on cultural and political life. Understanding news enables the public to make key life decisions, e.g., choosing a place to live or a political orientation. For this reason, there is an increasing need for techniques to present news data in a meaningful and effective manner.

4.2 Objective for the News Domain

In this chapter, we adapt the abstract map objective of Chapter 3 to the news domain. First, we need to understand our data: news articles have a title, a body, and a time stamp. To simplify the problem, we assume that all articles come from reputable sources.

Formally, our universe consists of a set of news articles \mathcal{D} , and a set of features \mathcal{W} . Elements of \mathcal{W} are named entities and noun phrases, that we got by processing the articles with off-the-shelf NLP tools.

4.2.1 Coherence: Measuring influence without links

Let us start with coherence. Recall the coherence objective of Section 2.1:

$$Coherence(d_1, \dots, d_n) = \max_{\text{activations}} \min_{i=1 \dots n-1} \sum_w Influence(d_i, d_{i+1} | w) \mathbb{1}(w \text{ active in } d_i, d_{i+1}) \quad (*)$$

In order to apply coherence to the news domain, we need a notion of *influence*($d_i, d_j | w$) – the influence of document d_i on d_j w.r.t. word w . Intuitively, *Influence*($d_i, d_j | w$) is high if (1) the two documents are highly connected, and (2) w is important for the connectivity. Note that w does not have to appear in either of the documents (refer again to Figure 2.2 for intuition).

Several methods for measuring influence have been proposed. The vast majority of them focus on directed weighted graphs (e.g., the web, social networks, citations), where influence

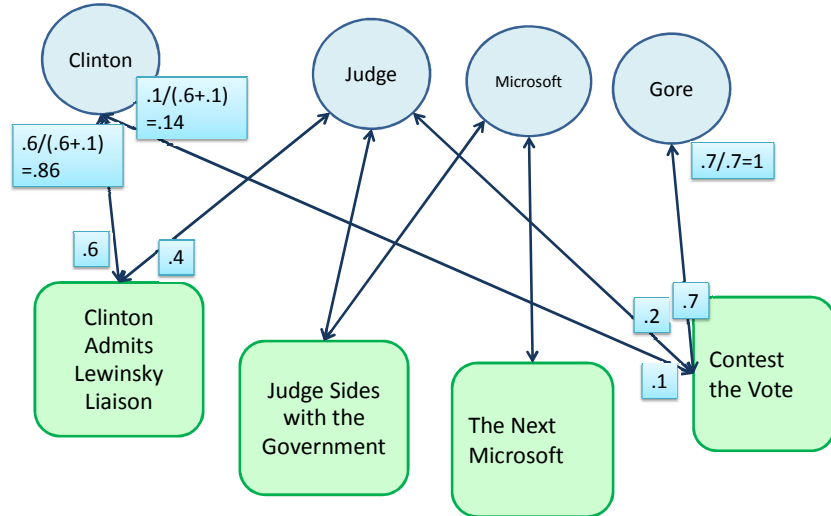


Figure 4.1: A bipartite graph used to calculate influence.

is assumed to propagate through the edges. Methods such as authority computation [Kleinberg, 1999], random graph simulations [Kempe et al., 2003] and random walks [Brin and Page, 1998] all take advantage of the edge structure.

However, in the news setting no edges are present. Adding artificial edges (formally known as “link prediction” [Liben-Nowell and Kleinberg, 2007]) is a complicated and challenging task. In this section, we explore a different notion of influence; despite the fact that this notion is based on random walks, it requires no edges.

First, we construct a bipartite directed graph, $G = (V, E)$. The vertices $V = V_D \cup V_W$ correspond to documents and words. For every word w in document d , we add both edges (w, d) and (d, w) . Refer to Figure 4.1 for a simple graph: there are four (square) documents, and four (circle) words. The leftmost article, about Clinton admitting Lewinsky liaison, is connected to the words ‘Clinton’ and ‘Judge’.

Edge weights represent the strength of the connection between a document and a word. The tool we used for word extraction [Copernic] assigns importance to each word; we use these weights for document-to-word edges. Alternatively, we can use TF-IDF weights. Since we interpret weights as random walk probabilities, we normalize them over all words in the document. For example, the rightmost article is mostly (.7) about Al Gore, and somewhat about ‘Judge’ (.2) and ‘Clinton’ (.1). The word-to-document weights are computed using the same numbers, but normalized over the documents. The word ‘Gore’ can only be reached by a single document, so the edge weight is $\frac{.7}{.7} = 1$. We now use this weighted graph to define influence between documents.

As mentioned before, $Influence(d_i, d_j | w)$ should be high if the two documents are strongly connected, and w plays an important role in this connection. Intuitively, if the two documents are connected, a short random walk starting from d_i should reach d_j frequently. We first compute the stationary distribution for random walks starting from d_i . We control the expected length with a random restart probability, ϵ . The stationary distribution is the fraction of the time the walker

spends on each node:

$$\Pi_i(v) = \epsilon \cdot \mathbb{1}(v = d_i) + (1 - \epsilon) \sum_{(u,v) \in E} \Pi_i(u) P(v | u)$$

where $P(v | u)$ is the probability of reaching v from u .

Intuitively, if d_i and v are very related, $\Pi_i(v)$ is high, as many walks reach v . We now need to factor in the effect of word w on these walks. In particular, we are interested in knowing how many of the walks went through word w before reaching v . To do this, we turn w into a sink node: let $P^w(v | u)$ be the same probability distribution as $P(v | u)$, except there is no way out of node w . Let $\Pi_i^w(v)$ be the stationary distribution for this new graph. If w was influential, the stationary distribution measured at d_j would decrease a lot: in Figure 4.1, without the word ‘Judge’ article 1 (leftmost) is no longer reachable from article 2.

The influence on d_j w.r.t. w is defined as the difference between these two distributions, $\Pi_i(d_j) - \Pi_i^w(d_j)$. Figure 2.2 shows an example of word-influence results calculated by this method. Refer to Section 2.1.1 for a detailed explanation.

The effect of ϵ

The random walk results depend a lot on the choice of ϵ . ϵ controls the expected length of a walk: in expectation, we experience a random restart every $\frac{1}{\epsilon}$ steps.

Prescribing the optimal random-walk length is hard. Each walk should be long enough to explore the “natural cluster” of its starting point d_i , but not long enough to forget where it came from. In particular, lists of influential word created by very long walks tend to include words which are unrelated to d_i . On the other hand, very short walks tend to only produce words from the *immediate* neighbourhood of d_i , losing higher-order word co-occurrences in the process.

Table 4.1 further demonstrates the effect of ϵ . We have computed the influence between two articles about OJ Simpson’s trial. The articles discussed the bloody gloves recovered at the scene of the murder. Table 4.1 shows the top ten influential words for different values of ϵ : for the sake of demonstration, we show the value used in our experiments (0.25), and two extreme values (0.01 and 0.99).

First, we note that the top two words (‘glove’ and ‘apparel’) are the same across the table. This is because these words appear in both d_i and d_j . In addition, they are amongst the most important words in d_i (as reflected by their importance scores). Thus, many of the random walks from d_i to d_j have gone through these words.

When $\epsilon = 0.01$ (long walks), the list contains many words which are ubiquitous throughout the Simpson story, but not necessarily related to the glove episode (e.g. ‘Judge Ito’, ‘Ronald Goldman’, ‘jury’ and ‘police’). As expected, prevalent aspects of the story are heavily represented. For example, the words ‘black’ and ‘detective Mark Fuhrman’ are related to the racial tension of the Simpson case.

On the other hand, when $\epsilon = 0.99$ (short walks), the list contains mostly words which appear directly in d_i (but not necessarily in d_j). When we examine the influence scores, we first notice that they are all very low. This is to be expected, since the probability of reaching d_j is lower to begin with. However, even after normalizing the influences, we still observe a very large gap between the first two words and the rest of the list. As noted before, these words (‘glove’ and ‘apparel’) appear in both d_i and d_j , and thus participate in a path of length two. Longer paths are

0.01	0.25	0.99
glove	glove	glove
apparel	apparel	apparel
David Margolick	Richard Rubin	David Margolick
Judge Lance Ito	Aris Isotoner	murder
Ronald Goldman	Michael Connors	Ronald Lyle Goldman
Mark Fuhrman (detective)	video recording	Simpson murder
black	David Margolick	Los Angeles
jury system	Marcia Clark (deputy dist. atty.)	Nicole Brown Simpson
Kenneth Noble	Richard Rubin	Simpson
police	Bloomington	Johnny Cochran Jr.

Table 4.1: Top 10 influential words for different values of ϵ

much less likely when ϵ is high, and thus words which lie on those longer paths did not get to play an important role in the walks.

Finally, $\epsilon = 0.25$ seems to achieve a healthy balance: the list of words includes mostly words which are directly related to the glove story, but do not necessarily appear in either d_i or d_j . For example, Richard Rubin is a glove designer who testified that leather gloves shrink when exposed to liquid, Aris Isotoner is the company that manufactured the gloves, and Michael Connors is a photographer who took a picture showing OJ Simpson, several years before the trial, wearing a pair of gloves resembling those found at the crime scene.

4.2.2 Coverage

After adapting coherence to the news domain, we focus our attention on coverage. In Section 3.1.2 we required coverage to satisfy two properties:

1. **Relation to Single-Document Coverage:** $cover_{\mathcal{M}}(e)$ should be a function of single-document coverage:

$$cover_{\mathcal{M}}(e) \equiv g(\{cover_{d_i}(e) \mid d_i \in docs(\mathcal{M})\})$$

2. $cover_{\mathcal{M}}(e) \equiv f(docs(\mathcal{M}))$ is submodular.

In order to instantiate the coverage definition of Section 3.1.2, we first need a set of objects to cover, \mathcal{E} . Since we only have the articles' content, we chose the words of \mathcal{W} . $cover_{d_i}(w)$, the amount that document d_i covers feature w , is defined as the corresponding tf-idf value.

Next, we extend $cover(\cdot)$ to maps. We view map coverage as a Bernoulli process (a series of biased coin flips): each document in the map tries to cover feature w with probability $cover_{d_i}(w)$. The coverage of w is the probability at least one of the documents succeeded¹:

$$cover_{\mathcal{M}}(w) = 1 - \prod_{d_i \in docs(\mathcal{M})} (1 - cover_{d_i}(w))$$

¹If $cover_{d_i}(w)$ are very small, we may want to sample more than once from each document.

Thus, if the map already includes documents which cover w well, $cover_{\mathcal{M}}(w)$ is close to 1, and adding another document which covers w well provides very little extra coverage of w . This encourages us to pick articles which cover other features, promoting diversity.

Claim 4.2.1. *The Bernoulli-process view of coverage fulfills the requirements of Section 3.1.2: it is a submodular function of single-document coverages.*

Proof. By construction, $cover_{\mathcal{M}}(e)$ is a function of single-document coverage. In order to prove submodularity, fix $e \in \mathcal{E}$. Define a set function f that corresponds to $cover_{\mathcal{M}}(e)$.

$$f(X) = 1 - \prod_{d \in X} (1 - cover_d(e))$$

Then, we get

$$\forall Y \supseteq X \quad f(X \cup \{x\}) - f(x) = cover_x(e) \cdot \prod_{d \in X} (1 - cover_d(e)) \geq \quad (4.2.1)$$

$$cover_x(e) \cdot \prod_{d \in Y} (1 - cover_d(e)) = \quad (4.2.2)$$

$$f(Y \cup \{x\}) - f(x) \quad (4.2.3)$$

since $cover_d(e) \in [0, 1]$. Therefore, $cover_{\mathcal{M}}(e)$ is submodular. \square

The last thing we are missing from the description of the coverage objective is the definition of the weights λ_e . In the absent of prior knowledge about the user preferences, we set the weight of a feature to be its frequency in the dataset (after the removal of stopwords). This causes the map to prefer highly-mentioned topics. We can use ideas of Section 3.3 to learn personalized weights.

As before, we model the amount \mathcal{M} covers the corpus as the weighted sum of the amount it covers each feature:

$$Cover(\mathcal{M}) = \sum_e \lambda_e cover_{\mathcal{M}}(e)$$

4.2.3 Connectivity

In Section 3.1.3, we defined the connectivity objective as a pairwise function of the metro lines:

$$Conn(\mathcal{M}) = \sum_{i < j} Conn(\pi_i, \pi_j)$$

Our user studies indicated that the type of connection between two lines (one article, multiple articles, position along the line) was not as important to the users as its mere existence. Therefore, we simply define connectivity as the number of lines of Π that intersect:

$$Conn(\mathcal{M}) = \sum_{i < j} \mathbb{1}(\pi_i \cap \pi_j \neq \emptyset)$$

4.2.4 Examples of Maps and Chains

After devising an objective function for the news domain, we apply the algorithms of the previous chapters to find good chains and maps. In the following, we show several examples.

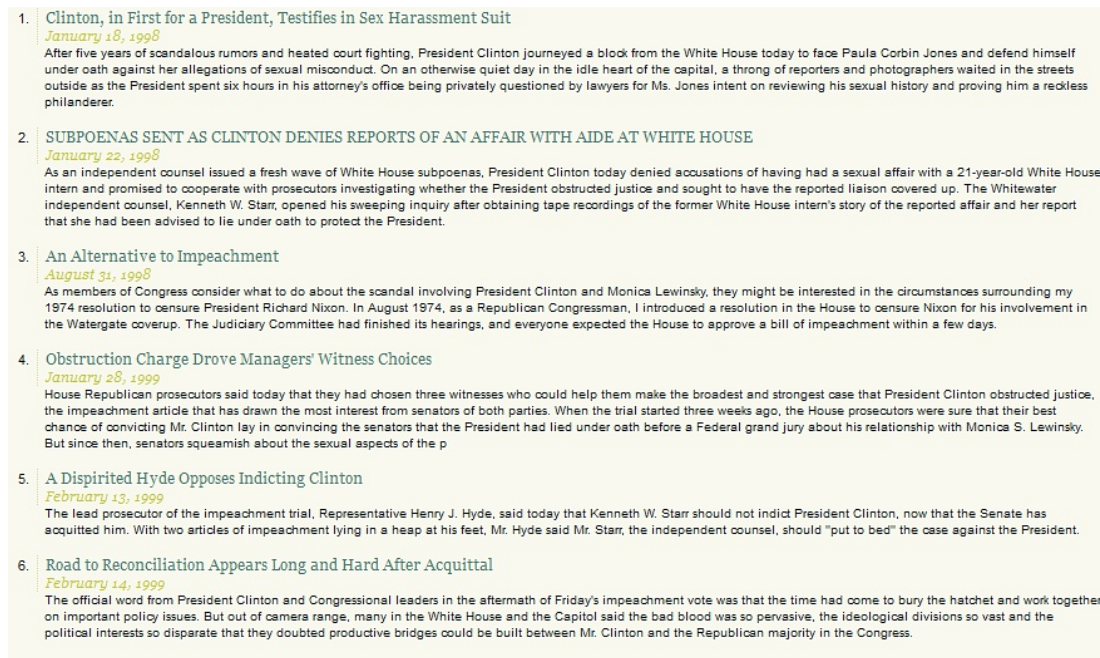


Figure 4.2: A map about the legal process leading to Clinton's acquittal.

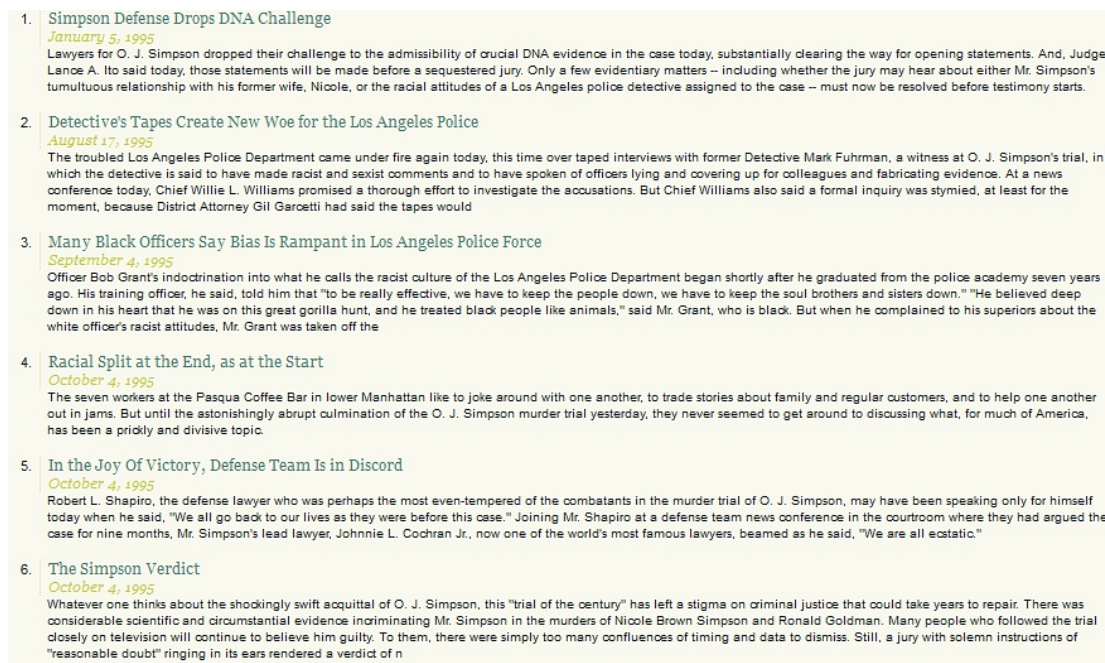


Figure 4.3: A map about the racial aspects of OJ Simpson's trial.

Figure 4.2 shows a chain connecting Clinton’s first testimony to his acquittal. The chain focuses on the legal process, including subpoenas and witness choices. Figure 4.3 shows a chain about the racial aspect of OJ Simpson’s trial, including the LA detective’s racial comments and the racial divide in opinion polls.

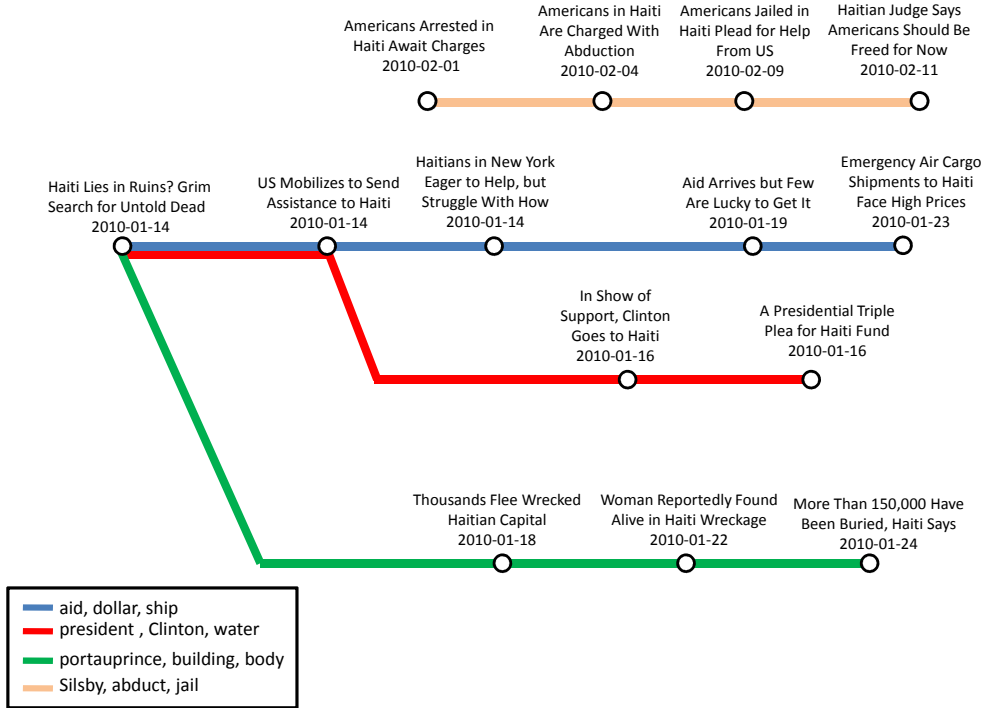


Figure 4.4: Map about the earthquake in Haiti, before interaction.

Figure 3.4 displays a map for the query ‘Gree* debt’. The main storylines discuss the austerity plans, the riots, and the role of Germany and the IMF in the crisis.

Figure 4.4 shows a map for the query ‘Haiti earthquake’. It includes lines about the aid efforts, US presidents’ help, damages to the Haitian capital, and kid abduction charges against a group of American missionaries.

4.3 User study: News

Performance evaluations of information retrieval tasks often focuses on canonical labeled datasets (e.g., TREC competitions) amenable to the standard metrics of precision, recall and variants thereof. The standard methods do not seem to apply here, as they require labeled data, and we are not aware of any labeled dataset suitable for our task. As a result, we evaluated our methods by running them on real data and conducting user studies to capture the utility of our algorithms

as they would be used in practice. In Section 4.3.1 we conduct user studies for the Connect the Dots task (Chapter 2); in Section 4.3.2 we test Metro Maps (Chapter 3).

4.3.1 Connect the Dots: User Study

We evaluate our algorithm on real news data from the New York Times and Reuters datasets (1995-2003). We preprocessed more than half a million articles. These articles cover a diverse set of topics, including world news and politics, economy, sports and entertainment.

Google News Timeline:

- Osama bin Laden is denounced by his family
- Osama Family's Suspicious Site
(Web designer from LA buys a bizarre piece of Internet history)
- Are you ready to dance on Osama's grave?
(How should one react to the death of an enemy?)
- Al-Qaeda behind Karachi blast
- LIVE FROM AFGHANISTAN: Deadline of Death Delayed for American Journalist
- Killed on Job But Spared 'Hero' Label
(About Daniel Pearl)

Connect the Dots:

- Dispatches From a Day of Terror and Shock
- Two Networks Get No Reply To Questions For bin Laden
(Coverage of September 11th)
- Opponents of the War Are Scarce on Television
(Coverage of the war in Iraq and Afghanistan)
- 'Afghan Arabs' Said to Lead Taliban's Fight
- Pakistan Ended Aid to Taliban Only Hesitantly
- Pakistan Officials Arrest a Key Suspect in Pearl Kidnapping
(Pearl abducted in Paksitan while investigating links to terror)
- The Tragic Story of Daniel Pearl

Figure 4.5: Example output chains for Connect-Dots and Google News Timeline. Users were given access to the full articles. The GNT chain is a lot less coherent, and includes several insignificant articles, e.g., an article about a domain name that once belonged to bin Laden's family.

We considered some of the major news stories of recent years: the OJ Simpson trial, the impeachment of Clinton, the Enron scandal, September 11th and the Afghanistan war. For each story, we selected an initial subset of 500 – 10,000 candidate articles, based on keyword-search. The size of the candidate subset depended on the search results. For example, the number of articles mentioning ‘Clinton’ was much higher than those mentioning Enron.

For each article, we extract named entities and noun phrases using Copernic Summarizer [Copernic]. In addition, the NYT dataset includes important meta-data such as taxonomy and section. We remove infrequent named entities and very common nouns and noun phrases (e.g., “year”).

Our main goal was to construct chains representing the stories, and have users evaluate them. For each story, we chose several pairs of articles. We then tried finding chains linking each pair using the following techniques:

- **Connecting-Dots** As described in [Shahaf and Guestrin, 2010], but using the rounding technique we had at the time of the user studies, based on iteratively removing articles.² The typical value of K was 6 or 7. $kTotal$ was set to 15, and $kTrans$ was set to 4. We used the speed-up methods of Chapter 6, and allowed ten minutes for the creation of a chain.
- **Shortest-path** We constructed a graph by connecting each document with its nearest neighbours, based on Cosine similarity. If there was no such path, we increased the connectivity of the graph until a path was found. If the path was too long, we picked a subset of K evenly-spaced documents.
- **Google News Timeline** [Google] GNT is a web application that organizes news search results on a browsable, graphical timeline. The dataset is different, making comparison hard. Also, the input is a query string. We constructed such a string for each story, based on s and t , and picked K equally-spaced documents between the dates of our original query articles. The strings we have used were ‘Clinton Lewinsky’, ‘OJ Simpson’, ‘Enron’, ‘Daniel Pearl’+‘World Trade Center’, and ‘Lewinsky’+‘Elections’.
- **Event threading (TDT)**[Nallapati et al., 2004] is a method to discover sub-clusters in a news event and structure them by their dependency, generating a graph. We found a path in this graph from the cluster including s to the cluster including t , and picked a representative document from each cluster along the path. Again, if the chain was too long, we chose K equally-spaced articles.

First, we presented 18 users with a pair of source and target articles. We gauged their **familiarity** with those articles, asking whether they believe they knew a coherent story linking them together (on a scale of 1 to 5). We showed the users pairs of chains connecting the two articles, generated by the above methods in a double-blind fashion. We asked the users to indicate

- **Relevance:** which chain captures the events connecting the two articles better?
- **Coherence:** which chain is more coherent?

²During each iteration, we solve the LP from [Shahaf and Guestrin, 2010]. We exclude the article with the lowest activation score from the next iterations (setting $node-active_i = 0$). We stop when exactly K of the $node-active_i$ variables are set to 1. Since at every iteration we remove one article, the process is guaranteed to stop after $|\mathcal{D}| - K + 1$ iterations. In practice, it reaches a solution within a few iterations.

- **Redundancy:** which has more redundant articles?

Helping users gain better understanding of a story is the main goal of this paper. In order to quantify this, we also measured the **effectiveness** of the chains. We asked users to estimate how their answer to the **familiarity** question changed after reading each chain. **Effectiveness** is the fraction of the familiarity gap closed. For example, if the new familiarity is 5, this fraction is 1 (gap completely closed). If the familiarity did not change, the fraction is 0.

Example output chains are shown in Figure 4.5. Figure 4.6 shows the results of our user-study. After analyzing the results, we identify two types of stories: *simple* and *complex*. Simple stories tend to focus around the same event, person or institution (e.g., the OJ Simpson trial/the Enron story). Those stories can usually be summarized by a single query string. In complex stories, however, the source and target article are indirectly connected through one or more events (e.g., Lewinsky-impeachment-elections, September 11th-Afghanistan war-Daniel Pearl).

The top plot shows the **effectiveness** (closing the **familiarity** gap) for each of the methods. The bottom of the plot shows familiarity scores for each story before reading any chain. For example, we can see that the Enron story is the least-known story out of the five.

Our algorithm outperforms the competitors on all stories but Enron. The difference is especially pronounced for complex stories. In simple stories, such as Enron, it seems that the simple method of picking K evenly-spaced documents from GNT was sufficient for most people. However, when the story could not be represented by a single query, the effectiveness of GNT decreased.

Surprisingly, the performance of GNT for the OJ story was much worse than its performance for the Enron story. A closer examination revealed that the number of articles about OJ far exceeded the number of Enron articles. Many of the OJ articles were esoteric at best, so picking equally-spaced K documents tended to produce poor results (a book of a former juror made it to the best-seller list, etc.). Furthermore, more of our users were familiar with the OJ story beforehand, so there was less room for improvement.

As expected, shortest path did not perform well. Event threading achieved better results; however, for simple stories, sometimes the clusters were too big. In the Enron story, both s and t belonged to the same cluster, rendering the chain useless. Also, the fact that we pick a representative for each cluster at random might have hurt its performance.

The plot on the bottom of Figure 4.6 shows the percentage of times each method was preferred to another in terms of relevance, coherence and non-redundancy. Users could prefer one chain, or state that both are equally good/bad. Therefore, the numbers do not sum to 100%. The results are grouped based on the type of story (simple vs. complex). Our algorithm is amongst the best in all measures at a statistically significant level. Most importantly, it achieves the best coherence scores, especially for complex stories. We discuss some of the interesting findings below.

Relevance and Redundancy: As expected, for all methods, relevance is good for simple stories but achieving low redundancy is harder. There is a tradeoff – redundancy is easy to avoid by picking random, possibly irrelevant articles. Relevance is easy to achieve by picking articles similar to s or t , but then redundancy would be high.

Google News Timeline is doing well in terms of relevance for simple stories. However, the chains it generates tend to include somewhat insignificant articles, especially for complex stories.

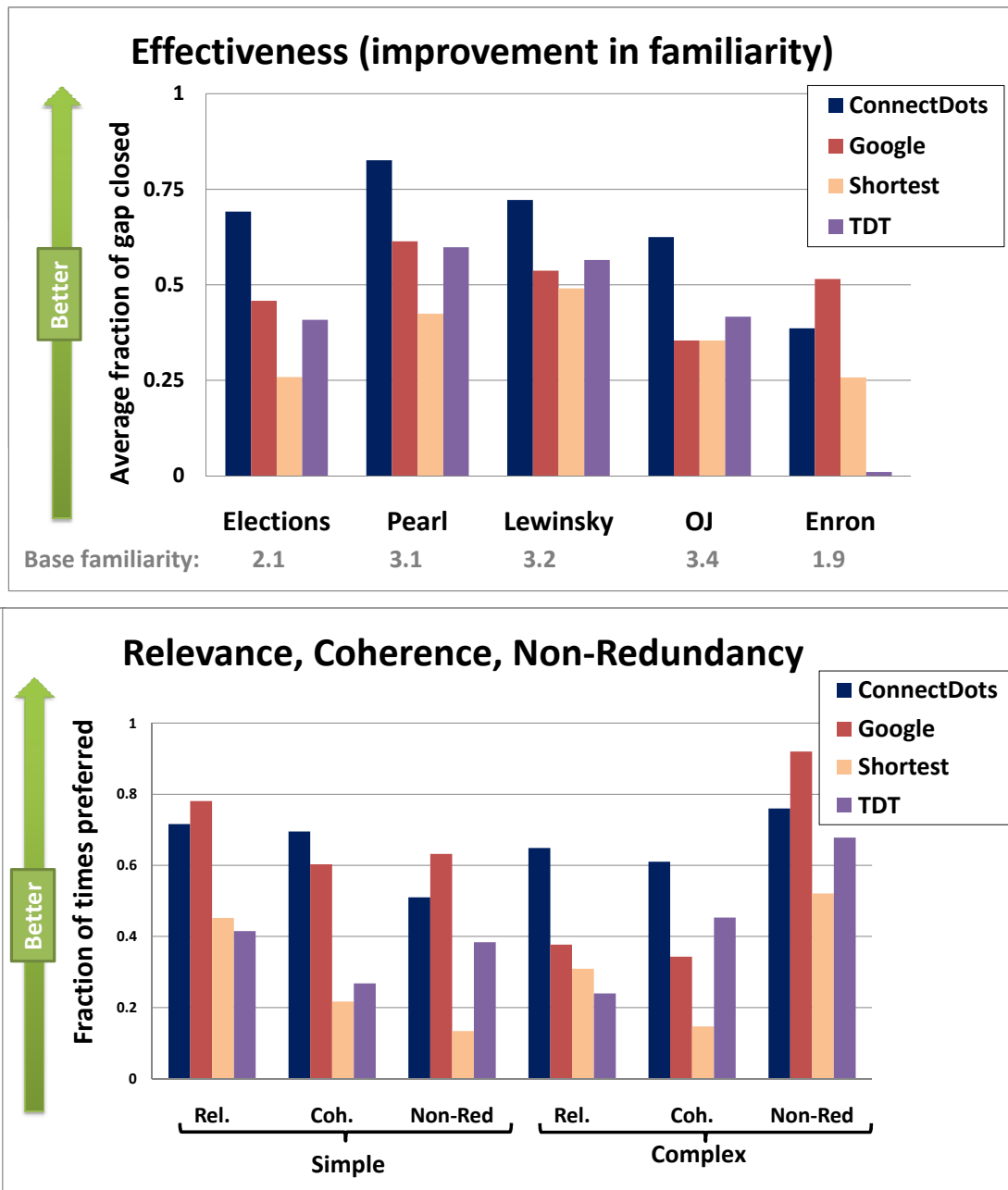


Figure 4.6: Top: Evaluating effectiveness. The average (over users) of the fraction of familiarity gap which was closed after reading a chain. Numbers at the bottom indicate the average familiarity with each story (on a scale of 1 to 5) before reading any chain. Bottom: Relevance, coherence, and non-redundancy (broken down by simple vs. complex stories). The y axis is the fraction of times each method was preferred, compared to another chain. Users could mark chains as ‘equally good’, and therefore the numbers do not sum to 1. Our algorithm outperformed the competitors almost everywhere, especially for complex stories.

The clusters of Event Threading seem to reduce its redundancy, compared to shortest-path.

Coherence: Together with **effectiveness**, this is perhaps our most important metric for evaluating this work. Our algorithm outperforms the other methods, especially in the complex case. This indicates that the notion of coherence devised in this paper matches what the actual users perceive. Interestingly, event threading outperformed GNT for complex stories. This is because the GNT keywords were based on *s* and *t*, and did not capture the intermediate events.

4.3.2 Metro Maps: User Study

In our user study, we evaluate the effectiveness of metro maps in aiding users navigate, consume, and integrate different aspects of a multi-faceted information need. Our experiments were designed to answer the following questions:

Accuracy: How well do the documents selected for the map summarize the topic of the task?

Micro-Knowledge: Can the maps help users retrieve information faster than other methods?

Macro-Knowledge: Can the maps help users understand the big picture better than other methods?

Structure: What is the effect of the map structure?

We assembled a corpus of 18,641 articles from the International section of the *New York Times*, ranging from 2008 to 2010. This corpus was selected because of the material's relative accessibility, as news articles are written with a broad reader population in mind. Stopword removal and stemming have been performed as a preprocessing step.

We created three news exploration tasks, representing use cases where the reader is interested in learning about the trapped Chilean miners, the earthquake in Haiti, and the debt crisis in Greece. We refer to these tasks as *Chile*, *Haiti*, and *Greece*. Our tasks were chosen in order to cover different scenarios: The *Chile* task is very focused, concentrating on a single geographic location and a short time period. The *Haiti* task is broader, and the *Greece* task was the most complicated, as it spans multiple countries for a long period of time. In the following, we outline our evaluations.

Accuracy

In this study, we evaluate the map's content. Before we let users interact with our system and look for information, we want to know whether the information is there at all.

For each task, three domain experts composed a list of the top ten events related to the task. The experts composed their lists separately, and the top ten events mentioned most often were chosen. For example, *Chile* events included the accident, miners' discovery, miners' video, drill beginning and completion, first and last miner outside, release from the hospital, media coverage of the saga, and the presidential ceremony held in the miners' honor.

We then asked the experts to identify those events in metro maps of different sizes (3-6 lines of length at most 6). Below we measure *subtopic recall* (fraction of the important events that are successfully retrieved) of our method. In general, results are high: many of the important events are captured.

Lines	3	4	5	6
Chile	80%	100%	100%	100%
Haiti	50%	70%	80%	80%
Greece	30%	60%	60%	70%

Table 4.2: subtopic recall (fraction of the important events that are successfully retrieved) per map size.

Note that high subtopic precision (fraction of retrieved documents which are relevant to the top ten events) is not a desired property of metro maps: high precision means that the maps is very focused on a small set of events, implying repetitiveness. If the top ten events are already covered, submodular coverage will try to cover side stories as well.

Micro-Knowledge and Structure

In this study, our goal is to examine maps as retrieval tools; we wish to see how maps help users answer specific questions. We compare the level of knowledge (per time step) attained by people using our prototype vs. two other systems: Google News and TDT. **Google News** is a computer-generated site that aggregates headlines from news sources worldwide. News-viewing tools are dominated by portal and search approaches, and Google News is a typical representative of those tools. **TDT** [Nallapati et al., 2004] is a successful system which captures the rich structure of events and their dependencies in a news topic.

We computed maps by methods of Chapter 3. We set $m=3$ for quick computation. After experimenting with several other queries, we set the coherence threshold to top 15%. Instead of fixing the number of chains, we continued to add chains until additional coverage was less than 20% of the total coverage (since we use greedy coverage, there will be at most 5 chains).

We implemented TDT based on [Nallapati et al., 2004] (cos+TD+Simple-Thresholding). We used the same articles \mathcal{D} for maps and for TDT. We picked \mathcal{D} using broad queries: ‘chile miners’, ‘haiti earthquake’ and ‘gree* debt’. We queried Google News for ‘chile miners’, ‘haiti earthquake’ and ‘greece debt’ (plus appropriate date ranges). We did not restrict Google News to NYTimes articles, as not all of them are included. We ensured that all systems display the same number of articles: for Google News, we picked the top articles. For TDT, we picked a representative article from each cluster. The purpose of the study was to test a single query. We defer the evaluation of the interactive component to Section 4.3.3.

We note that comparing the different systems is problematic, as the output of Google News and TDT is different both in content and in presentation (and in particular, cannot be double-blind), so it is hard to know what to attribute observed differences to. In order to isolate the effects of document selection vs. map organization, we introduce a hybrid system into the study: the system, **Structureless metro maps** displays the same articles as metro maps but with none of the structure. Instead, articles are sorted chronologically and displayed in a fashion similar to Google News.

We recruited participants in the study via Amazon Mechanical Turk. Each user chose the number of tasks they were interested in doing out of the three tasks available. For each selected task, one of the four methods was assigned randomly. To make the users more comfortable with

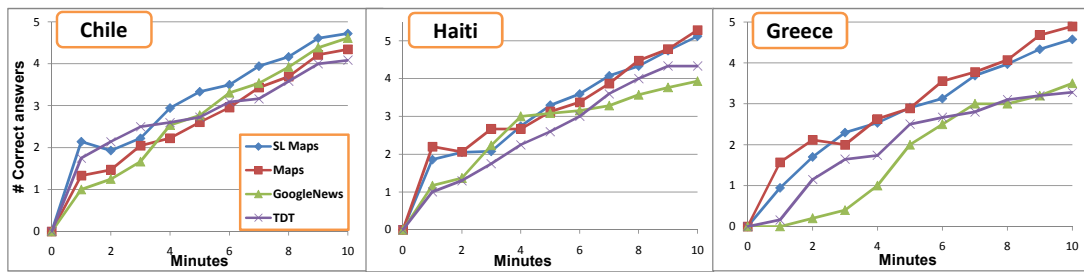


Figure 4.7: User study results: average number of correct answers amongst users vs. time. As the task gets more complex, metro maps become more useful.

the system (and unfamiliar map interface), we asked them to do a warm-up task: copy the first sentence of the tenth article. After the warm-up, users were asked to answer a short questionnaire (ten questions), composed by domain experts. Users were asked to answer as many questions as possible in 10 minutes. In order to counter bias introduced by prior knowledge, the users had to specify the article where the answer was found. A *honey pot* question (an especially easy question, that we expect all workers to be able to answer) was used to identify spammers. After removing users who got this question wrong, we were left with 338 unique users performing 451 tasks.

A snapshot of the users' progress (number of correct answers) was taken every minute. Our interest is twofold: we wish to measure the user's *total knowledge* after ten minutes, and also the *rate* of capturing new knowledge. Figure 4.7 shows the results. x axis corresponds to time, and y axis corresponds to the average number of correct answers.

The results indicate that metro maps are especially useful for complex tasks, such as Greece. In this case, maps achieve higher scores than Google and TDT at the end of the test, as the advantage of structure outweighs the cost of ingesting the additional structure and grappling with an unfamiliar interface. Perhaps more importantly, the rate of capturing new knowledge is higher for maps.

The structureless methods do better for the simple task of Chile. Upon closer examination of the users' browsing patterns, it seems that many of the answers could be found in the (chronologically) first and last articles; the first article provided the basic facts, and the last summarized the story. We believe that this is the reason for the map's performance.

Let us examine Structureless Maps. As discussed earlier, the fact that Structureless Maps outperforms Google News is due to article selection. Metro maps and structureless maps seem comparable, but Metro Map users acquire knowledge more quickly, especially for complex stories; e.g., consider the first few minutes of the Greece task in Figure 4.7.

As a side note, the small number of correct answers is worrisome. We found that the main cause of mistakes was date-related questions; many of the participants entered the article's date, rather than the event's. Since about 30% of our questions involved dates, this affected the results severely. In addition, the majority of Turk users are non U.S.-based (and non-native English speakers)³. When we conducted a preliminary survey across CMU undergrads, the average number of correct answers was significantly higher.

Finally, we compare the *ease of navigation*. If a user has a question in mind, we estimate the

³<http://www.behind-the-enemy-lines.com/2010/03/new-demographics-of-mechanical-turk.html>

difficulty of finding an article containing the answer by computing the number of articles that users clicked per correct answer:

Maps	SL Maps	Google	TDT
2.1	3.74	5.28	4.91

Table 4.3: Ease of navigation: number of articles that users visited per correct answer.

Metro maps require the least amounts of clicks to reach an answer. Most importantly, maps did better than structureless maps, demonstrating the utility of the structure.

Macro-Knowledge

The retrieval study in the previous section evaluated users' ability to answer specific questions. We are also interested in the use of metro maps as high-level overviews, allowing users to understand the big picture.

We believe that the true test of one's own understanding of a topic is their ability to explain it to others. Therefore, we recruited 15 undergraduate students and asked them to write two paragraphs: one summarizing the Haiti earthquake, and one summarizing the Greek debt crisis. For each of the stories, the students were randomly assigned either a metro map or the Google News result page (stripped of logo and typical formatting, to avoid bias).

We then used Mechanical Turk to evaluate the paragraphs. At each round, workers were presented a two paragraphs (map user vs. Google News user). The workers were asked which paragraph provided a more complete and coherent picture of the story; in addition, they justified their choice in a few words ('Paragraph A is more...').

After removing spam, we had 294 evaluations for Greece, and 290 for Haiti. 72% of the Greece comparisons preferred map paragraphs, but only 59% of Haiti. After examining the Haiti paragraphs, we found that the all paragraphs were very similar; most followed the same pattern (earthquake, damages, distributing aid). None of the paragraphs mentioned the Haitian child smugglers, and only one mentioned the temporary laws for dislocated Haitians, despite the fact that both stories appeared in the map. A possible explanation was given by one of the participants: "I chose to not use the American politics. If this is a summary about the event I wanted to remain as objective as possible". In other words, map users avoided some storylines intentionally. As in the previous section, maps are more useful for stories without a single dominant storyline ('the event'), like Greece.

Finally, Figure 4.8 shows tag clouds of the words workers chose to describe the Greece paragraphs. Sample map paragraphs descriptions include 'gives a clear picture' and 'gives a better understanding of the debt crisis'. Google News paragraph descriptions included 'good but just explained about the facts' and 'more like a list of what happened'.

4.3.3 Interaction

In this section we apply the interaction framework of Sections 2.3 and 3.3 to the news domain.



Figure 4.8: Tag clouds representing descriptions of Google News (left) and Map (right) paragraphs. Note maps receive more positive adjectives.

Interaction Usage Example: Connect the Dots

In Section 2.3 we discussed two refinement mechanisms, refinement and feature-based feedback.

The refinement mechanism allows a user to indicate areas in the chain which should be further refined; a refinement may consist of adding a new article, or replacing an article which seems out of place. Figure 2.7 shows an example of refinement.

In a pilot user study, we showed users a chain and asked them to perform a refinement operation (asking for insertion/replacement). We then returned two chains, obtained from the original chain by (1) our local search, (2) adding an article chosen randomly from a subset of candidate articles, obeying chronological order. We asked the user to indicate which chain better fit their request. Users preferred the local-search chains 72% of the time.

In a feature-based feedback study⁴ we showed users two chains – one obtained from the other by increasing the importance of 2-3 words. We then showed them a list of ten words containing the words whose importance we increased and other, randomly chosen words. We asked which words they would pick in order to obtain the second chain from the first. Our goal was to see if users can identify at least some of the words. Users identified at least one word 63.3% of the times.

Interaction Usage Example: Connect a Dot

In Section 3.3.2 we explored an interactive variant of connecting the dots, called *Connect-A-Dot*. In *Connect-A-Dot*, the user fixes a single article d , and *incrementally* builds a chain around it.

We demonstrate our system on the New-York Times dataset. We define our set of features as topics from a latent Dirichlet allocation (LDA) [Blei et al., 2003] topic model learned on the noun phrases and named entities described above. We can directly define $cover_j(i) = P(u_i | doc_j)$, which in the setting of topic models is the probability that doc_j is about topic i . We use the Mallet [McCallum, 2002] implementation of LDA with 20 topics and the default parameter settings. When we start from the general keyword ‘OJ Simpson’, our top five suggestions for a starting point are:

- (1a) A Bit Reluctantly, a Nation Succumbs to a Trial’s Spell
- (1b) Few Can Avoid Harsh Glare Of Murder Trial’s Spotlight
- (1c) Evidence Is Powerful, but He’s Still O.J.

⁴Using an older feedback mechanism. See Shahaf and Guestrin [2010] for details.

- (1d) The Simpson Factor in Race Relations
- (1e) Jurors and Judge Ito: Their Private Lives

Those articles cover various aspects, from the media coverage, through evidence, race relations, and even judge and jury. When we pick 1c (evidence), the system proposes the following options:

- (2a) Tempers Flare Over Simpson DNA Expert's Drug Use
- (2b) Blood Drops Analyzed for Simpson Jury
- (2c) Coroner Says Time of Death Is Imprecise
- (2d) Simpson Defense Changes Glove Tactics
- (2e) Former Simpson Juror Sees Weak State Case

Most of the articles are related to evidence, but they cover different types of evidence, e.g., blood and time of death. When we pick 2d (glove), the system starts to converge. The articles are still mainly about evidence, and almost all of them mention the gloves.

- (3a) Blood on a Simpson Glove Matches Victim's, Expert Says
- (3b) Near-Unanimous Verdict: Blunder at Simpson Trial
- (3c) Key Fibers Linked to Simpson Vehicle
- (3d) Simpson Defense Changes Glove Tactics
- (3e) North Carolina Judge Rules Against Simpson Legal Team

Suppose that, in the previous step, we would have picked article 2a (DNA) instead of 2d (glove). In this case, the algorithm's choice of articles is very different, and much more focused around DNA evidence:

- (3a') Simpson Prosecutors Decide Pathologist Won't Testify
- (3b') Blood on a Simpson Glove Matches Victim's, Expert Says
- (3c') Simpson DNA Papers Go to Smithsonian
- (3d') Again Suggesting Botched Inquiry, Simpson Defense Cross-Examines State DNA Expert
- (3e') At the Bar; The jury is still out on the effects of long, televised trials.

Interaction Usage Example: Map

Figure 4.4 shows a map for the query 'Haiti earthquake'. The blue line revolves around the aid efforts, the red line – about US presidents help, and the green line around the damages to the Haitian capital. The orange line is the one picked last by the greedy coverage algorithm; it discusses kid abduction charges against a group of American missionaries.

We then decreased the importance of the word 'abduct' and recalculated the map. See Figure 4.9 for the result: the abduction line has been replaced by a line about Haitians and the US (in particular, immigration services). Note that the other lines have changed very little, because the first choices of the greedy algorithm were barely affected. The changes may be attributed to the local search stage as well.

In Figure 4.10, we increase the importance of ‘aid’ and ‘rescue’. A new line replaces the (low-coverage) orange line again, this time focusing on the health crisis in Haiti. Note that because this topic is related to the other three lines, our connectivity improves as well.

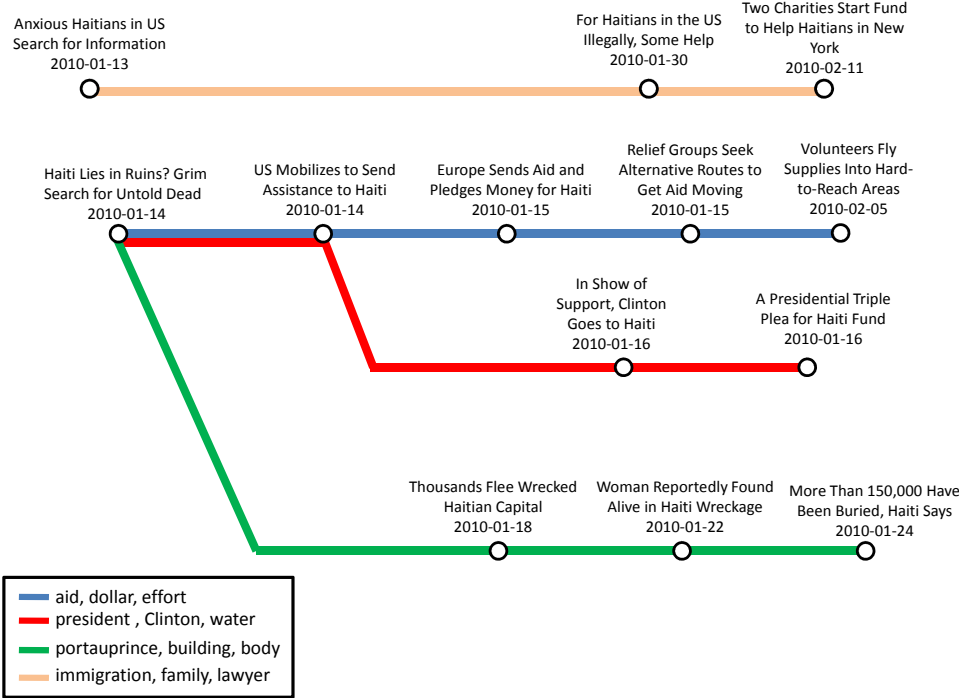


Figure 4.9: Map about the earthquake in Haiti, after decreasing the importance of the word ‘abduct’.

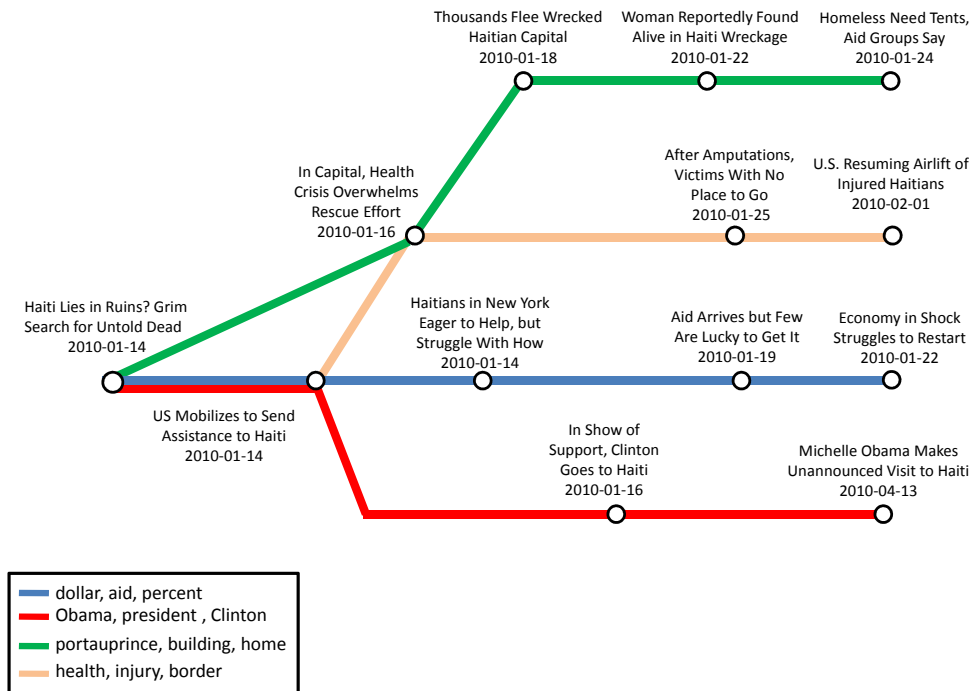


Figure 4.10: Map about the earthquake in Haiti, after increasing the importance of ‘aid’ and ‘rescue’.

4.4 Summary

In this chapter, we applied the ideas of Chapter 3 to the news domain. We defined coherence, coverage and connectivity relying solely on the content of the articles. Pilot user studies demonstrate that the objective we propose captures the users’ intuitive notion of coherence, and that our algorithms effectively help users understand the news; analyzing the results, we learn that maps are especially useful for stories without a single dominant storyline. Finally, we show the potential of interaction mechanisms to personalize maps and chains.

Chapter 5

Case Study 2: Science



Unbeknownst to most historians, Einstein started down the road of professional basketball before an ankle injury diverted him into science.

This chapter is based on [Shahaf et al., 2012a].

In the previous chapter we applied metro maps to the news domain. We now explore another domain: scientific publications.

5.1 The Need for Maps of Science

The scientific community today finds itself overwhelmed by the increasing numbers of publications; relevant data is often buried in an avalanche of publications, and locating it is difficult.

We consider as a sample motivation the creation of valuable literature exploration tools that could help people entering a new field, such as new graduate students or experts reaching beyond their traditional disciplinary borders.

5.2 Objective for Scientific Papers

In this chapter, we adapt the abstract map objective of Chapter 3 to the scientific domain. Let us take a close look at our data. As before, scientific publications have a title, a body and/or an abstract, and a time stamp. They often come with meta data, such as authors and venue.

While in the news domain we were limited to articles' content alone, the scientific domain provides additional structure in the form of the citation graph. In this section, we take advantage of this additional structure and define an objective more suited for scientific papers.

Formally, our universe consists of a set of scientific papers \mathcal{D} , and a set of features \mathcal{W} . Elements of \mathcal{W} are named entities and noun phrases, that we got by processing the papers with off-the-shelf NLP tools. We also have access to a citation graph over \mathcal{D} .

5.2.1 Coherence for Scientific Papers

Let us start with coherence. Recall the coherence objective of Section 2.1.1:

$$Coherence(d_1, \dots, d_n) = \max_{\text{activations}} \min_{i=1 \dots n-1} \sum_w Influence(d_i, d_{i+1} | w) \mathbb{1}(w \text{ active in } d_i, d_{i+1}) \quad (*)$$

In order to apply coherence to the scientific domain, we need a notion of *influence*($d_i, d_j | w$) – the influence of document d_i on d_j w.r.t. word w . The influence notion of the previous chapter relied exclusively on article content. However, the simplicity of the representation can sometimes result in incoherent chains. To illustrate the problem, consider the following three papers:

p1: Multiagent planning with factored MDPs / Guestrin et al / NIPS '01

p2: Timing and power issues in wireless sensor networks / Aakvaag et al / ICPP '05

p3: Social network analysis for routing in disconnected delay-tolerant manets / Daly et al / MobiHoc '07

These papers share many words, such as ‘network’, ‘probability’ and ‘cost’, and thus can achieve a good coherence score. However, they clearly do not follow a coherent research line. The problem may be alleviated by higher-level features (e.g., distinguishing between different uses of ‘network’); in this section, we choose instead to take advantage of the side information provided by the citation graph, and define a coherence notion more suited for scientific papers. In particular, computing influence may benefit from the citation graph.

The citation graph explicitly captures the way papers influence each other: the content of a publication is often affected by cited work, the authors’ prior work and novel insights. The influence notion proposed in BKS [El-Arini and Guestrin, 2011] captures exactly this behaviour. In BKS, the authors define a directed, acyclic graph G_w for every feature w in the corpus. Nodes represent papers that contain w and the edges represent citations and common authorship.

To capture the degree of influence, BKS defines a weight $\omega_{u,v}$ for each edge $u \rightarrow v$ in G_w , representing the probability of direct influence from paper u to paper v with respect to feature w . Some probability is assigned to ‘novelty’, the case that feature w in paper v was novel.

Given a feature-specific weight for each edge in G_w , BKS defines a probabilistic, feature-specific notion of influence between any two papers in the document collection:

Definition 5.2.1 (Direct Influence [El-Arini and Guestrin, 2011]). *Let G_w^r be a random subgraph of G_w , where every edge $u \rightarrow v$ is included in G_w^r with probability $\omega_{u,v}$. The influence between papers p_i and p_j w.r.t. w is the probability there exists a directed path in G_w^r between p_i and p_j .*

The BKS notion of influence has many attractive properties: it is simple, and appears to capture the way ideas travel along the citation graph. However, using it for coherence severely limits the chains we can hope to identify. According to definition 5.2.1, the only pairs of papers that can have influence between them are ancestor-descendant pairs in G_w for some feature w . Therefore, chains with high influence are likely to contain only papers that directly build on top of one another, especially papers by the same authors.

Consider papers p_2 and p_3 from above. Their notion of ‘network’ is similar, but there is no direct path from p_2 to p_3 in the corresponding graph. To mitigate this problem, we introduce a different notion of influence. Rather than requiring that p_i influence p_j , we are only interested in whether feature w in p_i and feature w in p_j refer to the same idea. To capture this property, we modify the notion of influence:

Definition 5.2.2 (Ancestral Influence). *The influence between papers p_i and p_j with respect to feature w is the probability p_i and p_j have a common ancestor in G_w^r .*

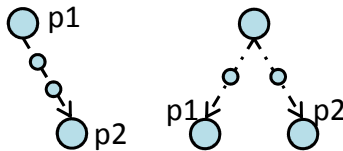


Figure 5.1: Direct (left) vs. ancestral influence (right).

See Figure 5.1 for an illustration of the difference between direct influence (left) and ancestral influence (right). In order for p_i to have direct influence on p_j , there has to be a path from p_i to p_j .

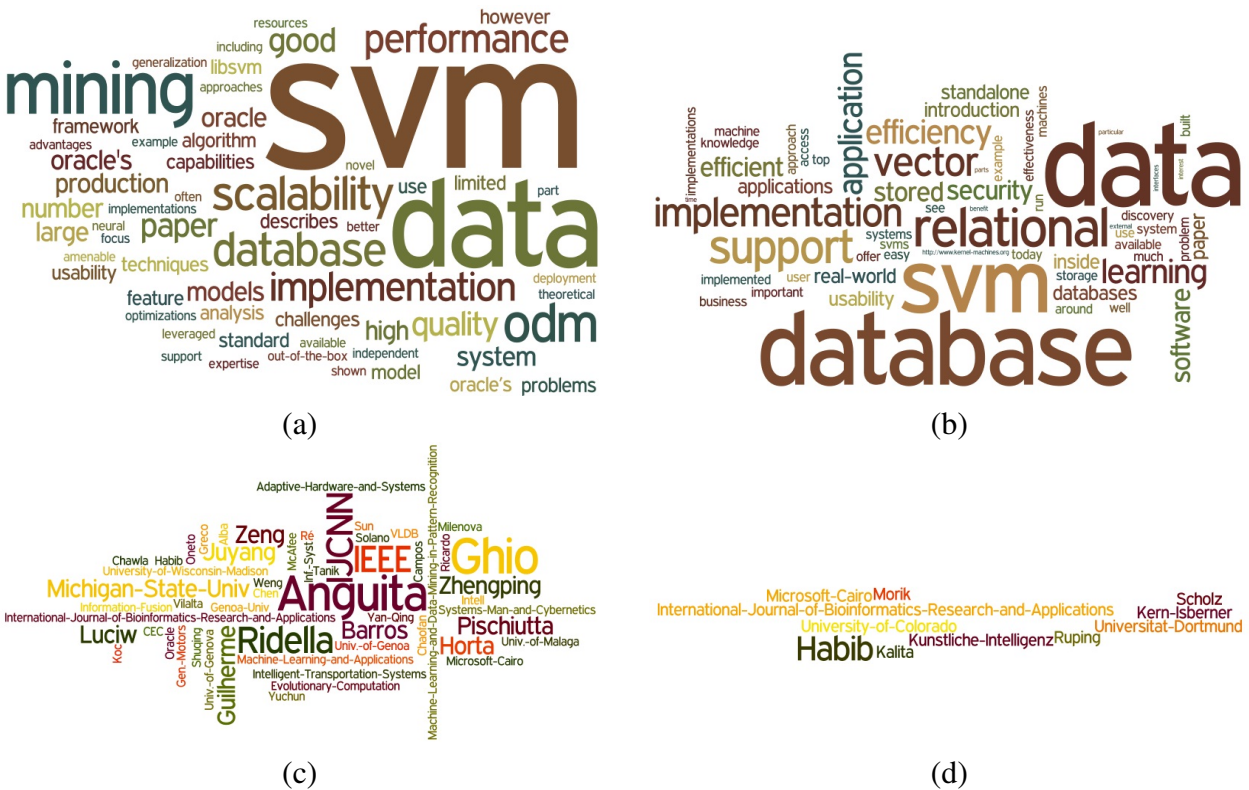


Figure 5.2: Tag clouds for p_1 and p_2 . The size of a word is proportional to its frequency. (a-b) p_1 and p_2 's content, respectively. (c-d) Venues and authors of papers affected by p_1 and p_2 , respectively. Note that (a) and (b) are very similar, but (c) and (d) are not.

In order for p_i to have ancestral influence on p_j , it is sufficient that they have a common ancestor in the graph. The ancestor can also be p_i itself.

As for p_2 and p_3 : with no direct path among them, their direct influence is zero. However, as both cite Perkins' 1999 networks paper, their ancestral influence is non-zero.

5.2.2 Coverage

We now explore coverage notions for the scientific domain. Let us start by choosing the set of elements to cover, \mathcal{E} .

What to cover?

In [Shahaf et al., 2012b], we only had the articles' content to rely upon, and thus the covered elements were *features*. We denoted the amount an article p covered a feature e by $cover_p(e)$, and looked for a set of articles that, when combined, achieved high coverage for many important features.

However, when we applied the same technique to scientific papers, we encountered a problem: papers with similar content may appear exchangeable w.r.t. their coverage, but they will not

necessarily be equivalent in the user’s eyes. For example, the user may notice that the papers aim at different communities, or that one paper is more seminal than the other. Consider the following two papers:

p1: **SVM in Oracle database 10g: Removing the barriers to widespread adoption of support vector machines / Milenova et al**

VLDB '05 Proceedings of the 31st International Conference on Very Large Data Bases

p2: **Support Vector Machines in Relational Databases / Rüping**

SVM '02 Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines

The content of *p1* and *p2* is similar. Figures 5.2(a)-(b) display the papers as tag clouds: both papers share many of their important words (‘data’, ‘database’, ‘svm’, ‘implementation’). Numerous other words have a closely related match (‘performance’/ ‘efficiency’, ‘Oracle’/ ‘relational database’).

One way to distinguish between the aforementioned papers is to examine their impact. Figures 5.2(c)-(d) show tag clouds of authors and venues for papers *citing p1* and *p2*. Figure 5.2(c) has more words than 5.2(d), implying that *p1* has affected more unique authors and venues than *p2*. Interestingly, despite the similar content of the papers, there is almost no intersection between the papers citing them; only a single paper cites both (Mona Habib from Microsoft Cairo).

Based on this intuition, we propose to use the papers themselves as elements of coverage, \mathcal{E} . A paper *p* should cover itself and the papers it has had impact on. By this definition, a high-coverage set of papers consists of papers that, when combined, had impact on a large portion of the corpus.

The idea that a paper covers its descendants (and not its ancestors) may seem counterintuitive at first. After all, how can a paper cover future contributions? Nevertheless, we believe that examining a paper’s ancestors merely helps understanding the context in which the paper was written, while its descendants truly reveal the gist of its contribution.

Coverage of a single paper: Desiderata

We would like papers to cover their descendants. Instead of a hard, binary notion of coverage, we prefer a softer notion, allowing us to express that descendants are covered to various degrees (depicted as a gradient in Figure 5.3a).

Let us concentrate on the degree to which paper *p* covers its descendant *q*, $cover_p(q)$. In order to evaluate the impact that *p* had on *q*, we examine the way *q* is connected to *p* in the citation graph. Intuitively, if *q* can be reached from *p* by many paths, *p* had a high impact on *q*. Since impact is diluted with each step, shorter paths are more important than longer ones.

Before we devise a coverage formulation based on paths between *p* and *q*, we consider another point: impact is not necessarily transitive. Consider, for example, Figure 5.4. The figure outlines a (small) fraction of the descendants of Nicolo Cesa-Bianchi’s paper, ‘How to Use Expert Advice’. As before, edges indicate citation. A snippet from the citation text appears by each edge.

The left branch of Figure 5.4 revolves around Online Learning Theory. The papers in this branch (#2 and #3) build on top of each other. Intuitively, the root paper had impact on both

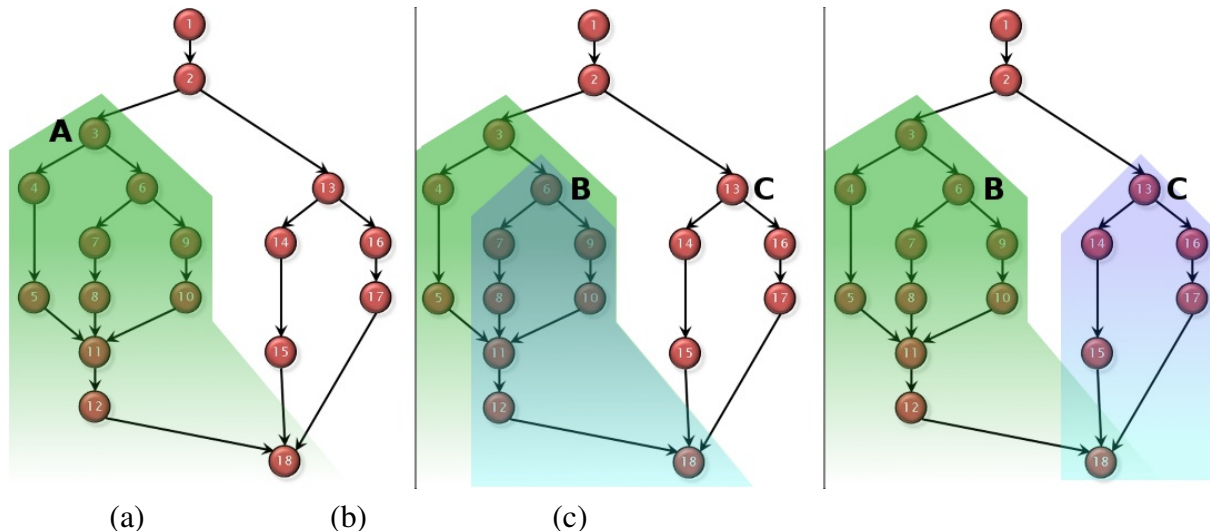


Figure 5.3: A simple citation graph. Edges traverse in the direction of impact, from cited to citing paper. (a) Coverage of document A. Gradient indicates different degrees of coverage. (b-c) The effect of adding papers B and C (respectively) to paper A. Since B’s descendants are already covered to some extent by A, we prefer C.

of them. In contrast, the right branch is more difficult to follow. Both descendants deal with extending the battery life of devices, but while paper #4 is a direct application of the root paper, paper #5 is not. In fact, when #5 cites #4, the citation reads ‘Note that our protocol is different from previous work’. In other words, paper #5 is no longer relevant to the root node, and should not be covered by it.

The difference between the two branches can be captured by the coherence notion of Section 3.1.1: The left branch is much more coherent than the right one. Based on that intuition, we only want a paper to cover the descendants that can be reached by a *coherent* path. Unlike Section 3.1.1, we are only interested in direct-influence coherent chains (Definition 5.2.1), as they model the true impact of a paper.

Coverage of a Single Paper: Formulation

In the previous section, we provided desiderata for $cover_p(q)$: coverage is high if there are many short and coherent paths between p and q . In order to formalize this idea, we employ the technique of random walks.

Let q be a paper. Consider a walk from q to its ancestors, taking only coherent paths into account. At each step, the walker either terminates (with probability α), or chooses an ancestor uniformly at random among the coherent paths that extend the current walk. If there are many short, coherent paths between p and q , there is a high probability that the walk reaches p before termination. We denote this probability by $cover_p(q)$.

Let us formalize this intuition now. Since we only consider coherent paths, it is more convenient to formulate coverage in terms of walks performed directly on a *coherence graph* G . A coherence graph is a graph representing all coherent chains in the domain (See Figure 5.5 for an example. In Section 3.2.1 we explain how to encode the graph compactly). Each vertex v of G

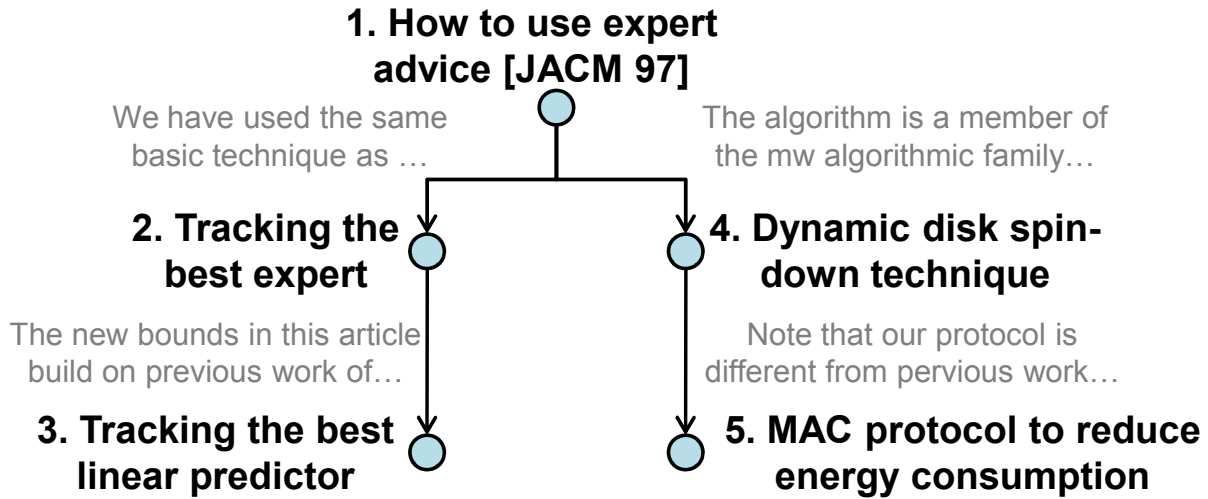


Figure 5.4: Two branches in the citation graph. The left branch is coherent; the right one is not.

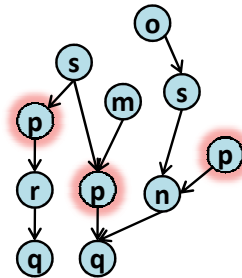


Figure 5.5: Coherence graph. Nodes represent papers (names appear inside). Paths represent coherent chains. Each paper may have multiple corresponding vertices: the highlighted vertices are all copies of paper p .

corresponds to a single paper, which we denote $paper(v)$; each paper p may have multiple corresponding vertices in G , which we denote $copies(p)$. In Figure 5.5, $copies(p)$ are highlighted.

Let G be a coherence graph. For each paper q , we construct the graph G_q by reversing the direction of all edges in G and adding an additional vertex, v_q . v_q is the starting vertex of our walk. We connect v_q to each vertex of G which corresponds to paper q , $copies(q)$. This way, a walk from v_q will always proceed to a copy of q , and then to its ancestors in the coherence graph G . Since the graph is a DAG, the probability that a walk reaches vertex v is easy to compute. We first compute a topological ordering on G_q , and compute the probabilities in this order:

$$cover_v(q) = \begin{cases} P(v_q \rightarrow v), & v \in copies(q) \\ (1 - \alpha) \cdot (\sum_{u:u \rightarrow v} P(u \rightarrow v) \cdot cover_u(q)), & \text{o/w} \end{cases}$$

where $P(u \rightarrow v)$ is the probability the walker chose to go from vertex u to vertex v . We want the walker to choose uniformly among the coherent paths that extend the current walk; in other

words, we want to bias the walker towards ancestors that participate in many coherent paths. Therefore, we compute for each vertex v the number of coherent paths that end in v , $\#Path(v)$. For example, the number of paths that end in the vertex marked ‘n’ in Figure 5.5 is two (o,s,n and p,n). Since G_q is a DAG, computing the number of paths takes polynomial time. The probability that the walker chooses to go from vertex v to vertex u is proportional to $\#Path$:

$$P(u \rightarrow v) = \frac{\#Path(v)}{\sum_{w:u \rightarrow w} \#Path(w)}$$

We now have a coverage notion for vertices of \mathbb{G} . However, we are interested in a coverage notion for *papers*. In order to compute the coverage of paper p , we need to sum up the scores of all vertices in $copies(p)$:

$$cover_p(q) = \sum_{v \in copies(p)} cover_v(q)$$

This score corresponds to the probability of reaching p before termination. In particular, since p can never appear more than once along a path in G , this score always less than 1.

Map Coverage

Now that we have defined coverage of a single document, let us define coverage of a map. In Section 3.1.2 we required that coverage satisfies two properties:

1. **Relation to Single-Document Coverage:** $cover_{\mathcal{M}}(e)$ should be a function of single-document coverage:

$$cover_{\mathcal{M}}(e) \equiv g(\{cover_{d_i}(e) \mid d_i \in docs(\mathcal{M})\})$$

2. $cover_{\mathcal{M}}(e) \equiv f(docs(\mathcal{M}))$ is submodular.

Similar to Section 4.2.2, we view set coverage as a sampling procedure: each paper p_i in the map tries to cover document q with probability $cover_{p_i}(q)$. The coverage of q is the probability at least one of the documents succeeded.

$$cover_{\mathcal{M}}(q) = 1 - \prod_{p_i \in docs(\mathcal{M})} (1 - cover_{p_i}(q))$$

As before, if the map already includes papers which cover q well, $cover_{\mathcal{M}}(q)$ is close to 1, and adding another paper which covers q well provides very little extra coverage of q . This encourages us to pick papers which cover new areas of the graph, promoting diversity.

Figures 5.3b and 5.3c illustrate this idea. Suppose we already have paper A in our map, and we need to choose between papers B and C, whose content is similar. Figures 5.3b and 5.3c show the effect of choosing B and C, respectively. Since B’s descendants have already been covered by A, we would prefer to choose C. (Note that since our coverage is soft, choosing B will still provide gains in coverage.)

Claim 5.2.3. *The notion of coverage fulfills the requirements of Section 3.1.2.*

The proof is identical to Proof 4.2.2.

Finally, we model the amount \mathcal{M} covers the corpus as the weighted sum of the amount it covers each paper:

$$Cover(\mathcal{M}) = \sum_q \lambda_q cover_{\mathcal{M}}(q)$$

With no prior knowledge about the user’s preferences, we set all of the weights to 1. This is equivalent to asking for a map which covers as much of the corpus as possible. We can apply the techniques of Section 3.3 to personalize the weights according to user feedback. In addition to providing feedback for words, we can take advantage of the meta-data and allow users to increase and decrease certain venues or authors. Similar to El-Arini and Guestrin [2011], we can analyze users’ bibliography files to infer their preferences.

5.2.3 Connectivity

In Section 3.1.3, we defined the connectivity objective as a pairwise function of the metro lines:

$$Conn(\mathcal{M}) = \sum_{i < j} Conn(\pi_i, \pi_j)$$

In the news domain (Section 4.2.3) we simply defined connectivity as the number of lines of Π that intersect:

$$Conn(\mathcal{M}) = \sum_{i < j} \mathbb{1}(\pi_i \cap \pi_j \neq \emptyset)$$

Unfortunately, this simple objective does not suffice in the scientific domain. Consider the two chains in Figure 5.6: the top chain describes the progress of margin classifiers – from perceptrons, through linear SVMs, to kernel machines. The bottom chain describes the progress of face-recognition challenge problems in vision: from facial feature location, through face detection, to face recognition. Both chains are clearly related; the vision papers use techniques from the theory chain. However, there is no way to find an article that would belong to both chains, unless we sacrifice coherence considerably. As a result, maps that optimize the aforementioned connectivity notion are often disconnected.

Finding papers that would belong to both chains may be difficult, but we can easily find theory papers that have had a big impact on vision papers. For example, some of the vision papers in Figure 5.6 directly cite papers from the theory chain. These citations are depicted as dashed lines.

Figure 5.6 motivates us to prefer a softer notion of intersection. Rather than requesting that the lines intersect, we also accept lines which are related to each other:

$$Conn(\mathcal{M}) = \sum_{i < j} \mathbb{1}(\pi_i \cap \pi_j \neq \emptyset) + \gamma \cdot cover(\pi_i, \pi_j)$$

where $cover(\pi_i, \pi_j)$ is the maximal $cover_p(q)$ for $p \in \pi_i, q \in \pi_j$, or vice versa. We choose to use the maximum (instead of sum) in order to encourage connections between as many pairs of lines as possible. Scoring all the connections between π_i and π_j may lead to maps where only

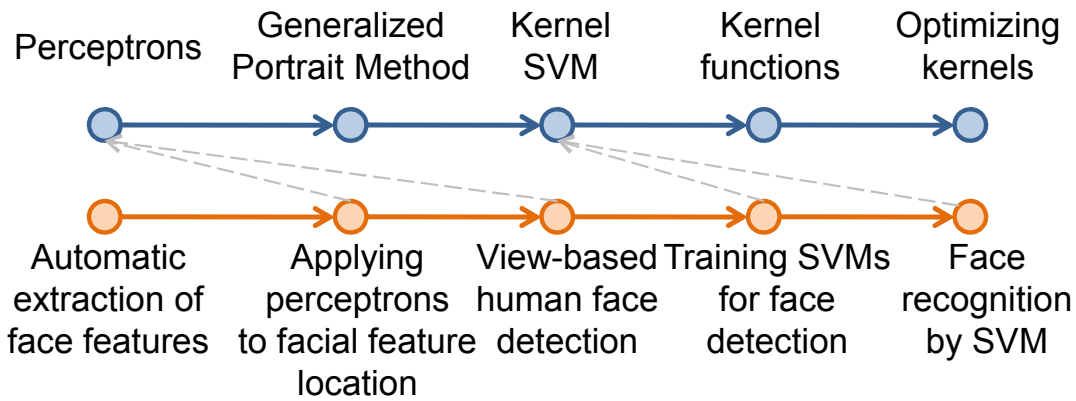


Figure 5.6: Two coherent chains (theory of SVMs, application of SVM to vision). The chains do not intersect, yet are related: the application chain uses tools from the theory chain. Dashed gray lines indicate impact.

a few lines are very well-connected, and the rest are disconnected. The parameter γ is chosen empirically.

This softer notion of intersection is especially suited to scientific literature. Publications offer a rich palette of interaction possibilities, such as affirmation, criticism, contrast, methodology, and related work. Exposing the relationships between two lines of research can prove extremely valuable to researchers.

5.2.4 Example Maps

After devising an objective function for the news domain, we apply the algorithms of the previous chapters to find good maps.

Figure 5.7 shows a part of a map computed for the query ‘Reinforcement Learning’. As can be seen, the map depicts multiple lines of research: MDPs, robotics and control, multi-agent cooperation, bounds and analysis, and exploration-exploitation tradeoffs. The map shows how the MDP line affects the multi-agent and robotics lines, and how the exploration-exploitation line interacts with the analysis line. Those relations are depicted as gray dashed paths. Note that the map does not capture all the interactions; for example, connections between MDPs and the analysis line are not captured.

As mentioned in Section 5.2.3, intersection is rare for broad queries. Figure 5.8 shows one such intersection between two lines in the SVM map. One line is about large-scale SVMs, the other is about multi-class SVMs. The lines intersect at Keerthi’s paper about large scale multi-class linear SVMs.

5.3 User study: Science

In our user study, we evaluated the effectiveness of metro maps in aiding users navigate, consume, and integrate different aspects of a specific, multi-faceted information need.

Evaluating metro maps in the scientific domain poses some significant challenges. Since the metro-map output is unique, we cannot conduct a double-blind comparison study, as subjects inevitably differentiate between the different systems. Therefore, we cannot have a within-subject study, but are instead forced to choose a between-subject design. This design, in itself, causes a new problem: since we need a different group of participants for each condition tested (metro-map or competitor), we cannot tailor the query to users. Rather, we have to find a single domain such that all of our participants will (1) be able to read scientific publications in that domain and (2) not know the domain well in advance.

We recruited 30 participants from our university. All participants were graduate students with background in Machine Learning or related fields. The domain we chose was Reinforcement Learning. The machine learning background of the participants was enough to make them comfortable with the subject, but none of them had conducted research in the field or studied it extensively.

We asked participants to imagine themselves as first-year graduate students embarking on a research project in Reinforcement Learning. The participants were asked to conduct a quick literature survey. In particular, they were asked to update a survey paper from 1996: identify up to five research directions that should be included in the updated survey, and list a few relevant papers for each direction. We recorded participants' browsing histories, and took a snapshot of their progress every minute. We limited their time to 40 minutes to simulate a quick first pass on papers.

We used the ACM dataset to compute a map for the query 'Reinforcement learning'. The dataset contains more than 35,000 papers from ACM conferences and journals. As the number of papers is relatively small, scalability was not an issue. We extracted features as described in [El-Arini and Guestrin, 2011]. We had two conditions, *GS* and *MP+GS*: In *GS*, participants were allowed to use Google Scholar¹, a search engine that indexes scholarly literature. In the second condition (*MP+GS*), participants were given the pre-computed metro map, and asked to pretend that they stumbled upon it; they were not instructed how to use the map. In addition to the map, the participants could access Google Scholar.

We also included two simulated conditions in the study, *MP* and *WK*: In *MP*, we pretended our map was the user's output, and listed all of its papers. In *WK*, we used references from the Wikipedia article about reinforcement learning.

We decided to compare against Wikipedia and Google Scholar since they represent two of the most popular starting points for research queries today. Other systems we considered including in the comparative analysis were either unavailable for download, or very restricted in the span of the scientific domain represented.

Before grading, we discarded data from four participants. One did not understand the task, and wrote a (nice) essay about reinforcement learning. The others, despite visiting many web pages, listed less than 5 papers when time ran out.

We had an expert judge evaluate the results of the rest of the participants. We combined all of the papers that users had entered into one list. Each entry includes the paper's information and URL. In addition, we listed the labels that the users supplied for each paper. The judge did not know the method used to find the papers.

¹<http://scholar.google.com>

Our expert judge scored the papers on a 3-point scale: 0 – Irrelevant, 1 – 1: Relevant, 2 – Seminal. Each label was given a 0-1 score, based on whether it was a good match to the paper. The results are summarized below.

5.3.1 Results and Discussion

Information collection patterns

Avg:	Pages visited	Papers listed	Visited/Listed
GS	46	12.2	4.51
MP+GS	36.3	9.75	3.79

Table 5.1: Information collection patterns.

The table shows the average number of web pages visited throughout the session, the average number of papers listed by the user, and the average ratio of pages visited to papers listed. GS users visited more pages and listed more papers on average. However, when looking at the average ratio, only one out of 4.5 pages visited by GS users was added to their list, while MP+GS added one out of 3.8. In other words, the map users were more focused: they may have visited less pages, but they found these pages satisfactory.

Precision

Users’ satisfaction level is important, but the real test is the expert’s opinion. The next table shows the average normalized scores given by the judge: For each user, we calculate the average paper score and average label score. Then, we average over the users in each condition:

Avg:	Normalized Score	Normalized Label Score
GS	74.2%	71.6%
MP+GS	84.5%	80.2%

Table 5.2: Precision scores: paper score and label score.

Both the paper and label scores of MP+GS users are higher than the scores of GS users (the median scores exhibit similar behaviour). In addition, the average number of seminal papers discovered by GS users was 1.2 , while MP+GS users have discovered on average 1.62 seminal papers.

The simulated Wikipedia user WK did not do well: out of 15 references, only four qualified for the study (papers published after 1996), and only two were deemed relevant. In Wikipedia’s defense, the other references included seminal books, which could have been useful for our hypothetical first-year student.

Finally, let us examine the map (MP) user performance. Comparing the map directly to user output is challenging as the map contained 45 papers, many more than the average user. Out of these papers, seven were deemed seminal, and 21 were deemed relevant. Interestingly, many of

the papers that were deemed irrelevant were used as bridges between relevant (or seminal) papers in the map.

The finding that many of the map users did not identify the seminal papers in the map is somewhat concerning. A possible explanation may be that the users were instructed to focus on at most five lines of research, while the seminal papers were spread among more lines. Note that despite this fact, the average normalized score of MP+GS users is still higher than the score for the map. In any case, this phenomenon highlights the need for more targeted research on locating and visualizing important nodes in the map.

Recall

In addition to measuring precision (the fraction of retrieved papers that are relevant), we also tested user's *recall* (the fraction of relevant papers retrieved). It is not enough for the users to find good papers; rather, it is also important that they do not overlook important research areas.

In order to measure recall, we have composed a list of the top-10 subareas of reinforcement learning by going over conference and workshop tracks and picking the most frequent topics. Each user had to list up to five research directions; for each user, we computed the fraction of these directions that appeared in our top-10 list. GS users received an average score of 46.4%, while MP+GS users outperformed them with an average score of 73.1%.

Finally, further analysis of the snapshots taken throughout the study provides anecdotal evidence of the utility of the map. Several MP+GS users started by composing a short list of research directions; throughout the session, these users have progressively added papers to each direction. GS users, in contrast, did not exhibit this 'big picture' behaviour.

5.3.2 User Comments

After the study, we asked the map users to tell us about their experience. Below are some of their comments:

Most importantly, many participants found the map useful in making sense of the field. Some of the participants had trouble interpreting elements of the map, or felt like the map was more suited for researchers with deeper background knowledge. We found that many of the negative comments could be addressed by improvements in the design of the user interface.

5.4 Summary

In this chapter, we applied the ideas of Chapter 3 to the scientific domain. We exploited the additional structure available to us in the form of the citation graph to re-define coherence, coverage and connectivity.

In particular, we (1) characterized the probability that ideas in two papers stem from a common source, (2) quantified the impact of one paper on the corpus, and (3) proposed a notion of connectivity that captures how different lines of research can still interact with each other, despite not intersecting.

We conducted validation studies with users that highlight the promise of the methodology. Map users found better papers and covered more important areas than users of popular competitors.

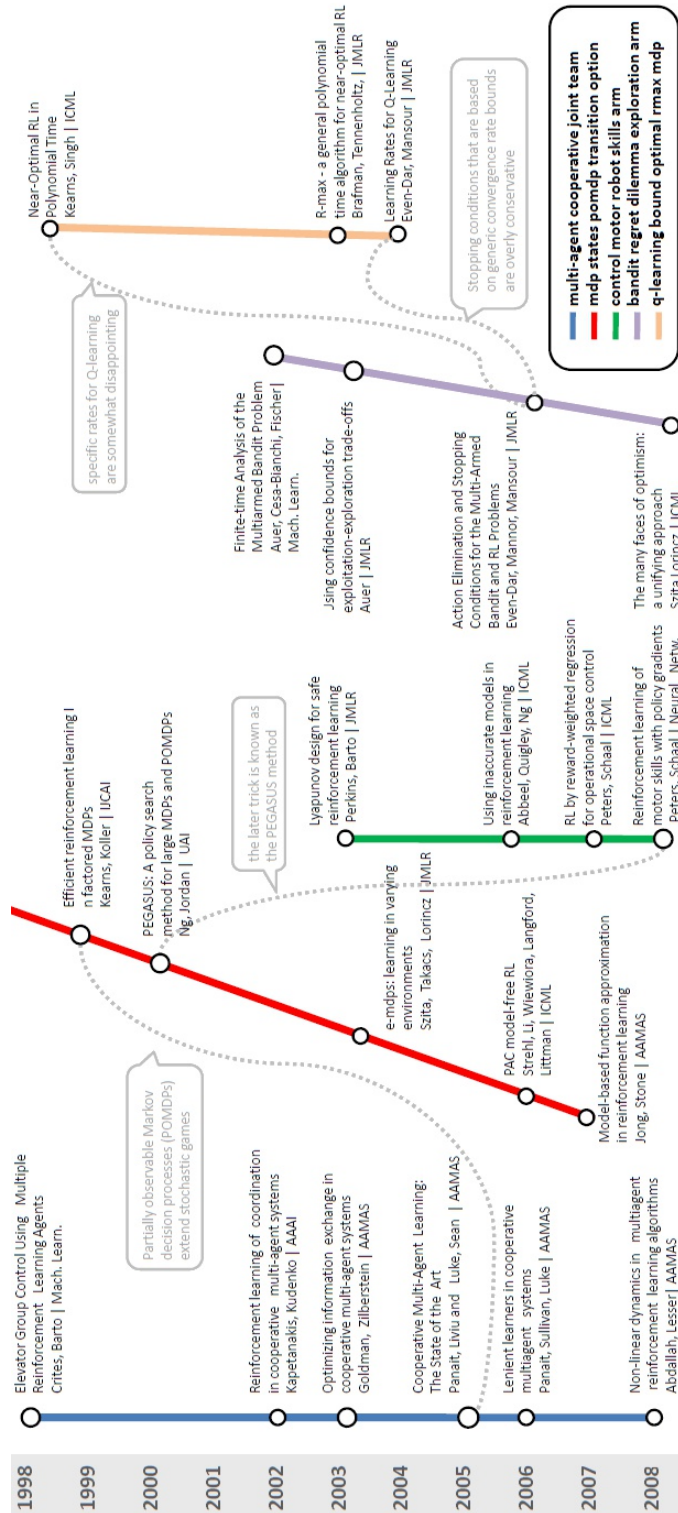


Figure 5.7: Part of the map computed for the query ‘Reinforcement Learning’. The map depicts multiple lines of research (see legend at the bottom). Interactions between the lines are depicted as dashed gray lines, and relevant citation text appears near them.

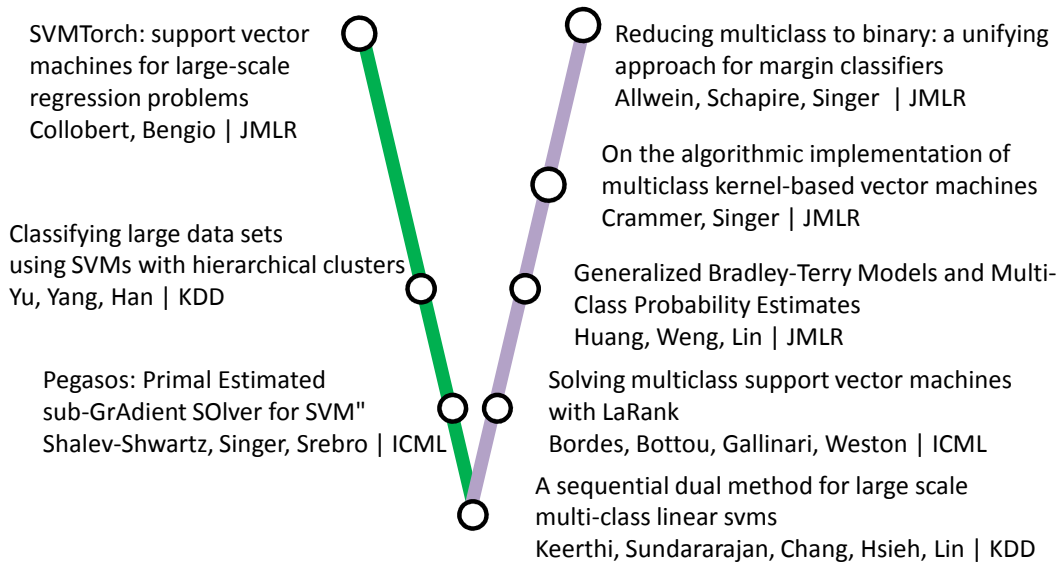


Figure 5.8: A segment of a map computed for the query SVM/ Support vector machine, showing the intersection of two lines: multi-class SVMs and large-scale SVM. In the interest of space, we condensed the time line.

Positive: “Helpful... gave me keywords to search for” / “I noticed directions I didn’t know about... Haven’t heard of predictive state representations before” / “Useful way to get a basic idea of what science is up to” / “That was a great starting point” / “Easy to identify research groups... in this context, this guy is good” / “Timeline is very useful”

Negative: “Takes a while to grasp” / “For a beginner, some papers are too specific... may be more useful after I read some more” / “Legend is confusing if you do not know the topic in advance” / “Didn’t necessarily understand the logic behind edges... why don’t you draw words on edges?” / “It is hard to get an idea from paper title alone”

Chapter 6

Implementation and Analysis



“Notice all the computations, theoretical scribbles, and lab equipment, Norm.... Yes, curiosity killed these cats.”

In this chapter we discuss practical implementation issues. We provide a high-level view of our algorithm, analyze its complexity and propose ways to speed up the computation. The main goal of this chapter is to assist those interested in building similar systems; others may well skip to the next chapter.

6.1 Algorithm Overview

We start by discussing our algorithm (Algorithm 4). The algorithm has three main phases: (1) preparations, where it gathers all the input needed for map computation, (2) the map computation itself, and (3) displaying the map to the user. In the following, we briefly discuss each of them.

Algorithm 4: MetroMaps

```
// Preparations
1 Obtain the input set  $\mathcal{D}$ 
2 Obtain external information about  $\mathcal{D}$  (e.g., citation graph)
3 Obtain coverage elements  $\mathcal{E}$ 
4 Obtain weights  $\lambda_e$  for each element  $e \in \mathcal{E}$ 
5 Obtain  $cover_d(e)$  for each  $d \in \mathcal{D}, e \in \mathcal{E}$ 
6 Prune  $\mathcal{D}$ , if necessary
7 Obtain features  $\mathcal{W}$  from  $\mathcal{D}$ 
8 Obtain a transition restriction graph  $G_{tr}$ 
9 Obtain  $influence(d_i, d_j | w)$ 
10 Prune  $G_{tr}$ , if necessary

// Compute the Map
11 Compute coherence graph  $\mathbb{G}$ 
12 Extract a set of top-coverage chains from  $\mathbb{G}$ 
13 Local search to improve connectivity

// Present Map to User
14 Compute line labels
15 Compute  $(x, y)$  coordinates for every article
16 Draw!
```

Preparations In the preparations stage, the algorithm gathers all the input needed for map computation.

The algorithm starts by obtaining a set of documents \mathcal{D} to summarize (Line 1). \mathcal{D} could be specified explicitly or computed from a query (applying standard IR methods, such as query expansion). In addition to \mathcal{D} , we obtain all the external information needed to compute the map later (for example, the citation graph).

In Lines 3-5 we gather the pieces needed to compute coverage: element weights and single-document coverage functions. In Lines 7-9 we prepare the elements needed for the computation of coherence, namely influence functions and the valid transition graph G_{tr} .

Constructing the Map The map construction stage (Lines 11-13) is the heart of the algorithm.

In Line 11, the algorithm builds a coherence graph. The graph encodes all coherent chains over the documents \mathcal{D} . In Line 12 we extract a set of high-coverage chains from the graph. The number of chains K may be specified in advance, or (as some stories are more complex than others) one may prefer to add lines until coverage gains fall below a threshold. Finally, in Line 13 we perform local search to increase the map connectivity.

Presenting the Map to the User In the last phase of Algorithm 4, we display the map to the user. In order to facilitate navigation in the map, we have added a *legend* feature. We assign a few characteristic words to each line (see Figure 3.4). The words chosen to describe each line are words carrying the highest incremental coverage for that line, assuming the other lines have been read. In other words, for each line π_i we found the words maximizing

$$cover_{\cup_j \pi_j}(w) - cover_{\cup_{j \neq i} \pi_j}(w)$$

Since the other lines may contain a large number of documents, coverage of some important elements may saturate. Therefore, we adjust the sampling procedure of Section 4.2.2 to sample from only l of the documents of the other lines.

Next, we assign (x, y) coordinates to each paper. We have identified several design principles common to metro maps; based on those principles, we designed the map layout algorithm.

In order to minimize the number of kinks and turns in a line, we break each line into segments. A segment is a maximal sub-line that does not intersect with other lines. For each segment, we represent each segment by its two endpoints, and run the layout algorithm on the *segment graph*. Once we find a layout, we add the rest of the nodes along the segment. This way, the only kinks in the graph are around intersections.

We decided to use one of the axes as a temporal axis, and set the other coordinate according to a force-directed layout algorithm. The algorithm assigns forces among edges as if they were springs, and simulates the graph like a physical system. We have modified the algorithm so that the graph respects chronological order among vertices. The output of force-directed algorithms is often aesthetically pleasing, and there are few unnecessary crossing edges.

6.2 Speeding Up the Computation

We have reviewed the three main phases of Algorithm 4. As our goal is to build interactive systems, we need to ensure that the algorithm can answer a query within reasonable time. In this section we analyze the computational cost of the algorithm, and propose methods to speed up our bottlenecks.

Let us take a closer look at the three phases of the algorithm. The preparations phase can be performed offline, during preprocessing of the data. The visualization phase normally operates

on tiny graphs, and thus is very fast. Therefore, we will focus our attention on speeding up the **map construction** phase.

6.2.1 Complexity of Map Construction

Before proposing methods for speeding up the map construction phase, let us analyze its complexity. The map construction phase is composed of the following steps:

1. In the first step, we apply generalized best-first search to compute the vertices of the coherence graph (Section 3.2.1). In the worst case, the search procedure might solve $O(|\mathcal{D}|^m)$ linear programs (all possible chains of length m). For such a graph, the number of edges may be as high as $O(|\mathcal{D}|^{2m})$.
2. The coverage step (Section 3.2.2) requires K iterations of the greedy algorithm – one per each metro line. Each iteration requires solving $O(|\mathcal{D}|^2)$ orienteering problems using the polynomial algorithm of Chekuri and Pal [2005].
3. The connectivity step (Section 3.2.3) performs local search. Each iteration requires in the worst case solving another $O(K|\mathcal{D}|^2)$ orienteering problems.

The three steps are computationally intensive. In the following, we propose practical methods to overcome the computational hurdles.

6.2.2 Scaling Up the Coherence Graph

In Line 11 the algorithm constructs a coherence graph using a *generalized best-first* (GBF) search strategy: at each iteration, we expand the node corresponding to the highest-coherence chain, generate all of its extensions and evaluate their coherence as well (see Chapter 3 for more details).

The GBF strategy may not be practical when examining all $O(|\mathcal{D}|^m)$ chains of length m . From the description above, we can think of two ways to speed the search up: restricting the search space, or reducing the amount of work per expanded node.

We start by outlining two ways to restrict the search space of GBF.

Restricting the Document Set: In Line 11, the algorithm finds all coherent chains over the documents of \mathcal{D} . However, many of these chains are not likely to ever participate in a map. The choice of chains for the map is based on their *coverage*; chains with low coverage are rarely selected.

Therefore, it may be possible to prune low-coverage documents in advance, as they are not likely to be chosen for the map (Line 6). In the scientific domain, for example, this heuristic translates to pruning low-impact papers. Note that the submodularity of our coverage function guarantees that if a document has low coverage, its coverage can only decrease as new papers are added to the map. Therefore, low-coverage papers are guaranteed to remain low-coverage. However, we must keep in mind that low-coverage papers may still be useful: in particular, low-coverage papers sometimes act as the “missing link” between two high-coverage papers when forming coherent chains.

Another type of papers that can be pruned is near-duplicate papers. For example, in the news domain, multiple sources may report the same event. These reports all provide low incremental

coverage w.r.t. each other, and can be used interchangeably in chains. Therefore, we do not need all of them when constructing the coherence graph.

Importantly, the pruned documents may not be used for the coherence graph, but they are still taken into consideration when evaluating coverage in later steps. In addition, note that if we use personalized coverage we may have to re-adjust our document selection when the user provides feedback.

Restricting Possible Transitions: In Line 8 of Algorithm 4 we obtain G_{tr} , a graph restricting the transitions that can participate in chains. Restricting the transitions can significantly speed up GBF; in addition, this approach leads to a natural anytime algorithm, as one can start from a sparse G_{tr} and incrementally add edges.

In the simplest case, G_{tr} is a directed clique; all transitions obeying chronological order are possible. However, G_{tr} can impose further constraints, such as minimal/maximal time difference between the articles. As another example, we may remove all edges whose best possible coherence is below some threshold. In our experiments, we chose to restrict the *degree* of G_{tr} : clustering algorithms can be used to determine the closest d neighbours of each vertex, and include only these edges in G_{tr} .

As a side note, notice that the choice of restrictions may depend on the query (Line 10). For example, setting minimal/maximal time difference between consecutive articles may depend on the timespan of the documents of \mathcal{D} .

We proposed two ways to restrict the search space of our generalized best-first search procedure. However, the amount of work per node may still be high, as evaluating each chain requires an external LP solver. We now propose an alternative method for scoring chains.

Scoring Simple Chains: Note that the CTD coherence notion was created with long chains in mind. However, for the coherence graph we only need to concentrate on chains of length m . We can take advantage of the fact that shorter chains are *simpler*: articles are often more related to each other, and there is less topic drift. In terms of Section 3.1.1, selected words are active throughout the entire chain ($kTotal=kTrans$). This makes it easier to identify the active words; for example, given a chain to score, we can pick the words with the highest median (or average) value and activate them throughout the entire chain. This way, no LP solver is needed, and scoring a chain is fast. While this method is not guaranteed to find the same words as the optimum, it achieves good results in practice.

The idea that shorter chains are *simple* gives rise to new possibilities. If short coherent chains consist of highly-related articles, perhaps we can do better than blindly traverse the search space:

Bottleneck Paths Suppose we knew a set of words that are guaranteed to be the active words in some unknown short, coherent chain; our goal is to recover the corresponding chain. In this case, we can take all possible transitions and score them based on the given set of words. The result is a weighted directed graph G , whose vertices correspond to documents \mathcal{D} .

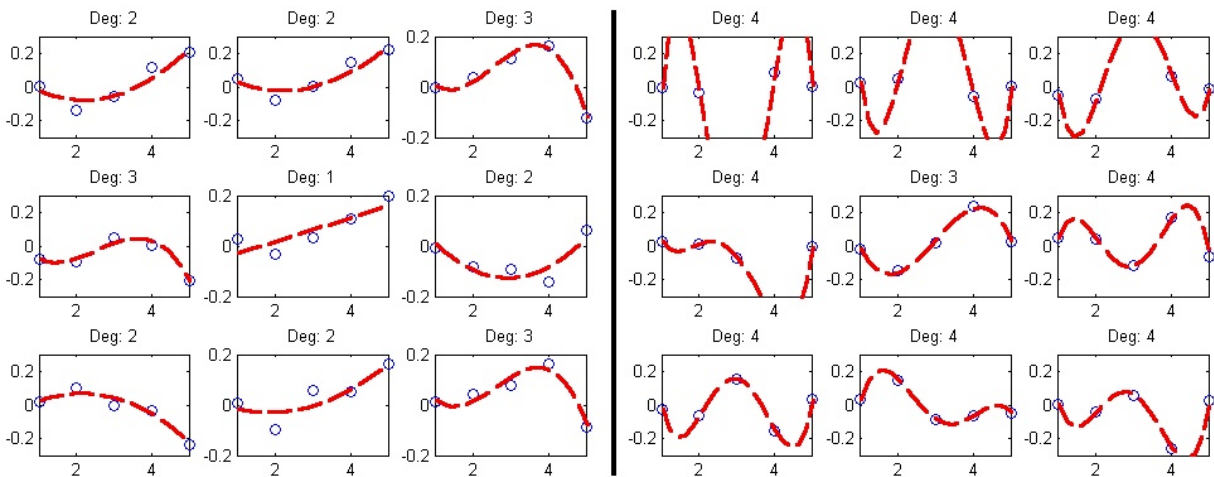


Figure 6.1: Behaviour of the top-9 features for Chain A (left), and A' (see in text, right). x axis represents document position in the chain, y axis represents PCA feature value. Circles correspond to the actual feature values in the chain, and dashed lines are the lowest-degree polynomial which fits these points within a specified error. The degree of each polynomial appears above it.

A path p in G corresponds to a chain. Importantly, the coherence of the chain (with respect to the chosen words) is the minimum edge weight across p , also known as its *bottleneck*. The problem of finding the top coherent chain of length m directly translates to a variant of the *bottleneck path* problem: find the top bottleneck path of length m . This problem is easily solvable (polynomial time) via dynamic programming techniques; we can also generalize the solution to find the top- k bottleneck paths.

We are still left with the problem of identifying good sets of active words. The number of candidate sets is exponential in \mathcal{W} , and thus enumeration is infeasible. We can take advantage of topic models (e.g., [Blei et al., 2003]) to identify salient words of different topics; alternatively, we could choose pairs of documents (especially similar ones) and identify their top common words.

Word Trends As mentioned above, short coherent chains do not suffer from a lot of topic drift. Given a chain, the optimal active words are often easy to identify. We now propose another way to take advantage of this fact and identify potentially good chains without using GBF.

We examined the behaviour of active words throughout various chains. We observe that in many good chains, active words exhibit a smooth behaviour: words become progressively more (or less) important as the chain advances, or stay relatively stable. In poor chains, active words tend to fluctuate more.

In order to formalize this, we look at the series of coefficients of each word throughout the chain, and try to express it as a low-degree polynomial. Figure 6.1 demonstrates this approach. We tested two chains: chain A is a coherent chain about the debt crisis, and Chain A', which was obtained from Chain A by replacing an intermediate article by an article about Hungary and the Greek debt. Intuitively, this change made Chain A' less coherent.

Figure 6.1 shows the behaviour of the top 9 features for Chain A (left), and A' (right). As

words were too noisy, we used PCA components instead. x axis represents document position in the chain, y axis represents (PCA) feature value. Circles correspond to the actual values, and dashed lines are the lowest-degree polynomial which fits them within a specified error. The degree of each polynomial appears above it (note that one can always fit a polynomial of degree 4 through 5 points). As expected, Chain A needs lower-degree polynomials than Chain A'.

We have experimented with low-degree polynomials as a measure of chain quality. Chains displaying low-degree behaviour were usually coherent, but some of our hand-picked coherent chains required a high degree. That is, the process seems biased towards false negatives. However, according to our observations, this bias does not pose a problem when \mathcal{D} is large enough. If a coherent chain did not score well, there was usually a similar chain which scored better.

Algorithm 5: FindShortChains(\mathcal{D}, m)

input : \mathcal{D} a set of documents, m desired length
output: A set of chains of length m .

```

1 for  $K$  iterations do
2   Randomly select  $(d, d') \in \mathcal{D}^2$  ;
   // Create a model of a chain between  $d$  and  $d'$ 
3   foreach  $(w) \in \text{importantFeatures}(\{d, d'\})$  do
4      $\lfloor$   $Model_{d,d'}(w) = \text{polyfit}((1, d(w)), (m, d'(w)))$  ;
   // Evaluate other documents
5   Initialize array  $FitDocs$  to  $\emptyset$ ;
6   foreach  $d'' \in \mathcal{D} \setminus \{d, d'\}$  do
7     // Find best position for  $d''$  (null if far)
8      $i'' = \text{bestPos}(d'', Model_{d,d'})$  ;
9     if  $i'' \neq \text{null}$  then
10     $\lfloor$   $FitDocs[i''] = FitDocs[i''] \cup \{d''\}$  ;
11    $score = \text{getScore}(FitDocs)$  ;
12   Record best model as  $Model^*$ , and its  $FitDocs$  array as  $FitDocs^*$ ;
13 if  $Model^*$  has a low score then return ;
   // Good model found. Reestimate from fitting docs
14 foreach  $(w) \in \text{importantFeatures}(FitDocs^*)$  do
15    $\lfloor$   $Model_{new}^*(w) = \text{polyfit}(FitDocs^*(w), degree)$  ;
16 return  $\text{extractChains}(FitDocs, Model_{new}^*)$  ;
```

The observations above motivated us to seek an algorithm for finding short chains with smooth word behaviour. We are inspired by RANSAC [Fischler and Bolles, 1981], a method originating from the statistics literature (and made popular by the vision community). RANSAC is a general parameter estimation approach designed to cope with a large proportion of outliers in the input data. In a nutshell, it is a resampling technique, generating candidate solutions by using few datapoints to estimate the underlying model parameters.

Our algorithm is described in Algorithm 5. In each iteration, we randomly select a set of candidate pairs of articles, $\{(d, d')\}$, to be used as endpoints of the chain (Line 2). We then

hypothesize a *model* for a coherent chain linking d and d' (Line 4). A model is a sequence of predicted values for each feature. For example, if d and d' both display high levels of feature w , we expect the rest of the documents in any coherent chain to display similar high levels. If d displays higher levels of w , we expect to observe this trend in the rest of the chain. Since we only have two points to estimate the model from, we simply fit a linear function between them.

Next, we want to evaluate our model. Intuitively, a model is good if we can find many coherent chains that behave similarly to the model’s prediction. In order to find such chains, we first try to locate *articles* that closely fit the model’s prediction. For each article d'' , we find the best position in the chain: i.e., the position that minimizes d'' ’s distance from the model (Line 7, function `bestPos`). If d'' is close enough to the model, this position is recorded in the *FitDocs* array.

After finding a set of documents that closely fit the model’s prediction *FitDocs*, we want to compute the number of chains that can be generated from them. It is important to note that the number of documents in *FitDocs* is not enough: for example, if many documents fit the model’s prediction for position 2 but no documents could be found for position 3, no chains match the model.

If there are no chronological constraints, the number of chains is simply $\prod_i |FitDocs[i]|$: there are $|FitDocs[i]|$ options for the i th position. Otherwise, we construct a directed acyclic graph corresponding to chronological constraints: we create a layer of vertices for each position i , and add edges between a vertex at layer i and a vertex at layer $i + 1$ only if the document at layer $i + 1$ is newer. We add d as a source and d' as a sink. Thus, each path from d to d' corresponds to a valid chain, and we are left with the task of counting the paths. Since the graph is a DAG, we can apply a linear-time algorithm to count the number of possible paths.

The score of the model is the number of coherent paths it captures. We repeat the process for multiple candidate pairs d, d' , and then pick the best model; i.e., the model that predicted a large number of chains. Since the model was estimated only from the initial two articles, we re-estimate it from all of the fitting articles *FitDocs* (Line 14). Finally, we extract short chains that are a close fit to the re-estimated model.

Our algorithm is not guaranteed to succeed, since it may not draw documents that capture a good model. However, since many document pairs do encode a good model, the algorithm works well in practice. It is also fast and easy to parallelize. In addition, the algorithm provides an interesting interpretation of m -coherence: One can think of a chain as a ride through feature-space. Each sub-chain has a smooth trajectory, when projected on important-feature axis. Because of the large overlap between sub-chains, we are only allowed gentle adjustments to the steering wheel as we progress throughout the chain.

As expected, the algorithm tends to recover topics that are heavily represented in the dataset; topics that are poorly represented are less likely to be sampled. Nevertheless, the chains recovered are of comparable quality to the chains recovered by methods of Section 3.2.1.

A Note on Data Structures

In the previous sections we have proposed ways to speed up the *computation* of the coherence graph. Let us now consider the *representation* of the computed graph.

Note that in the worst case, the number of vertices of the coherence graph could be $O(|\mathcal{D}|^m)$. For such a graph, the number of edges may be as high as $O(|\mathcal{D}|^{2m})$.

However, because of the way edges are defined, we can store the graph in time and space much smaller than $O(|\mathcal{D}|^{2m})$. We can create bins for each possible prefix and suffix of length $m - 1$, and hash each vertex to the two bins corresponding to its prefix and suffix. Since there is an edge between two vertices iff one’s suffix is identical to the other’s prefix, we do not need to explicitly represent the edges. Thus, we can represent the graph using the prefix and suffix hash tables alone, taking only $O(|\mathcal{D}|^m)$ in expectation.

6.2.3 Scaling Up Coverage and Connectivity

After discussing scaling up Line 11, we turn our attention to Lines 12 and 13. Both lines are very similar in nature; the computational bottleneck in both lines is the need to solve many orienteering problems. However, two factors ameliorate this concern.

First, evaluating chains is easy to parallelize, since there is no dependency between different chains (a.k.a. “embarrassingly parallel”). Therefore, we can speed the algorithm up simply by adding more processors.

Second, we note that we do not have to re-evaluate all candidate chains at every stage of the greedy algorithm. Rather, we can evaluate the candidate chains lazily. The correctness of this lazy procedure follows directly from submodularity [Leskovec et al., 2007], and leads to far fewer (expensive) evaluations of chains. We can also take advantage of independence again and evaluate several of the top chains in parallel. Lazy evaluations lead to dramatic speedups in both Lines 12 and 13, without losing approximation bounds.

6.3 Running Times

Below we measure the running time of our algorithm for a few sample queries. The algorithm is implemented in matlab on a single core machine; we believe that parallelizing the algorithm (as discussed above) and re-implementing it in a more efficient language will result in reasonable running time.

	Number of Papers	Running Time (s)
MDP	155	94
Reinforcement Learning	438	325
SVM	1348	1907
Information Retrieval	1705	2071

Table 6.1: Running times for MetroMaps.

Note that while the size of the query set \mathcal{D} is important, it is not the only factor determining the running time; the structural properties of \mathcal{D} (and in particular, the number of coherent paths) play an important role as well.

User queries can range from broad (‘Science’) to narrow (‘Correlated Q-Learning’). Our current implementation can handle queries of several thousands of documents at most; broader

queries (for example, all computer science papers) are not feasible. However, note that we restrict the size of the maps in order to avoid clutter in the visualization. Maps usually include 5-6 lines, and 20-30 papers; in other words, the result of a “computer science” query, even if we could compute it efficiently, is unlikely to cover a large fraction of the corpus. The Chilean miners story, on the other hand, was almost linear, and thus could be effectively covered by a map (note that this property would not change even if we had ten times more articles about the miners). Until we have hierarchical maps with cluster-nodes and zooming capabilities, it is best to keep the map size constraints in mind when formulating queries.

Chapter 7

Related Work



Laboratory peer pressure

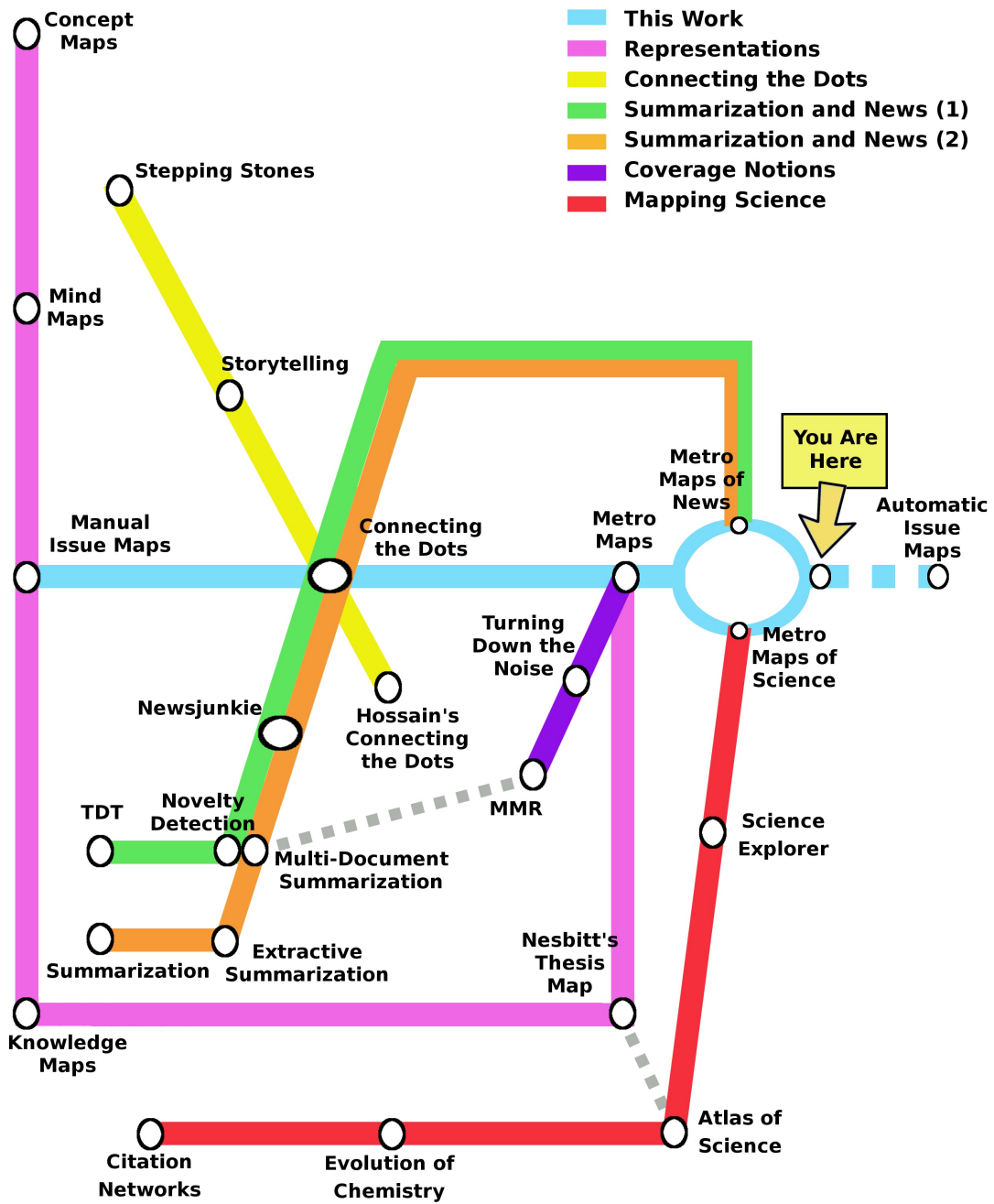


Figure 7.1: Related Work. The light-blue line details the evolution of this work. The red line revolves around visual representations for science, while the pink line focuses on map representations. The green and orange lines concentrate on methods for summarizing collections of documents, especially in the news domain. Variations of “Connecting the Dots” problem appear along the yellow line. Finally, notions of coverage are covered by the purple line.

To the best of our knowledge, the problem of automatically constructing metro maps is novel. Nevertheless, there has been extensive work done on a myriad of related directions. Figure 7.1 shows a metro map outlining some of these directions.

The light-blue line details the evolution of this work: from manual issue maps, through connecting the dots, to metro maps and our two test cases. In the following, we discuss the rest of the map.

7.1 Visual Metaphors

Consider Figure 7.1 again. The pink line concentrates on visual representations of data, in particular ones employing a map metaphor; many of these representations have been presented in Section 1.1.

Concept maps [Novak, 1990] are a graphical representation where nodes represent concepts, and edges represent the relationships between concepts. The central concept is at the top of the map and the various subcomponents appear further down the map. Figure 1.2(left) shows an example of concept map answering the question “Why do we have seasons?”.

Knowledge maps [O’Donnell et al., 2002] are similar to concept maps, but their edges are labeled by a fixed vocabulary of symbols. The fixed vocabulary may be domain specific, and is often presented in an abbreviated format (“P” for “part”, “C” for “characteristic”). Information in nodes can be larger than a concept (e.g., a paragraph), and the maps themselves are often less hierarchical than concept maps.

Issue maps (or argument maps) display the structure of an argument. They includes components such as a main contention, premises, co-premises, objections, rebuttals and lemmas. Typically an issue map is a directed graph, with nodes corresponding to propositions and edges corresponding to relationships (e.g., dispute or support). Figure 1.3 shows an example of an issue map dealing with perhaps the oldest AI question: can computers think?

Mind maps [Buzan and Buzan, 1995] are another popular representation. Mind maps are usually radial, with branches flowing from the center of the map. Color and pictures are very emphasized. See, for example, a map about personal health in Figure 1.2 (right).

The final representation on the pink line is Nesbitt’s metro map drawing. The drawing portrays the interconnecting ideas running through Nesbitt’s doctoral dissertation [Nesbitt, 2004]. Note Nesbitt’s map is connected to the red line, as the map appears in the Atlas of Science [Borner, 2010].

While the objectives of mapping tools are usually similar, they are used for different purposes. Mind maps focus on imagination and exploring associations between concepts. Concept maps aim at explaining the relationships between concepts, and hence understanding the concepts themselves. Issue maps are meant to expose the structure of an argument, and allow users to evaluate them the soundness of argument premises and the inferential connections. For a comparison between concept maps, mind maps and other visual representations, see [Eppler, 2006].

Our decision to focus on maps is there also motivated by the strong empirical support for map representations in enhancing, retaining and improving knowledge. The meta-analysis of [Nesbit and Adesope, 2006] found that mapping activities are more effective for attaining knowledge

retention and transfer, in comparison with activities such as reading text, attending lectures, and participating in class discussions. The benefits held across a broad range of educational levels, subject areas, and settings. Much of this benefit may be due to greater learner engagement occasioned by maps.

Other studies found that Mind maps were shown to increase memory recall for undergraduate students when compared to traditional note taking [Farrand et al., 2002], and also to reduce cognitive load [Sarker et al., 2008]. Rewey et al. [1991] showed that students recall more central ideas when they learn from a knowledge map than when they learn from (informationally isomorphic) text. The benefits of maps are often higher for students with low prior knowledge.

Most importantly, all of the representations mentioned above are *manually* constructed, whereas our metro maps are constructed automatically.

7.2 Connecting the Dots

The Connecting the Dots problem appeared in the literature in a variety of forms and domains.

Kumar et al. [2006] formulate a new data mining problem called *storytelling* as a generalization of redescription mining. Storytelling aims to explicitly relate object sets that are disjoint by finding a chain of approximate redescriptions between the sets. For example, if some London travel books (Y) overlap with books about places where popes are interred (G), and some of which are books about ancient codes (R), then the sequence of approximate redescriptions $Y \Leftrightarrow G \Leftrightarrow R$ is a story.

The strength of a story is determined by the weakest transition. The strength of a transition measured by its Jaccard coefficient (the ratio of the size of common elements to elements on either side of the redescription). Since this is a local measure, any global notion of coherence is lost.

In [Hossain et al., 2010], the goal is to find hammock paths, which are a generalization of traditional paths: A hammock is a pair of objects which share common features. A hammock path is a sequence of objects, such that each two consecutive ones form a hammock.

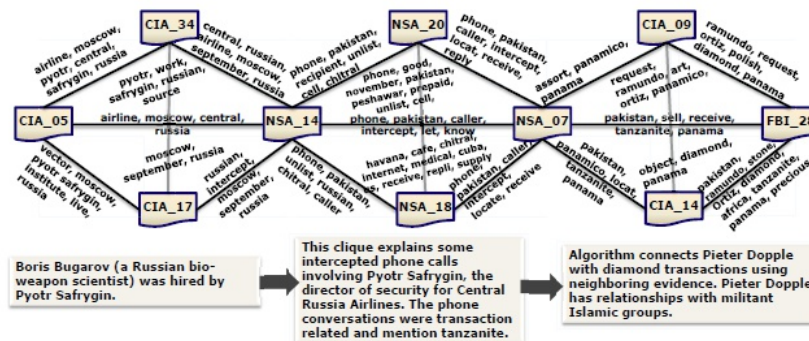


Figure 7.2: Helping intelligence analysts make connections via chains of cliques [Hossain et al., 2011].

In [Hossain et al., 2011], Hossain proposes a system for helping intelligence analysts make connections. The system constructs a chain of documents; it takes into consideration two parameters: (1) A distance threshold, capturing the maximum acceptable distance between two

neighboring articles, and (2) a minimum clique size threshold, capturing the minimum size of a clique between two neighboring articles (Figure 7.2). Given a document, the system uses the distance threshold and clique size requirements to identify a set of possible successors to be used in the chain. Note that the process of finding a successor does *not* take the end document into consideration.

Connecting the Dots has also been explored in non-textual domains. The authors of [Heath et al., 2010] propose building graphs, called Image Webs, to represent the connections between images in a collection, and discover meaningful paths in them. Images may depict the same static scene, or they may be related in a more subtle way (for example, two different buildings at a university are connected by a campus bus that frequently stops near each building). The authors’ main tool for path discovery was shortest-path, and their chains may exhibit stream-of-consciousness behaviour, similar to the shortest-path chains of Section 2.1; in contrast, our notion of coherence is global.

In [Wang et al., 2012] the authors generate pictorial storylines for given topics from text and image data. The system constructs a multi-view object graph, and selects a set of nodes using an approximation algorithm for the Minimum-Weight Dominating Set Problem. Finally, it creates a storyline using a directed Steiner tree algorithm. Steiner trees, similar to shortest-path, do not guarantee a global theme of the storyline.

Several methods have concentrated on connecting two points of interest via *multiple* paths. In [Faloutsos et al., 2004], the goal is to extract a small (amenable to visual inspection) subgraph that best captures the connections between two nodes of the graph. The authors interpret the graph as an electrical network, and choose the subgraph that can deliver as much electrical current as possible. See Figure 7.3 for a connection graph between Alan Turing and Sharon Stone. One connection is through the actress Kate Winslet, because she starred in a movie about the Enigma cipher machine. Note that edges in the graph are computed locally, and paths in them are not necessarily coherent.

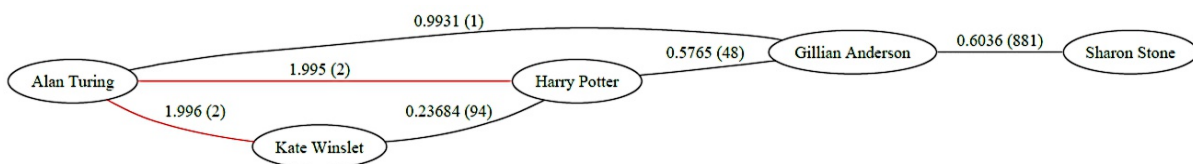


Figure 7.3: A connection graph between Alan Turing and Sharon Stone [Faloutsos et al., 2004].

Das-Neves et al. [2005]; Fox et al. [2006] propose a system that takes an input two related but separable topics. The goal is to retrieve one or more sequences of documents that support a valid set of relationships between the two topics. The output is a connected network of chains of evidence. Each chain is made of a sequence of additional topics (stepping stones); each topic in the sequence is logically connected to the next and previous one. Together, the chains provide a rationale (a pathway) for the connection between the two original topics.

For example, if the query is “distributed systems” and “file system”, middle nodes connecting these endpoints could be “distributed applications” and “high performance”.

Both electrical networks and stepping stones offer some insight regarding the connection between their endpoints. However, compared to metro map chains, both representations lack

structure. In addition, we believe that the choice of a single concept (or entity) as a unit of analysis is too fine to be used for explaining complex topics.

In all of the methods mentioned above, there is no *global* notion of coherence. The neighbouring items are chosen because their similarity passed some threshold, or they belong to a spanning tree. We believe that the notion of coherent paths facilitates the process of knowledge acquisition for the users.

On a related note, the *narrative generation* community [Turner, 1994; Niehaus and Young, 2009] has sought to explore the notion of narratives and the ways to model them. In particular, what makes narrative different from a list of events? However, their task seems to be fundamentally different from ours. Much of the work involves producing natural-language experiences for a user (e.g., a computer game), and focus on planning-like operators and inferences. In contrast, we do not try to generate any natural-language content, neither do we make up new plots. Nevertheless, some of the work done on evaluating narratives [Rowe et al., 2009] may be useful for our purposes.

7.3 Summarization and News

An active area of research focuses on summarizing collections of documents, especially in the news domain. Consider the green and orange lines in Figure 7.1.

The green line starts with Topic Detection and Tracking (TDT) [Allan et al., 1998]. TDT is an initiative aiming at finding and following new events in unsegmented streams of news reporting. TDT traditionally considered topics as flat clusters, but has since moved on to hierarchical structures [Allan et al., 2003] that can explicitly model dependencies between events [Nallapati et al., 2004].

TDT has led to investigations of a variety of problems related to novelty detection, in particular First Story Detection (FSD) [Allan et al., 2000]. In FSD, the task is to mark each story as “first” if it is the first to discuss an event. An event is something that happens at some specific time and place, e.g., an earthquake.

Similarly, Kleinberg [Kleinberg, 2002] used randomized infinite-state automata to model burstiness and hierarchical structure in text streams. Unlike most of the TDT work, Kleinberg’s work did not focus on news. Rather, it explored email archives and other time-stamped document collections.

Assessing the novelty of a text is closely related to finding its most informative sentences. The latter task is called *extractive* text summarization (as opposed to *abstractive* summarization methods, which attempt to rephrase the information in the text). Recently, there has been a lot of research dealing with *multi-document* extractive summarization, where the goal is to extract summarized information from multiple texts about the same topic [Radev et al., 2002].

Several news summarization systems that build upon the ideas above have been proposed. The Columbia Newsblaster project [Evans et al., 2004] applies multiple summarization systems to the texts. Newsjunkie [Gabrilovich et al., 2004] identifies the novelty of stories and custom-tailors newsfeeds to a user, based on the information that they have already reviewed (Figure 7.4). Other timeline systems include [Swan and Jensen, 2000; Yan et al., 2011].

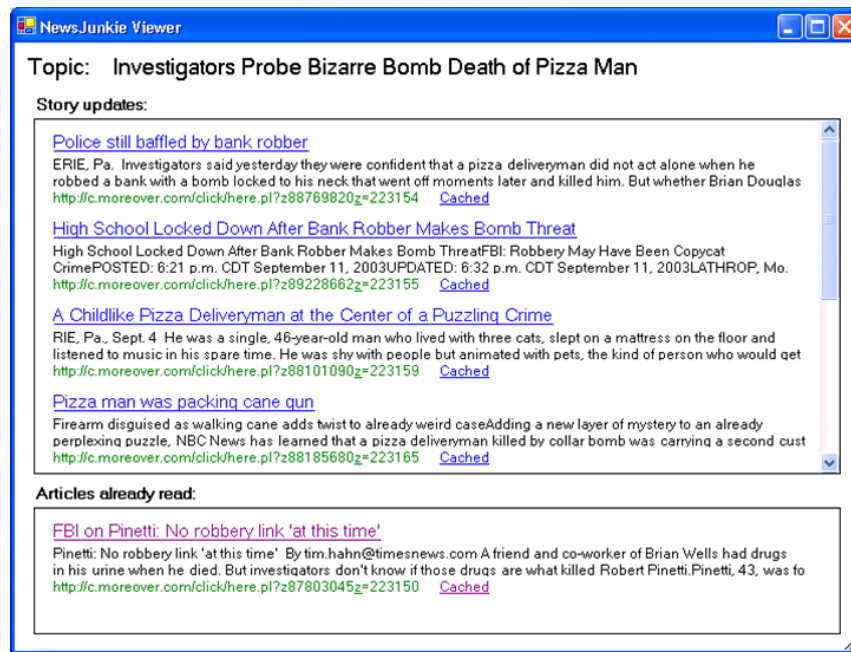


Figure 7.4: Newsjunkie story interface [Gabrilovich et al., 2004].

Several methods have gone beyond timelines and attempted to produce graphs. In email threading [Lewis and Knowles, 1997], the task is to discover connections between email messages. Electronic mail is considered easier than news, since it usually incorporates a strong structure of referenced messages. In [Mei and Zhai, 2005], the authors studied how to discover sub-clusters in a news event and structure them by their dependency, generating a graph structure. However, the authors do not address the notion of coherence at all, and constructing a chain of coherent articles from the output graph is hard. In addition, it seems like the method is best-suited for simple news stories, i.e., stories that can be summarized in one or two keywords (e.g., “tsunami”).

Our system goes beyond the list-of-sentences or list-or-articles models, or even graphs with locally-computed edges. Metro maps offer a *more structured* form of output. Not only does our system pick nuggets of information to display to the user, it explicitly shows connections among different story-lines.

7.4 Notions of Coverage

The purple line in Figure 7.1 explores notions of coverage. Note that the purple line is connected to the novelty detection and summarization stops on the news lines. In fact, coverage and novelty are very similar; both concepts aim to reduce redundancy, while extracting fresh nuggets of interesting data.

Finding novel pieces of information is a strong theme of subtopic retrieval [Zhai et al., 2003b; Chen and Karger, 2006; Carbonell and Goldstein, 1998]. In subtopic retrieval, the task is to

retrieve documents that cover many subtopics of the given query. In the traditional information retrieval setting, it is assumed that the relevance of each document is independent of the other documents. However, in subtopic retrieval the utility of a document is contingent on the other retrieved documents. In particular, a newly retrieved document is relevant only if it covers subtopics other than the ones covered by previous documents. Thus, the concept of relevance in subtopic retrieval is similar to our notion of coverage.

We have chosen to use the coverage notion from [El-Arini et al., 2009] in this work because of its submodularity. Other notions, such as MMR [Zhai et al., 2003a] (in the context of ranking and summarization) do not provide approximation guarantees, and could not be combined with our orienteering algorithm.

7.5 Mapping Science

Many attempts at mapping science have been made over the years, and several systems exist for summarizing and visualizing scientific literature (see [Borner, 2010] for a compendium). However, the output of these systems is often not suitable for a starting researcher.

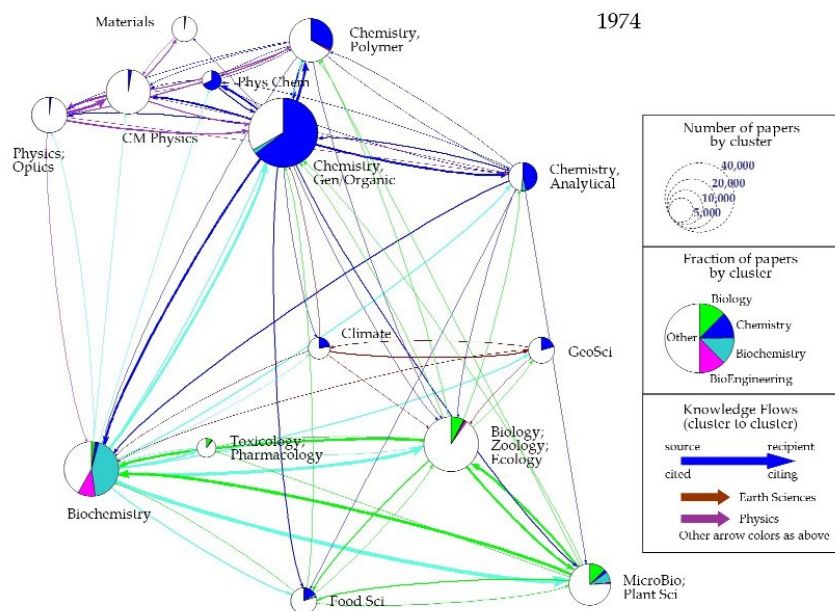


Figure 7.5: Mapping Chemistry [Boyack et al., 2009]. Map of the 14 disciplines, fractions of papers by field for each discipline, and knowledge flows between disciplines for 1974.

Some systems' level of granularity is too coarse: Boyack et al. [2009] provide a graph-summary of chemistry research, where each node corresponds to a *cluster* of disciplines ('Biology-Zoology-Ecology') or clusters of journals (see Figures 7.5,7.6). Bassecoulard and Zitt [1999] produce a hierarchical graph, where nodes correspond to clusters of journals. [Campanario, 1995] used self-organizing maps study interrelationships among journals.

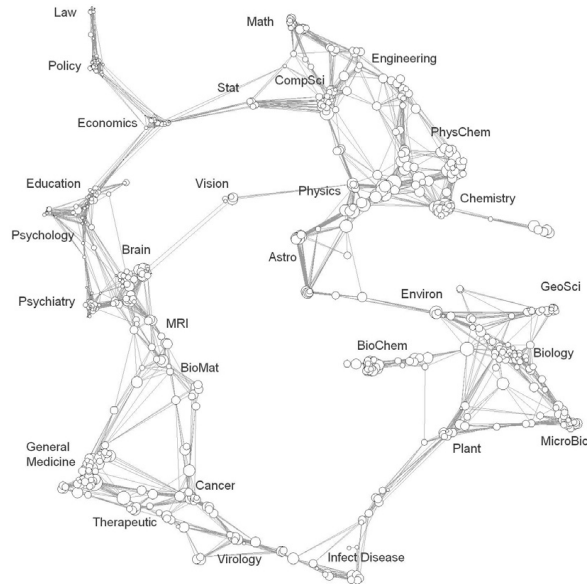


Figure 7.6: Mapping Chemistry [Boyack et al., 2009]. Each node is a cluster of journals, and is sized to show numbers of papers in the journal cluster.

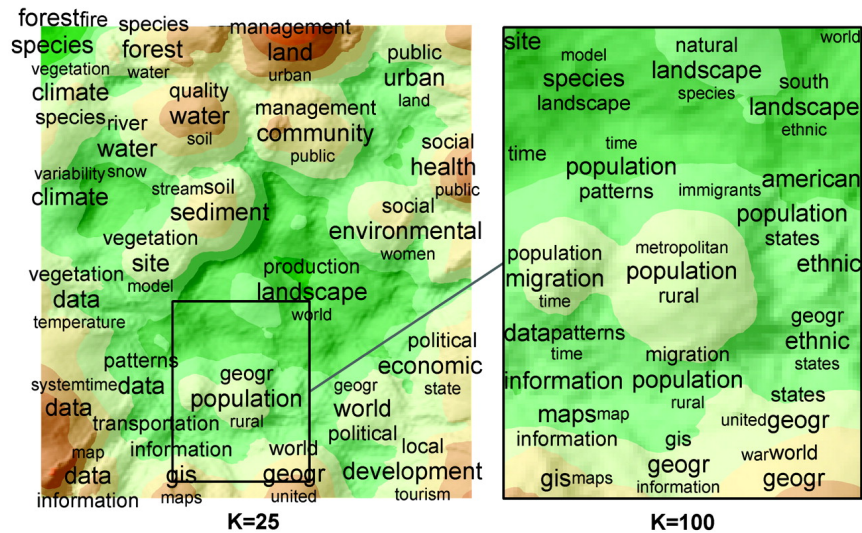


Figure 7.7: Mapping Geography abstracts to visualize the domain [Skupin, 2004].

Other systems' level of granularity is too fine. [Skupin, 2004] uses words from abstracts of geography papers to visualize the domain (Figure 7.7). TextArc uses words as their unit of analysis as well (see "Visualization of The History of Science" in [Borner, 2010]).

We believe that in order to allow researchers to understand how a field is organized, a finer level of granularity is needed. For this reason, we chose *papers* as our unit of analysis. Most current tools that work at this level of granularity provide visualizations of citation or co-citation networks, where papers are nodes [Chen, 2004; Dunne et al., 2010] (see Figure 7.8). Early

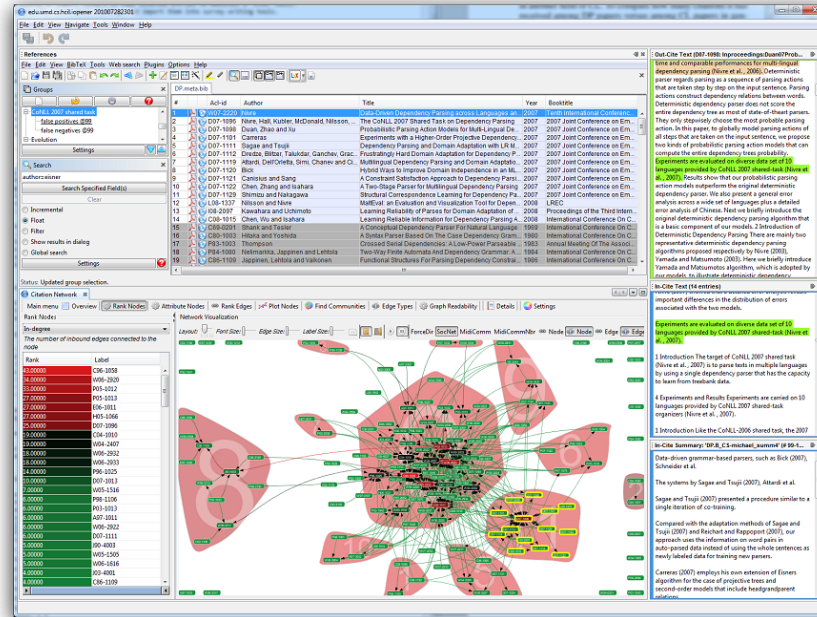


Figure 7.8: iOpener Workbench (Action Science Explorer) tool for summarizing research domains [Chen, 2004; Dunne et al., 2010].

examples include the citation network on DNA [Garfield et al., 1964] and nucleic acids [Allen, 1960]. See [Moya-Anegon et al., 2004] for more examples. Importantly, edges between papers are based on *local* computation, and there is no notion of coherent *lines of research*.

Interestingly, there have been several attempts at mapping the *impact* of science, in particular as a tool for guiding science investments [Lane and Bertuzzi, 2011; Lane, 2009]. It could be interesting to incorporate external data (e.g., funding agencies) into our maps, and quantify how the investments are linked to innovation.

7.6 Summary

In conclusion, our work differs from most previous work in several important aspects – **expressing information needs** and **structured output and interaction**. Our system’s input (a set of articles) might give rise to beyond-keyword input methods, allowing users to express more complex information needs. Our system’s output is interesting, too – instead of the common list of relevant documents, or a graph of strong (but local) connections, our output is more structured: several coherent chains of articles maximizing coverage of salient features of the corpus, and the interactions between the chains. We believe that visually exploring the map output and interacting with it can reveal new and interesting phenomena.

Chapter 8

Open Questions and Criticism



The work discussed in previous chapters opens numerous avenues for further investigations. In this chapter, we outline several promising directions.

In order to demonstrate these directions, let us first take a look at an overview of our current system. Figure 8.1 shows a simplified diagram of our system: our algorithm takes in a dataset and a query, and outputs a map. In the following, we use this diagram to highlight one major, high-level challenge for each component of the system. We then conclude with a number of more tangible challenges.

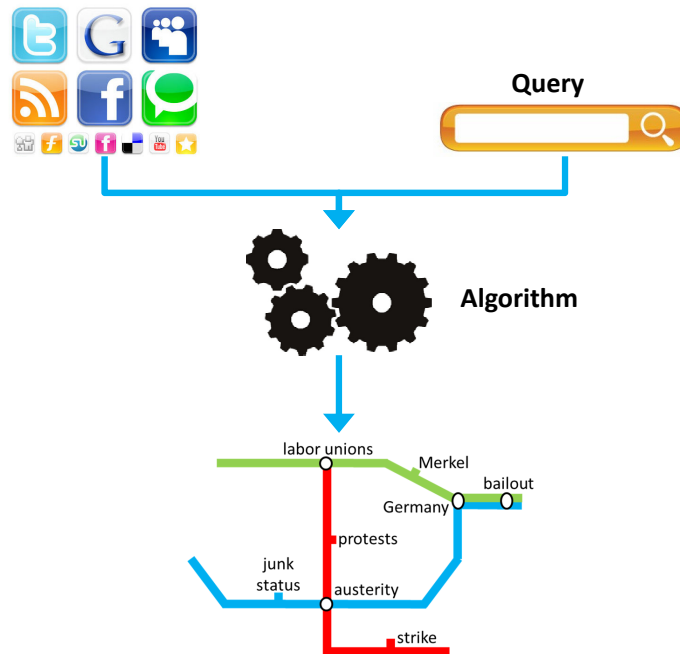
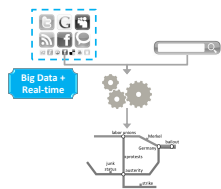


Figure 8.1: An overview of the metro map system. The algorithm (middle) receives data (top left) and a query (top right) and computes a map.

8.1 Scaling Up



The ever-growing amount of data is one of our system's biggest challenges. We wish our system to collect information from continuously updated web sites, blogs and social networks, at a rate of millions of documents per day. In order to do this, our methods must scale.

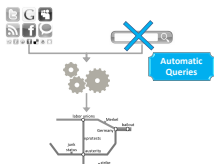
In addition, the real-time nature of domains such as news suggests that we focus on designing *streaming* and *incremental* algorithms. While static algorithms build the model from scratch every time the data is updated, incremental algorithms process each observed value only once, and thus scale well with data sizes.

For example, the first phase of the algorithm can be made incremental: we can cache the coherent chains found in earlier runs, as the addition of new articles does not make the old chains

less coherent. When new articles arrive, we can limit ourselves to finding chains involving the fresh data (similar to a methods for solving big hidden Markov models incrementally).

It may be more difficult to make the coverage phase of the algorithm incremental, as the addition of new articles can change the coverage of old chains (similar to the concept drift phenomenon). However, as the drift is usually slow, perhaps recalculating coverage weights once in a while can still be a satisfactory solution.

8.2 Query Mechanisms



The current implementation of metro maps relies on users specifying queries, but alternative query mechanisms can be conceived. For example,

Automatic Queries: Temporal analysis tools can allow us to identify textual phrases that are gaining in popularity, and formulate queries based on these phrases.

For example, MemeTracker [Leskovec et al., 2009] is a tool that tracks the quotes and phrases that appear most frequently over time across one million online sources, ranging from mass media to personal blogs. MemeTracker makes it possible to see how different stories compete for news and blog coverage each day, and how certain stories persist while others fade quickly (see Figure 8.2).

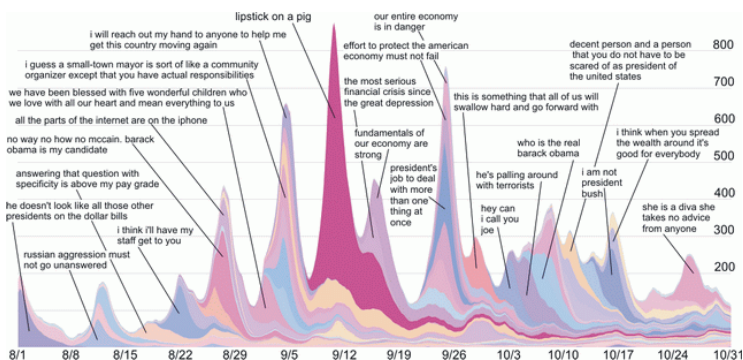


Figure 8.2: MemeTracker Leskovec et al. [2009]: Top 50 threads in the news with highest volume for the period Aug. 1 Oct. 31, 2008. Each thread consists of all news articles and blog posts containing a textual variant of a particular quoted phrase. (Phrase variants for the two largest threads in each week are shown as labels pointing to the corresponding thread.) The data is drawn as a stacked plot in which the thickness of the strand corresponding to each thread indicates its volume over time.

Note that MemeTracker is currently limited to analyzing quoted phrases. In order to gain full advantage of the data, we need to extend MemeTracker to cluster all textual different variants of the same phrase. Clustering textual fragments is interesting because we have very little information and thus traditional bag-of-words representations do not seem appropriate.

Similar to Section 8.1, our phrase clustering algorithm needs to be incremental, as the data is continuously flowing in and re-clustering the phrases every hour is infeasible. We propose to tackle the phrase clustering problem by considering phrases as short shingles and then use

locality-sensitive hashing to identify sets of phrases with long subsequences of words. This approach will allow us to develop scalable and incremental algorithms for identifying mutational variants of the same phrase.

Multiple Queries: Instead of a single query, users could specify multiple entities (e.g., “Angela Merkel” and “Goldman Sachs”). Similar to Connecting the Dots, The algorithm would then compute a map showing connections between those entities.

The problem of connecting two entities has several interesting characteristics. Even choosing the set of candidate documents \mathcal{D} is non-trivial, as some documents linking the two input entities might not mention either entity.

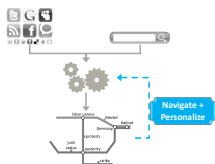
For example, consider two articles: One article accuses Goldman Sachs of helping Greece mask the true scale of its debt, and another article details Merkel’s plan to solve the Greek debt. In order to link these two articles we need articles about the Greek debt, but none of them necessarily mentions Goldman Sachs or Merkel. We may need to apply some form of query expansion to capture related entities, which in turn will help us find a good set of candidate documents \mathcal{D} .

In addition, we need to come up with a connectivity metric to score how well the map connects the two entities. Is a direct chain connecting the two entities better than a connection through an intersection of two lines?

We can push the metro-map metaphor further, and adapt metrics from the transit route network design problem (TRND) Fan et al. [2004]. In TRND, we are given a transit route network (e.g., a train network) and a demand matrix, representing the load between every two nodes in the network. In our case, the demand could be from articles mentioning one entity to articles mentioning the other. TRND scores the network, e.g., based on the percentages of the total demand trips that are able to reach their destination, and the number of transfers needed.

Beyond Keyword Search: As we noted earlier, users often know precisely what they want to find, but it is not easy for them to distill their ideas down into a few *keywords*. Note that the input to maps is not necessarily textual: all the algorithm needs is a set of documents \mathcal{D} to operate on. One may wish to find other ways for the users to specify \mathcal{D} , for example by specifying a few seed documents, or adapting ideas from El-Arini and Guestrin [2011].

8.3 Navigation and Personalization



Metro maps allow many interesting interaction forms. We have touched upon a few in the previous sections, but one may wish to explore different feedback mechanisms and visualization techniques. For example,

Different Resolutions: The user may zoom in to learn more about a topic, or zoom out to get a high-level overview.

In order to reveal patterns at multiple levels, we must generalize the notion of metro maps and explore hierarchical metro maps: instead of representing a single document, we would like stops along our metro lines to correspond

to regions of the next level. In other words, each vertex the map could correspond to a *cluster* of documents.

As is the case with flat maps, one of the major challenges of hierarchical maps is formulating their desired properties. In addition to coherence, coverage and connectivity, we propose three additional properties. First, a good hierarchy should help the user navigate quickly and with minimum effort. For this reason, the hierarchy has to be *compact*; it cannot be too wide or too deep. Second, the clusters should be *consistent*: documents in each cluster should have a common theme. Finally, all these properties should hold *recursively*: each cluster should contain a good (smaller) hierarchical map within it.

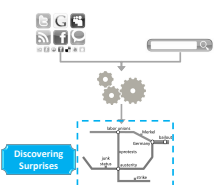
In order to build a good hierarchy, one might wish to build each level in a bottom-up fashion. We could start with a flat metro-map including all the documents, and at each iteration summarize the previous level using larger and larger clusters. Of course, the notions of coherence, coverage and connectivity need to be adapted to clusters.

Points of View: The user may ask to see two maps describing the same topic from two different points of view. In the news scenario, one easy way to achieve this is to compute two maps for the same query over two different datasets. Consider, for example, the New York Times vs. the Wall Street Journal.

Alternatively, the user may specify the point of view he is interested in, perhaps by giving a few examples. This is similar to the El-Arini and Guestrin [2011] notion of *trust*, where users can indicate that they trust a certain author with respect to a concept. A particularly interesting avenue to explore is the mechanism's ability of letting the user see a topic through the eyes of another person.

Line-based Feedback: The word-based feedback mechanism we explored in Sections 2.3, 3.3 may be too aggressive: increasing or decreasing the importance of a single word may result in an entirely new map. Allowing the user to focus on one line at a time (keeping the rest of the map fixed) may improve the user experience.

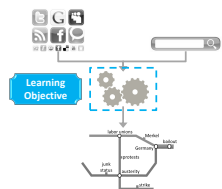
8.4 Quantifying Surprise



Currently, metro maps rely on popularity as a measure of importance, and thus produced maps that were mostly suited for users unfamiliar with the topic. We wish to explore other measures of importance; in particular, we are interested in identifying *surprising* and *insightful* connections in the data for the more advanced user.

For example, one may rely on techniques developed for Literature Based Discovery (LBD) Bruza and Weeber [2008]. LBD strives to find connections in scientific literature that are novel, and have not been previously explicitly published. In a nutshell, the idea is to extract multiple arguments from the literature, and then seek separate arguments which, when combined, yield an argument which cannot be found anywhere. This direction may also turn out to be useful for investigative journalism.

8.5 Learning an Objective



A large portion of this work has been devoted to crafting an objective function: formalizing the desired properties and their trade-offs. In the future, one may wish to *learn* an objective function directly.

The most natural notion from a machine learning perspective would be learning from *curated datasets*. For example, we can have users provide a single label for a map. However, this approach suffers from two limitations. First, from the point of view of the user, it is not very natural to provide feedback on an entire map. Second, since there are exponentially many such maps, we are likely to need an extensive amount of user feedback before we could learn this function.

For these reasons, we propose to learn an objective from *user interaction* logs instead. Search and interaction logs can provide a wealth of information that machine-learning algorithms can harness. We can extract training data from the logs and automatically tailor objective functions to a particular user or a group of users.

A particularly relevant research area for this direction is *web search ranking*. In ranking, collecting relevance judgments on retrieved documents is expensive and time consuming, whereas search logs allow for virtually unlimited data to be collected at very low cost. Furthermore, observations obtained in laboratory settings do not reflect real world usage, and do not allow systems to adapt to changing user behavior patterns. For these reasons, several approaches for automatically learning preferences for ranking have been developed [Radlinski and Joachims, 2005; Agichtein et al., 2006a,b]; we hope to extend some of their insights to metro maps.

8.6 Further Directions

In the previous section we have identified five key challenges. In the following we list several other questions we have encountered in our research.

Higher-level Features: In this work we have focused on low-level features, such as named entities and noun phrases. These features, while useful, are quite limited. It would be interesting to explore incorporating richer features into our framework. For example, we could exploit semantic relations between entities in the text. As another example, we could take advantage of the *structure* of a document.

In particular, in the scientific domain we could try to classify citations based on their position in the paper and on various stylistic and rhetorical cues (similar to [Garzone and Mercer, 2000]). Once we can identify the function of the citation (Assumption, Affirmation, Contrast, Methodology, Related Work, etc.) we may have a better understanding of the cited paper's influence.

Non-textual Data: It is worthwhile to explore metro maps for non-textual data. Two of the most interesting applications are

1. **Numerical Data:** Many of today's datasets are non text-based. Rather, they are often relational: The dataset consists of one or more tables, tables contain one or more records, and records contain one or more values (numerical, textual, categorical,

etc.). For example, one could imagine a metro map summarizing a dataset of medical records.

2. **Images:** In [Heath et al., 2010], Heath et al build graphs called Image Webs to represent massive image collections and understand their global structure; no coherence or coverage notions are discussed. Formulating desired properties for image-based datasets poses a unique set of challenges. For example, what constitutes a coherent chain of images?

Non-Chronological Maps: The documents we have dealt with all had time stamps, and our maps obeyed chronological order. However, one can envision maps without this constraint. For example, what could be the semantics of a circular metro line?

Removing the chronological constraints can also open up many interesting domains. For example, given the right coherence metric, we can apply our method to biology, using DNA strands as documents.

Storylines: There are many ways to extract storylines from a corpus. For example, we could use a generative model to explain the evolution of a story: if we assume that the observed data (\mathcal{D}) are sampled from a generative model, we can fit the model parameters to maximize the data likelihood (see, for example, [Ahmed and Xing, 2010]). Alternatively, we could treat it as a data-compression based, change detection problem (see [van Leeuwen and Siebes, 2008]).

Structure: In this work, we used a simple connectivity measure: if two lines are connected, the map should reflect it. This simple measure does not disclose other structural properties of the topic.

For example, consider the three stories from the user study of Section 4.3. Intuitively, the Chilean miners story is the simplest of the three, with a single dominant storyline. The earthquake in Haiti starts linearly, and then branches into many side-stories. The Greek debt story is the most complex of the three, with many storylines interleaving and intersecting. Ideally, our connectivity measure should convey this underlying structure.

Stability: How much are our maps robust to perturbations in the data? For example, imagine that the map computed for your query included a document d . If we remove d from the dataset and recompute the map, would the map change a lot, or can it substitute d for a similar document?

The greedy nature of our coverage algorithm can cause small perturbations in the data to have a big effect on the map. In the future, we can explore stabilizing mechanisms to increase the algorithm's robustness.

Cognitive User Modeling: Cognitive psychology and educational psychology have explored the ways people comprehend ideas and connections. For example, [Doignon and Falgagne, 1985; Albert and Lukas, 1999] provides a formal model for assessing a learner's knowledge of a topic. The assessment can be used to identify what the learner knows and what he is ready to learn.

Cognitive models have proven to be especially useful for *tutoring systems* [Weber and Specht, 1997; Corbett, 2001; Beaumont, 1994]. The cognitive model enables the tutor to

interpret students' problem-solving actions, while tracing their growing problem-solving knowledge. It would be interesting to incorporate insights from cognitive models into the map algorithm, and compute maps that can help people better comprehend the topic.

Annotation: We propose three annotation mechanisms that can improve the user's understanding of the map. First, annotating the *edges* can expose the relationships between two documents, and can prove extremely valuable to users. This is especially true for the scientific domain, as publications offer a rich palette of interaction possibilities (affirmation, criticism, contrast, methodology, related work, etc.).

Similarly, annotating a line with important keywords will create a map legend that can help the user find his way around the map more efficiently. In this work we labeled the lines using words with the highest incremental coverage given the other lines, but this solution leaves a lot to be desired. For example, many of the words were not useful unless the users knew the story in advance (the name of the Greek minister of finance), or were too general ("billion").

Finally, even node annotation can make a big difference in the user experience. Currently, we annotate the map nodes with the title of the corresponding document. While news articles often have informative titles, research paper titles are often understandable only to those who know the paper in advance. Furthermore, in both domains the titles are often very long, and can be shortened considerably. Perhaps it is worth exploring alternative node-annotation mechanisms, such as key words or even images.

User Interface: User interface has not been the focus of this work. Nevertheless, user interface is critical to the success of metro maps, as even minor usability problems can demotivate users. There are many usability studies and design decisions to be had, including layout of the map, ease of navigation, and semantic zooming.

Chapter 9

Conclusion and Discussion



How snakes say goodbye

Living in an “information society”, people today are constantly bombarded with information. While easy access to information has its benefits, many people are struggling to gain control over the abundance of information that threatens to engulf them.

Treatment and analysis of such excessive amounts of information require appropriate means. Search engines have made tremendous progress in recent years, and regularly help millions of users find information quickly and reliably. However, **search engines are limited** in their capacity to solve the information overload problem. Most importantly, search engines do not expose the underlying *structure* of a topic: by looking at search results, one cannot identify the relations and interconnections between the retrieved pieces.

We believe that a map can be an ideal tool for exposing the underlying structure of a topic. In the real world, cartographic maps have been relied upon for centuries to help us understand our surroundings and the relationships between neighbouring objects. Furthermore, map representations have been shown to be effective instruments to describe, explore and summarize large amounts of data. Well-designed maps can help users enhance, retain and improve knowledge.

We chose to focus on a particular form of map representation, called **issue maps**. Issue maps are meant to expose the structure of an argument, and allow users to evaluate the soundness of argument premises and the inferential connections. Unfortunately, generating good issue maps is a *manual*, time-consuming process. Furthermore, the maps cannot be easily updated when new data becomes available. In order to overcome these issues, we need to *automate* the creation of issue maps.

This thesis research revolves around the following statement:

“ We can effectively manage information overload by producing personalized issue maps in response to the user’s expressed needs. ”

In this work, we proposed two systems which serve as stepping stones on the way to automated issue maps. The systems incrementally expand on the set of possibilities determined by **expressing information needs, structured output and interaction**.

Our work has drawn from diverse areas such as data mining, information retrieval, algorithms, optimization theory, human computation, and even vision. Our approach is inspired by summarization and sensemaking approaches; we also leverage lessons learned from various data representation tools, especially visualizations of science. In the following sections, we summarize the main contributions of this thesis.

9.1 Connecting the Dots

Maps are complex, and creating one can be daunting. Before tackling an entire map, we focus on a simpler task: a map containing only a single line. To further simplify the task, we start by assuming that the endpoints of the line are known. In other words, we are interested in investigating methods for automatically *connecting the dots*. Given two articles, s and t , our

system automatically finds the most coherent chain of articles linking them together. Our main contributions are as follows:

- **Proposing the problem of finding a coherent chain.** Formalizing characteristics of a good chain and the notion of *coherence*.

The main challenge of connecting the dots is formalizing a notion of coherence. We argue that coherence is a *global* property of the chain, and cannot be captured by local interactions between neighbouring articles in the chain. Rather, a coherent chain can often be characterized by a small set of words.

We formulate coherence as an optimization problem, and look for a small set of words w that capture the essence of the story (*active* words). The choice of active words determines the score of each transition between neighbouring articles. The score of the chain is then determined by the score of its weakest link.

- **Adapting the notion of coherence to the news domain.** We formalized a notion of *influence* of a word w in a transition between two documents d_i and d_j . Since in the news domain we do not have links between articles, we had to rely on content alone. We construct a bipartite graph between words and documents, and computed the importance of word w for random walks between documents d_i and d_j .

- **Adapting the notion of coherence to the science domain.** When formalizing a notion of *influence*, we took advantage of the citation graph; the citation graph captures the way ideas are transferred between papers, and helps us disambiguate between similar concepts.

- **Providing an algorithm for connecting two fixed endpoints while maximizing chain coherence.** We apply a generalized best-first search strategy to find the most coherent chain. Given a chain, we use a Linear Programming solver to find the optimal set of words and evaluate its coherence.

- **Incorporating feedback and interaction mechanisms into our system,** tailoring stories to user preferences. We provide mechanisms to refine a chain, to chain the importance of certain words, or to incrementally build a chain.

In the latest case, our main challenge is to pick a *small* set of *diverse* candidate articles which cover possible ways to continue the chain. We rely on a submodular notion of coverage to pick the set of articles.

- **Evaluating our algorithm over real news data** and demonstrating its utility to news readers via a user study. Our user studies demonstrate that the objective we propose captures the users' intuitive notion of coherence, and that our algorithm effectively helps users understand the news.

9.2 Metro Maps

After we found a way to evaluate single chains, we proposed a methodology for creating structured summaries of information called metro maps. Metro maps consist of a concise structured set of documents which maximizes coverage of salient pieces of information. The documents are organized as a set of coherent storylines (metro lines). Most importantly, metro maps explicitly

show the relations among different pieces in a way that captures story development. Our main contributions are as follows:

- **Proposing the notion of a metro map and formalizing metrics characterizing good metro maps:** *coherence*, *coverage* and *connectivity*. The goal of coverage is twofold: we want to both cover *important* aspects of the story, but also encourage *diversity*. The goal of connectivity is to expose the ways storylines interact with each other.
- **Adapting metro maps to the news domain:**
 - We propose a notion of coverage that relies on the article content alone. We view map coverage as a Bernoulli process (a series of biased coin flips): each document in the map tries to cover feature w with probability $cover_{d_i}(w)$. The coverage of w is the probability at least one of the documents succeeded.
 - We propose a simple notion of connectivity relying on intersections between pairs of lines.
- **Adapting metro maps to the scientific domain**, taking advantage of the additional structure encoded in the citation graph:
 - Quantifying the impact of one paper on the corpus, using random coherent walks on the citation graph.
 - Proposing a notion of connectivity that captures how different lines of research can still interact with each other, despite not intersecting. We propose a soft notion of connectivity, rewarding lines which had impact on each other.
- **Providing efficient methods with theoretical guarantees** to compute these metrics and find a diverse set of high-impact, coherent storylines and their interactions. Our algorithm encodes all coherent chains as a graph, exploits submodularity to extract a set of high-coverage from the graph, and then applies local search to improve connectivity.
- **Integrating user preferences into our framework** by providing appropriate user-interaction models.
- **Performing validation studies with users** that highlight the promise of the methodology, as our method outperforms popular competitors. We show how our algorithms effectively help users understand the news, especially for stories without a single dominant storyline. In the scientific domain, we show that map users find better papers and cover more important areas than users of competitor systems.

9.3 Future Directions

In this work, we have taken several steps towards the goal of automated issue maps. This dissertation opens up many opportunities of future research, both theoretical and applicative. Below, we outline several promising directions:

- The ever-growing amount of data is one of our system’s biggest challenges. We wish our system to collect information from continuously updated web sites, blogs and social networks, at a rate of millions of documents per day. In order to do this, our methods must

scale.

- The current implementation of metro maps relies on users specifying queries, but alternative query mechanisms can be conceived. Temporal analysis tools can allow us to identify textual phrases that are gaining in popularity, and formulate queries based on these phrases.
- Metro maps allow many interesting interaction forms that we have not explored. For example, the user can zoom in to learn more about a topic, or zoom out to get a high-level overview. Alternatively, the user may ask to see two maps describing the same topic from two different points of view.
- Currently, metro maps rely on popularity as a measure of importance, and thus produced maps that were mostly suited for users unfamiliar with the topic. One may explore other measures of importance, for example a map exposing *surprising* and *insightful* connections in the data, aiming at a more advanced user.
- A large portion of this work has been devoted to crafting an objective function: formalizing the desired properties and their trade-offs. In the future, one may wish to *learn* an objective function directly from user interactions.

9.4 Usage of Maps

In this section, we wish to explore possible usages of maps. We rely on the traditional information retrieval framework, and characterize a user by an *information need*. The user formulates his information need in some query language and submits the query to a system. The user then examines the retrieved results; if he is not satisfied with the results, he may interact with the system until satisfied.

Several frameworks have been proposed to characterize information needs. For example, Belkin's Anomalous States of Knowledge (ASK) framework [Belkin et al., 1982] tried to model the cognitive state of the user, identifying different categories of information needs ("Information needed to produce directions for research"). In most of the early frameworks, information needs were adapted from the author's experience and thinking, or from interviews with expert human searchers (e.g., reference librarians).

The popularity of search engines has given rise to search-behaviour studies relying on query-log analysis. In his seminal work, Broder [Broder, 2002] went beyond the question of *what* users are searching for, and looked at *why* users are searching. The result is a 3-class taxonomy, classifying web queries according to their intent:

Navigational: reach a particular web site.

Informational: acquire information available in static form.

Transactional: reach a site through which the user can transact (e.g., a shopping site).

Others have extended this classification, proposing classes such as **Informational:Advice** (get advice or instructions), **Informational:Directed:Closed** (a question that has a single, unambiguous answer) [Rose and Levinson, 2004].

We would like to propose a similar taxonomy for information needs of map users. We do not have query logs to justify the taxonomy; rather, it can be thought of as an informed guess.

Before we discuss the taxonomy, it is important to note that maps are not intended to replace search engines; many of the queries submitted to search engines today are very focused, and can be satisfied by a single piece of information. In contrast, maps are designed to display connections between multiple pieces of information. In terms of Broder's taxonomy, maps are mostly useful for **Informational** queries; they are of little use for **Navigational** and **Transactional** queries. Thus, our taxonomy will focus on the **Informational** branch.

Informational queries are driven by a user's need to *learn* something. The need can range from simple (for example, fact lookup and question answering) to complex (understanding a topic). In order to formalize the different types of learning, we rely on Bloom's Taxonomy [Bloom et al., 1956] and on Marchionini's interpretation of it [Marchionini, 2006].

Bloom, together with a group of educators, undertook the ambitious task of classifying educational goals and objectives. In particular, they have identified six cognitive categories characterizing the processes of learning (see Figure 9.1). At the *Knowledge* Level, a student can recall terms, facts, and basic concepts. At the *Comprehension* level, a student can demonstrate understanding, for example by organizing or comparing important ideas. At the *Application* level, a student is able to use the knowledge he acquires for solving problems. At the *Analysis* level, a student can examine information, make inferences and find plausible generalizations. At the *Synthesis* level, a student can combine elements in new and useful patterns. Finally, at the *Evaluation* level a student makes judgments about information and defends them.

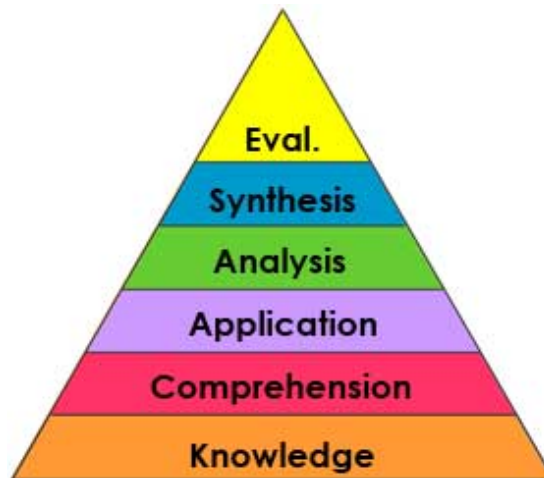


Figure 9.1: Bloom's cognitive categories [Bloom et al., 1956], lowest order processes to the highest: Knowledge (Remember), Comprehension, Application, Analysis, Synthesis (Create), Evaluation.

In our taxonomy (Figure 9.2) we distinguish between two main categories. The **Learn** category corresponds to the lower three levels of Bloom's taxonomy: The goal of the user is to acquire knowledge, to comprehend the state of some topic. The second category in our taxonomy, **Investigate**, corresponds to the higher levels of Bloom's taxonomy. The user aims to produce outcomes, and not merely to collect information.

For example, a user in the **Learn** category might be interested in **Surveying** a concrete topic (Reinforcement Learning, Debt Crisis). The topic may be new to the users, or could be a familiar one they wish to monitor. In a different scenario, the users may not have a concrete topic in mind,

- 1. Learn:** Get a high-level view of a topic, acquire knowledge.
 - 1.1. Survey:** Learn about a concrete topic.
 - 1.1.1. First-timer:** Learn about a topic new to you.
 - 1.1.2. Monitor:** Maintain awareness of the status of a topic.
 - 1.2. Navigate:** Find what is around a specific point.
- 2. Investigate:** Produce outcomes.
 - 2.1. Analyze/ Synthesize:** Identify patterns and relationships. Find evidence to support generalizations. Generate insight by connecting several pieces of knowledge.
 - 2.2. Negative Search:** Discover gaps in knowledge.
 - 2.2.1. Whodunit:** Find if a hypothesis is new. Similarly, find how several concepts have been connected in the past.
 - 2.2.2 Replacement:** What are the implications of replacing a building block with another one?
 - 2.4. Curriculum:** Learn what you do not know that you need to know (what should I have asked?).

Figure 9.2: Our proposed information-needs taxonomy of map users.

but rather a starting point whose surroundings they wish to explore and **Navigate**.

Navigation is a promising application for maps. Many news sites today already present a “Related Articles” feature, indicating articles which the user may find interesting. We propose to augment this feature by a map mechanism, allowing the user to see the article in broader context. Alternatively, adding a Connect-the-Dots sidebar to a news site allows the user to connect the article they are currently reading to other articles (see Figure 9.3).

Note that our **Learn** category does not include a **Lookup** case, corresponding to fact lookup and question answering. As described in Chapter 4, maps are less useful for this type of queries.

We now consider the second category in our taxonomy, **Investigate**. As mentioned earlier, this category corresponds to the higher levels of Bloom’s taxonomy, where the user is trying to produce his own outcomes.

In **Analyze/ Synthesize**, the user tries to transform existing data into new data. They may look for patterns in the data, and try to generate new insights. For example, a journalist may read about racism in LAPD during OJ Simpson’s trial, and look for similar cases; an economist may try to understand whether Spain is following the path of Greece into bankruptcy; a researcher may try to characterize application domains for which SVM is useful. Connecting the dots can also be thought about as a synthesis task.

Another interesting application is **Negative Search** [Garfield, 1970]. In negative search, the user looks for gaps in knowledge. Unlike most search systems today, negative search is recall-oriented: the user needs to explore as much of the information space as possible in order to be

sure that the information is not there.

Two interesting cases of **Negative Search** are **Whodunit** and **Replacement**: In **Whodunit**, the user's goal is to ensure that a particular hypothesis is new (for example, so that new research can begin, avoiding dead-end alleys). This is also very related to connecting the dots: the user may be interesting in connecting two areas, and thus has to learn about all the previous efforts to connect them. In **Replacement**, the task is slightly different: in this case, the user has a specific tool in mind, and he is looking for new places to apply it. For example, the user may have an algorithm that outperforms all previous algorithms in some task. Then, he may go on and see how the previous algorithms have been used, with the intention of starting parallel research lines using their tool of choice.

Finally, in **Curriculum**, the user has some task in mind (for example, buy a new camera), but he is unsure of what he needs to know in order to make the decision. Therefore, he is interested in coming up with a curriculum: a set of pieces of information that he needs to read, and their dependencies. The camera buyer may first read about different styles of cameras, from point-and-shoot to DSLR. Depending on the style chosen, he will be faced with many other pieces of information to digest, analyze and synthesize. The process continues until the user has enough information to execute the task.

9.5 Epilogue

In a way, the problem of constructing metro map does not have a single right solution. In this work we have merely dipped our toes in the water, proposing a direction. Nevertheless, we hope that this work opens a field of opportunities for future research; we believe that automated maps could become a set of powerful tools for any user who needs to make sense of large amounts of data, including web users, intelligence analysts, or scientists.

THE SIMPSON CASE: THE FUGITIVE
THE SIMPSON CASE: THE FUGITIVE; Simpson Is Charged, Chased, Arrested

By SETH MYDANS,
Published: June 18, 1994

LOS ANGELES, June 17— The police today charged O. J. Simpson with murdering his former wife and a friend of hers, then pursued him for about 50 miles along Southern California highways this evening before he finally surrendered outside his home here, ending a long day on the run.

The extraordinary pursuit, broadcast to the nation by the television networks, developed about six hours after Mr. Simpson suddenly vanished instead of surrendering to the authorities at midday as his lawyer had arranged.

The police undertook a vast manhunt and, by tracking calls placed from a cellular telephone inside a van, found him this evening in the vehicle, a white Ford Bronco, as it traveled up an Interstate highway in Orange County south of Los Angeles.

The van was apparently being driven by Al Cowlings, a longtime friend and former football teammate who had been at Mr. Simpson's side for much of the week. As the van made its way up the Interstate toward Los Angeles, Mr. Simpson held a gun to his own head, a spokesman for the California Highway Patrol said.

A phalanx of police cars, their lights flashing, fell in behind his vehicle, traveling about 35 miles an hour.

- SIGN IN TO E-MAIL
- PRINT
- SINGLE-PAGE

WHAT'S CONNECTED?

Legend:
Blue: Race
Red: DNA
Green: Verdict



THE SIMPSON CASE: THE FUGITIVE
THE SIMPSON CASE: THE FUGITIVE; Simpson Is Charged, Chased, Arrested

By SETH MYDANS,
Published: June 18, 1994

LOS ANGELES, June 17— The police today charged O. J. Simpson with murdering his former wife and a friend of hers, then pursued him for about 50 miles along Southern California highways this evening before he finally surrendered outside his home here, ending a long day on the run.

The extraordinary pursuit, broadcast to the nation by the television networks, developed about six hours after Mr. Simpson suddenly vanished instead of surrendering to the authorities at midday as his lawyer had arranged.

The police undertook a vast manhunt and, by tracking calls placed from a cellular telephone inside a van, found him this evening in the vehicle, a white Ford Bronco, as it traveled up an Interstate highway in Orange County south of Los Angeles.

The van was apparently being driven by Al Cowlings, a longtime friend and former football teammate who had been at Mr. Simpson's side for much of the week. As the van made its way up the Interstate toward Los Angeles, Mr. Simpson held a gun to his own head, a spokesman for the California Highway Patrol said.

A phalanx of police cars, their lights flashing, fell in behind his vehicle, traveling about 35 miles an hour.

- SIGN IN TO E-MAIL
- PRINT
- SINGLE-PAGE

WHAT'S CONNECTED?

- THE SIMPSON CASE: THE FUGITIVE (June 94)
- Blood on a Simpson Glove Matches Victim's (May 95)
- Near-Unanimous Verdict: Blunder at Simpson Trial (Jun 95)
- Key Fibers Linked to Simpson Vehicle (Jun 95)
- Simpson Defense Changes Glove Tactics (Aug 95)

CONNECT TO...
Simpson Judge Permits Evidence on Racial... (Jan 95)
Simpson Defense Changes Glove Tactics (Aug 95)
Simpson's Civil Trial Opens on Tight Rein (Sept 96)

Figure 9.3: Use Case (Illustration). Top: Adding a map sidebar to a news site allows a user to understand the broader context of the article, or navigate to related articles. Bottom: Adding a Connect-the-Dots sidebar to a news site allows the user to connect the article they are reading (top right) and other articles.

Bibliography

- Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 19–26, New York, NY, USA, 2006a. ACM. ISBN 1-59593-369-7. doi: 10.1145/1148170.1148177. URL <http://doi.acm.org/10.1145/1148170.1148177>. 8.5
- Eugene Agichtein, Eric Brill, Susan Dumais, and Robert Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 3–10, New York, NY, USA, 2006b. ACM. ISBN 1-59593-369-7. doi: 10.1145/1148170.1148175. URL <http://doi.acm.org/10.1145/1148170.1148175>. 8.5
- Amr Ahmed and Eric P. Xing. Timeline: A dynamic hierarchical dirichlet process model for recovering birth/death and evolution of topics in text stream. In *UAI'10*, pages 20–29, 2010. 8.6
- D. Albert and J. Lukas. *Knowledge Spaces: Theories, Empirical Research, and Applications*. Taylor & Francis, 1999. ISBN 9780805827996. URL <http://books.google.de/books?id=Cpu2sM2eCvgC>. 8.6
- J Allan, J Carbonell, G Doddington, J Yamron, Y Yang, U Amherst, and J A Umass. Topic detection and tracking pilot study, 1998. 7.3
- James Allan, Victor Lavrenko, and Hubert Jin. First story detection in tdt is hard. In *Proceedings of the ninth international conference on Information and knowledge management*, CIKM '00, pages 374–381, New York, NY, USA, 2000. ACM. ISBN 1-58113-320-0. doi: 10.1145/354756.354843. URL <http://doi.acm.org/10.1145/354756.354843>. 7.3
- James Allan, Rahul Gupta, and Vikas Khandelwal. Temporal summaries of new topics. In *SIGIR '01*, 2001. ISBN 1-58113-331-6. doi: <http://doi.acm.org/10.1145/383952.383954>. URL <http://doi.acm.org/10.1145/383952.383954>. 1
- James Allan, Ao Feng, and Alvaro Bolivar. Flexible intrinsic evaluation of hierarchical clustering for tdt. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 263–270, New York, NY, USA, 2003. ACM. ISBN 1-58113-723-0. doi: 10.1145/956863.956914. URL <http://doi.acm.org/10.1145/956863.956914>. 7.3
- Gordon Allen. Citation network. Citation Index to Genetics and General Science Literature,

- edited by Eugene Garfield. Research proposal by the Institute for Scientific Information, submitted to the National Science Foundation, 1960. 7.5
- E Bassecoulard and M Zitt. Indicators in a research institute: A multi-level classification of scientific journals. *Scientometrics*, 44(3):323–345, 1999. URL <http://www.springerlink.com/index/10.1007/BF02458483>. 7.5
- Ian H. Beaumont. User modelling in the interactive anatomy tutoring system anatom-tutor. *User Modeling and User-Adapted Interaction*, 4:21–45, 1994. ISSN 0924-1868. URL <http://dx.doi.org/10.1007/BF01142356>. 10.1007/BF01142356. 8.6
- N. J. Belkin, R. N. Oddy, and H. M. Brooks. Ask for information retrieval: Part ii. results of a design study. *Journal of Documentation*, 38(3):145–164, 1982. 9.4
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *JMLR*, 2003. 4.3.3, 6.2.2
- Benjamin S Bloom, Max D Engelhart, Edward J Furst, and Walker H Hill, editors. *Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain*. Longman, White Plains, NY, USA, 1956. (document), 9.4, 9.1
- Katy Borner. *Atlas of Science: Visualizing What We Know*. MIT Press, 2010. 7.1, 7.5, 7.5
- Kevin Boyack, Katy Börner, and Richard Klavans. Mapping the structure and evolution of chemistry research. *Scientometrics*, 79:45–60, 2009. ISSN 0138-9130. URL <http://dx.doi.org/10.1007/s11192-009-0403-5>. (document), 7.5, 7.5, 7.6
- S Brin and L Page. The anatomy of a large-scale hypertextual web search engine. In *Computer Networks and ISDN Systems*, 1998. 4.2.1
- Andrei Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, September 2002. ISSN 0163-5840. doi: 10.1145/792550.792552. URL <http://doi.acm.org/10.1145/792550.792552>. 9.4
- Peter D. Bruza and Marc Weeber. *Literature-based Discovery*. Springer, 2008. URL <http://eprints.qut.edu.au/15498/>. 8.4
- Tony Buzan and Barry Buzan. *The Mind Map Book*. BBC Books, London, 2 edition, 1995. 1.1, 7.1
- J. Campanario. Using neural networks to study networks of scientific journals. *Scientometrics*, 33(1):23–40, May 1995. doi: 10.1007/BF02020773. URL <http://dx.doi.org/10.1007/BF02020773>. 7.5
- Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based re-ranking for reordering documents and producing summaries. In *SIGIR*, 1998. 7.4
- Chandra Chekuri and Martin Pal. A recursive greedy algorithm for walks in directed graphs. In *FOCS '05*, 2005. ISBN 0-7695-2468-0. doi: <http://dx.doi.org/10.1109/SFCS.2005.9>. URL <http://dx.doi.org/10.1109/SFCS.2005.9>. 3.2.2, 2, 2
- Chaomei Chen. Searching for intellectual turning points: Progressive knowledge domain visualization. *PNAS*, 101(Suppl 1):5303–5310, April 2004. ISSN 0027-8424. doi: 10.1073/pnas.0307513100. URL <http://dx.doi.org/10.1073/pnas.0307513100>. (docu-

ment), 7.5, 7.8

Harr Chen and David Karger. Less is more. In *SIGIR*, 2006. 7.4

Copernic. Copernic, <http://www.copernic.com>. 4.2.1, 4.3.1

Albert T. Corbett. *Cognitive Computer Tutors: Solving the Two-Sigma Problem*. 2001. doi: 10.1007/3-540-44566-8_14. 8.6

Fernando Das-Neves, Edward A. Fox, and Xiaoyan Yu. Connecting topics in document collections with stepping stones and pathways. In *Proceedings of the 14th ACM international conference on Information and knowledge management, CIKM '05*, pages 91–98, New York, NY, USA, 2005. ACM. ISBN 1-59593-140-6. doi: 10.1145/1099554.1099573. URL <http://doi.acm.org/10.1145/1099554.1099573>. 7.2

Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a*. *J. ACM*, 32:505–536, July 1985. ISSN 0004-5411. doi: <http://doi.acm.org/10.1145/3828.3830>. URL <http://doi.acm.org/10.1145/3828.3830>. 2.2.1

Jean-Paul Doignon and Jean-Claude Falmagne. Spaces for the Assessment of Knowledge. *International Journal of Human-computer Studies / International Journal of Man-machine Studies*, 23:175–196, 1985. doi: 10.1016/S0020-7373(85)80031-6. 8.6

Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *SIGIR '08*, pages 595–602. ACM, 2008. ISBN 978-1-60558-164-4. doi: <http://doi.acm.org/10.1145/1390334.1390436>. URL <http://doi.acm.org/10.1145/1390334.1390436>. 3.3.1

Cody Dunne, Ben Shneiderman, Bonnie Dorr, and Judith Klavans. iopener workbench: tools for rapid understanding of scientific literature. In *Proc. 27th Annual Human-Computer Interaction Lab Symposium*, 2010. URL <http://www.cs.umd.edu/hcil/about/events/symposium2010>. (document), 7.5, 7.8

K El-Arini, G Veda, D Shahaf, and C Guestrin. Turning down the noise in the blogosphere. In *KDD '09*, 2009. ISBN 978-1-60558-495-9. doi: <http://doi.acm.org/10.1145/1557019.1557056>. 2.3, 7.4

Khalid El-Arini and Carlos Guestrin. Beyond keyword search: Discovering relevant scientific literature. In *KDD' 11*, August 2011. 5.2.1, 5.2.1, 5.2.2, 5.3, 8.2, 8.3

Martin J Eppler. A comparison between concept maps, mind maps, conceptual diagrams, and visual metaphors as complementary tools for knowledge construction and sharing. *Information Visualization*, 5(3):202–210, 2006. doi: 10.1057/palgrave.ivs.9500131. URL <http://ivi.sagepub.com/content/5/3/202.abstract>. 7.1

David Kirk Evans, Judith L. Klavans, and Kathleen R. McKeown. Columbia newsblaster: multilingual news summarization on the web. In *Demonstration Papers at HLT-NAACL 2004, HLT-NAACL–Demonstrations '04*, pages 1–4, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. URL <http://dl.acm.org/citation.cfm?id=1614025.1614026>. 7.3

Christos Faloutsos, Kevin S. McCurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In *KDD '04*, 2004. ISBN 1-58113-888-1. doi: <http://doi.acm.org/10.1145/>

- 1014052.1014068. URL <http://doi.acm.org/10.1145/1014052.1014068>. (document), 7.2, 7.3
- W. Fan, R.B. Machemehl, Southwest Region University Transportation Center (U.S.), University of Texas at Austin. Center for Transportation Research, and University Transportation Centers Program (U.S.). *Optimal Transit Route Network Design Problem: Algorithms, Implementations, and Numerical Results*. Southwest University Transportation Center, University of Texas at Austin, Center for Transportation Research, 2004. URL <http://books.google.com/books?id=5FyoSgAACAAJ>. 8.2
- Paul Farrand, Fearzana Hussain, and Enid Hennessy. The efficacy of the ‘mind map’ study technique. *Medical Education*, 36(5):426–431, 2002. ISSN 1365-2923. doi: 10.1046/j.1365-2923.2002.01205.x. URL <http://dx.doi.org/10.1046/j.1365-2923.2002.01205.x>. 1.1, 7.1
- Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24: 381–395, June 1981. ISSN 0001-0782. doi: <http://doi.acm.org/10.1145/358669.358692>. URL <http://doi.acm.org/10.1145/358669.358692>. 15
- Edward A. Fox, Fernando Das Neves, Xiaoyan Yu, Rao Shen, Seonho Kim, and Weiguo Fan. Exploring the computing literature with visualization and stepping stones & pathways. *Commun. ACM*, 49(4):52–58, April 2006. ISSN 0001-0782. doi: 10.1145/1121949.1121982. URL <http://doi.acm.org/10.1145/1121949.1121982>. 7.2
- E Gabrilovich, S Dumais, and E Horvitz. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. In *WWW '04*, 2004. ISBN 1-58113-844-X. doi: <http://doi.acm.org/10.1145/988672.988738>. (document), 1, 7.3, 7.4
- E. Garfield. When is a negative search result positive? *Essays of an Information Scientist*, 1970. 9.4
- E. Garfield, I. Sher, and R. Torpie. The Use of Citation Data in Writing the History of Science. Technical report, 1964. 7.5
- Mark Garzone and Robert Mercer. Towards an automated citation classifier. In Howard Hamilton, editor, *Advances in Artificial Intelligence*, volume 1822 of *Lecture Notes in Computer Science*, pages 337–346. Springer Berlin / Heidelberg, 2000. ISBN 978-3-540-67557-0. URL http://dx.doi.org/10.1007/3-540-45486-1_28. 8.6
- Google. Google news timeline, <http://newstimeline.googlelabs.com/>. 3
- Richard H. Hall and Angela O’Donnell. Cognitive and affective outcomes of learning from knowledge maps. *Contemporary Educational Psychology*, 21(1):94 – 101, 1996. ISSN 0361-476X. doi: 10.1006/ceps.1996.0008. URL <http://www.sciencedirect.com/science/article/pii/S0361476X96900089>. 1.1
- K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L.J. Guibas. Image webs: Computing and exploiting connectivity in image collections. In *CVPR '10*, june 2010. doi: 10.1109/CVPR.2010.5539991. 7.2, 2
- M. Shahriar Hossain, Michael Narayan, and Naren Ramakrishnan. Efficiently discovering ham-

- mock paths from induced similarity networks. *CoRR*, abs/1002.3195, 2010. 7.2
- M. Shahriar Hossain, Christopher Andrews, N. Ramakrishnan, and Chris North. Helping intelligence analysts make connections. In *Proceedings of the AAAI 2011 Workshop on Scalable Integration of Analytics and Visualization*, 2011. (document), 7.2
- D Kempe, J Kleinberg, and É Tardos. Maximizing the spread of influence through a social network. In *KDD '03*, 2003. ISBN 1-58113-737-0. doi: <http://doi.acm.org/10.1145/956750.956769>. 4.2.1
- Jon Kleinberg. Authoritative sources in a hyperlinked environment, 1999. 4.2.1
- Jon Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 91–101, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775061. URL <http://doi.acm.org/10.1145/775047.775061>. 7.3
- Deept Kumar, Naren Ramakrishnan, Richard F. Helm, and Malcolm Potts. Algorithms for storytelling. In *KDD '06*, 2006. ISBN 1-59593-339-5. doi: <http://doi.acm.org/10.1145/1150402.1150475>. URL <http://doi.acm.org/10.1145/1150402.1150475>. 7.2
- Julia Lane. Assessing the impact of science funding. *Science*, 324(5932):1273–1275, 2009. doi: 10.1126/science.1175335. URL <http://www.sciencemag.org/content/324/5932/1273.short>. 7.5
- Julia Lane and Stefano Bertuzzi. Measuring the results of science investments. *Science*, 331(6018):678–680, 2011. doi: 10.1126/science.1201865. URL <http://www.sciencemag.org/content/331/6018/678.short>. 7.5
- Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD*, 2007. 16, 6.2.3
- Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 497–506, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: 10.1145/1557019.1557077. URL <http://doi.acm.org/10.1145/1557019.1557077>. (document), 8.2, 8.2
- D. D. Lewis and K. A. Knowles. Threading electronic mail: A preliminary study. *Information Processing and Management*, 33, 1997. 7.3
- David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007. ISSN 1532-2890. doi: 10.1002/asi.20591. URL <http://dx.doi.org/10.1002/asi.20591>. 4.2.1
- Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, April 2006. ISSN 0001-0782. doi: 10.1145/1121949.1121979. URL <http://doi.acm.org/10.1145/1121949.1121979>. 9.4
- Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>, 2002. 4.3.3
- Q Mei and C Zhai. Discovering evolutionary theme patterns from text: an exploration of temporal

- text mining. In *KDD '05*, 2005. ISBN 1-59593-135-X. doi: <http://doi.acm.org/10.1145/1081870.1081895>. 7.3
- Felix Moya-Anegon, Benjamin Vargas-Quesada, Victor Herrero-Solana, Zaida Chinchilla-Rodriguez, Elena Corera-Alvarez, and Francisco J Munoz-Fernandez. A new technique for building maps of large scientific domains based on the cocitation of classes and categories. *Scientometrics*, 61:129–145, September 2004. 7.5
- R Nallapati, A Feng, F Peng, and J Allan. Event threading within news topics. In *CIKM '04*, 2004. ISBN 1-58113-874-1. doi: <http://doi.acm.org/10.1145/1031171.1031258>. 4, 4.3.2, 7.3
- G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14, 1978. 3.2.2, 6
- John C. Nesbit and Olusola O. Adesope. Learning with concept and knowledge maps: A meta-analysis. *Review of Educational Research*, 76(3):413–448, 2006. doi: 10.3102/00346543076003413. URL <http://rer.sagepub.com/content/76/3/413.abstract>. 1.1, 7.1
- K.V. Nesbitt. Getting to more abstract places using the metro map metaphor. In *Information Visualisation '04*, july 2004. doi: 10.1109/IV.2004.1320189. 7.1
- J Niehaus and R M Young. A computational model of inferencing in narrative. In *AAAI Spring Symposium '09*, 2009. 7.2
- Joseph D. Novak. Concept mapping: A useful tool for science education. *Journal of Research in Science Teaching*, 27(10):937–949, 1990. ISSN 1098-2736. doi: 10.1002/tea.3660271003. URL <http://dx.doi.org/10.1002/tea.3660271003>. 1.1, 7.1
- Joseph D. Novak and Alberto J. Cañas. The theory underlying concept maps and how to construct and use them. research report 2006-01 Rev 2008-01, Florida Institute for Human and Machine Cognition, 2006. URL <http://cmap.ihmc.us/Publications/ResearchPapers/TheoryCmaps/TheoryUnderlyingConceptMaps.htm>. (document), 1.2
- Angela O'Donnell, Donald Dansereau, and Richard Hall. Knowledge maps as scaffolds for cognitive processing. *Educational Psychology Review*, 14:71–86, 2002. ISSN 1040-726X. URL <http://dx.doi.org/10.1023/A:1013132527007.10.1023/A:1013132527007>. 1.1, 7.1
- Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. Introduction to the special issue on summarization. *Comput. Linguist.*, 28(4):399–408, December 2002. ISSN 0891-2017. doi: 10.1162/089120102762671927. URL <http://dx.doi.org/10.1162/089120102762671927>. 1, 7.3
- Filip Radlinski and Thorsten Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 239–248, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X. doi: 10.1145/1081870.1081899. URL <http://doi.acm.org/10.1145/1081870.1081899>. 8.5

- Kirsten L Rewey, Donald F Dansereau, and Jennifer L Peel. Knowledge maps and information processing strategies. *Contemporary Educational Psychology*, 16(3):203 – 214, 1991. ISSN 0361-476X. doi: 10.1016/0361-476X(91)90021-C. URL <http://www.sciencedirect.com/science/article/pii/0361476X9190021C>. 1.1, 7.1
- Sharon B. Reynolds, Michael E. Patterson, Lisa P. Skaggs, and Donald F. Dansereau. Knowledge hypermaps and cooperative learning. *Comput. Educ.*, 16(2):167–173, February 1991. ISSN 0360-1315. doi: 10.1016/0360-1315(91)90023-K. URL [http://dx.doi.org/10.1016/0360-1315\(91\)90023-K](http://dx.doi.org/10.1016/0360-1315(91)90023-K). 1.1
- Daniel E. Rose and Danny Levinson. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web, WWW '04*, pages 13–19, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: 10.1145/988672.988675. URL <http://doi.acm.org/10.1145/988672.988675>. 9.4
- J P Rowe, S W McQuiggan, J L Robison, D R Marcey, and J C Lester. Storyeval: An empirical evaluation framework for narrative generation. In *AAAI Spring Symposium '09*, 2009. 7.2
- Biplab K. Sarker, Peter Wallace, and Will Gill. Some observations on mind map and ontology building tools for knowledge management. *Ubiquity*, 2008(March):2:1–2:9, March 2008. ISSN 1530-2180. doi: 10.1145/1353568.1353570. URL <http://doi.acm.org/10.1145/1353568.1353570>. 1.1, 7.1
- Dafna Shahaf and Carlos Guestrin. Connecting the dots between news articles. In *KDD '10*, 2010. ISBN 978-1-4503-0055-1. doi: <http://doi.acm.org/10.1145/1835804.1835884>. URL <http://doi.acm.org/10.1145/1835804.1835884>. (document), 1, 2, 4
- Dafna Shahaf and Carlos Guestrin. Connecting two (or less) dots: Discovering structure in news articles. *ACM Trans. Knowl. Discov. Data*, 5(4):24:1–24:31, February 2012. ISSN 1556-4681. doi: 10.1145/2086737.2086744. URL <http://doi.acm.org/10.1145/2086737.2086744>. (document)
- Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. Metro maps of science. In *KDD '12*, 2012a. (document)
- Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. Trains of thought: Generating information maps. In *WWW '12*, 2012b. (document), 5.2.2
- Andre Skupin. The world of geography: Visualizing a knowledge domain with cartographic means. In *Proc. Nat'l Academy Sciences vol. 101, Suppl. 1*, pages 5274–5278, 2004. (document), 7.7, 7.5
- Russell Swan and David Jensen. TimeMines: Constructing Timelines with Statistical Models of Word Usage. In *KDD' 00*, 2000. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.3147.1,7.3>
- S R Turner. The creative process: A computer model of storytelling and creativity, 1994. 7.2
- Matthijs van Leeuwen and Arno Siebes. Streamkrimp: Detecting change in data streams. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 5211 of *Lecture Notes in Computer Science*, pages 672–687. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-87478-2. 8.6

- Dingding Wang, Tao Li, and Mitsunori Ogihara. Generating pictorial storylines via minimum-weight connected dominating set approximation in multi-view graphs. In *AAAI Conference on Artificial Intelligence*, 2012. URL <https://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5074/5257>. 7.2
- Gerhard Weber and Marcus Specht. User modeling and adaptive navigation support in www-based tutoring systems. pages 289–300. Springer, 1997. 8.6
- Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *SIGIR' 11*, 2011. ISBN 978-1-4503-0757-4. doi: <http://doi.acm.org/10.1145/2009916.2010016>. URL <http://doi.acm.org/10.1145/2009916.2010016>. 1, 7.3
- Cheng Xiang Zhai, William W. Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR '03*. ACM, 2003a. ISBN 1-58113-646-3. doi: <http://doi.acm.org/10.1145/860435.860440>. URL <http://doi.acm.org/10.1145/860435.860440>. 7.4
- Chengxiang Zhai, William W. Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR*, 2003b. 7.4