OpenAIR@RGU

The Open Access Institutional Repository
at Robert Gordon University

http://openair.rgu.ac.uk

Citation Details

Citation for the version of the work held in 'OpenAIR@RGU':

CHAKRABORTI, S., 2007. Introspective knowledge acquisition for case retrieval networks in textual case base reasoning. Available from *OpenAIR@RGU*. [online]. Available from: http://openair.rgu.ac.uk

Copyright

Items in 'OpenAIR@RGU', Robert Gordon University Open Access Institutional Repository, are protected by copyright and intellectual property law. If you believe that any material held in 'OpenAIR@RGU' infringes copyright, please contact openair-help@rgu.ac.uk with details. The item will be removed from the repository while the claim is investigated.

Introspective Knowledge Acquisition for Case Retrieval Networks in Textual Case Base Reasoning

Sutanu Chakraborti

A thesis submitted in partial fulfilment
of the requirements of
The Robert Gordon University
for the degree of Doctor of Philosophy

August 2007

Dedicated

To

Param Pujyapada Sri Sri Dada

and

My Parents

Abstract

Textual Case Based Reasoning (TCBR) aims at effective reuse of information contained in unstructured documents. The key advantage of TCBR over traditional Information Retrieval systems is its ability to incorporate domain-specific knowledge to facilitate case comparison beyond simple keyword matching. However, substantial human intervention is needed to acquire and transform this knowledge into a form suitable for a TCBR system. In this research, we present automated approaches that exploit statistical properties of document collections to alleviate this knowledge acquisition bottleneck. We focus on two important knowledge containers: relevance knowledge, which shows relatedness of features to cases, and similarity knowledge, which captures the relatedness of features to each other. The terminology is derived from the Case Retrieval Network (CRN) retrieval architecture in TCBR, which is used as the underlying formalism in this thesis applied to text classification.

Latent Semantic Indexing (LSI) generated concepts are a useful resource for relevance knowledge acquisition for CRNs. This thesis introduces a supervised LSI technique called "sprinkling" that exploits class knowledge to bias LSI's concept generation. An extension of this idea, called Adaptive Sprinkling has been proposed to handle inter-class relationships in complex domains like hierarchical (e.g. Yahoo directory) and ordinal (e.g. product ranking) classification tasks. Experimental evaluation results show the superiority of CRNs created with sprinkling and AS, not only over LSI on its own, but also over state-of-the-art classifiers like Support Vector Machines (SVM).

Current statistical approaches based on feature co-occurrences can be utilized to mine similarity knowledge for CRNs. However, related words often do not co-occur in the same document, though they co-occur with similar words. We introduce an algorithm to efficiently mine such indirect associations, called higher order associations. Empirical results show that CRNs created with the acquired similarity knowledge outperform both LSI and SVM.

Incorporating acquired knowledge into the CRN transforms it into a densely connected network. While improving retrieval effectiveness, this has the unintended effect of slowing down retrieval. We propose a novel retrieval formalism called the Fast Case Retrieval Network (FCRN) which eliminates redundant run-time computations to improve retrieval speed. Experimental results show FCRN's ability to scale up over high dimensional textual casebases.

Finally, we investigate novel ways of visualizing and estimating complexity of textual casebases that can help explain performance differences across casebases. Visualization provides a qualitative insight into the casebase, while complexity is a quantitative measure that characterizes classification or retrieval hardness intrinsic to a dataset. We study correlations of experimental results from the proposed approaches against complexity measures over diverse casebases.

Acknowledgements

I am deeply indebted to my supervisors Dr. Nirmalie Wiratunga, Dr. Robert Lothian and Dr. Stuart Watt for their rock-solid support throughout my three years of stay at Aberdeen. I have been through patches of turbulent times when I have had you as mentors and advisors, but most importantly as my friends and well-wishers. Everything I have here in this thesis, I owe it to you. Thank you.

I am grateful to Rahman Mukras, Ulises Cervino Beresi, Stewart Massie and Amandine Orecchioni, who have lent their helping hand to me for part of the work reported in this thesis. Thank you Ganesan for all that you have done to make sure I survived my not-so-brief illness in 2005. All my other colleagues at CTC including Muhammad, Bayo, Bhavani, Ratiba, Stella, Selpi, Sid, David, Ralph, and Zia have richly contributed to a vibrant research ambience; thank you for helping me out when I ran out of my idea reserves. Thanks Daniel for your evening music classes at CTC, sorry I'm still stuck at C and could not move over to the more fashionable C sharp. Thanks to Prof. Susan Craw for your bits and pieces of thoughtful advice throughout my stay. A special note of thanks to Dr. Ines Arana for being so accommodative of my concerns, in her capacity as research co-coordinator. I am indebted to anonymous reviewers of my conference papers; your valuable insights, suggestions and critical comments have played a significant role in shaping up this work.

I'll never have words enough to thank Dr. Yaji Sripada, Sandhya and dear little Adithya for all you have meant to me for these three years. You have spared your valuable time to ensure I always felt at home, and you have always been with me when I needed you most. I have loved the warmth of your company and profited immensely from your words of wisdom. A special word of thanks must go to my friends at my hall of residence, including Taoufik, Sohan, Boudi, Naresh, Hemant, Viju, Kavya, Ram, Sujith, the Samuels (1 and 2), Rajesh, Yuhong, Ricky, Alex, Rafal, Eva, Amith and Denali for all the fun we had together. My friends back in India, including Basuri, Ravindra, Kiran, Kishalay-da, Sailaja and Mahua have ensured I never suffer from e-solitude. I am grateful to Dr. Deepak Khemani, my MS-thesis advisor at IIT Madras, for all his encouragement and inspiration.

My parents have steered me through my most difficult times, as they have always done in my life. The debt of gratitude I owe them is beyond any expression. My sister, brother-in-law and my lovely little niece Aditi all cheered me up when I needed them the most. I benefited richly from all that my brother-in-law shared with me, regarding his PhD experiences. From my 4-year old niece, I learnt how to cope up with the pressure of being surrounded by people all more knowledgeable than me. She says "never give up", she is perhaps right. Thank you dear.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The White Rabbit put on his spectacles. 'Where shall I begin, please your Majesty?' he asked. 'Begin at the beginning,' the King said, very gravely, 'and go on till you come to the end: then stop.' Lewis Carroll, Alice's Adventures in Wonderland.

The volume of electronically recorded data is growing astronomically. It was estimated in 2006 that more data will be produced in 2007 than has been generated during the entire existence of humankind (Panurgy 2006). Disks are doubling every 18 months or so, and that is clearly not enough. The much bigger issue, however, is that human ability to absorb and use this growing mass of data has remained constant over the years. When data is available in structured form, as in databases or spreadsheets, automated approaches can be effective in crunching numbers and symbols, and helping us make sense of the data flood. Making sense of unstructured data still remains a largely unsolved problem. Examples of such free-form data abound in the form of e-mails, memos, notes from call centers and support operations, news, user groups, chats, reports, surveys, white papers, research articles, presentations and Web pages. The magnitude of the problem can be appreciated in the light of an estimate by Merrill Lynch that more than 85% of all business information exists as unstructured text (Bloomberg and Atre 2003).

An important aspect that characterizes structured data, as distinct from unstructured text, is the availability of an unambiguous context for interpreting the data. A database entry recording a number 45 under the field temperature, immediately elevates the status of the number 45 from data to usable information that can be processed for making inferences and associated meaningfully with other pieces of information ("this is the hottest and second most humid day of this summer, clearly explaining the discomfort"). In contrast, no such precise reference frame is available for handling unstructured data; thus the same fact can be expressed in several different ways in free text. Furthermore, the problem is compounded by the fact that natural language sentences are often ambiguous and ill-formed. The traditional approach to Natural Language Processing (NLP) involved formalizing rules of grammar to break sentences down to more meaningful "deeper" representations that capture interrelationship between structural elements, like phrases. Examples of phrases are noun phrases ("This thesis") and verb phrases ("sounds stupid") are interesting, since they usually correspond to natural semantic elements, which can help in constructing the meaning of the sentence. While this approach of "deep NLP" has found limited success in sentence-level understanding and machine translation, its applicability is limited by the fact that it is slow and does not scale well beyond single sentences to handle text at a paragraph or discourse level. A second approach to NLP is corpus-based, and relies on a probabilistic, as opposed to logical, model of language. There are several advantages of this approach (Russell and Norvig 2003):

Convenient training from data. Learning is based on simple estimates obtained by counting occurrences and co-occurrences.

Robustness. Statistical approaches can handle ill-formed sentences that do not conform to the grammar strictly. While linguistically driven NLP systems reject any ungrammatical string, corpus based approaches accept any string, albeit with a low probability.

Disambiguation. Statistical approaches typically assign different probabilities to each of the possible interpretations (senses); ambiguity is thus resolved by choosing the most likely interpretation.

Given huge volumes of unstructured texts, a common task is to look for texts, or parts thereof, that are relevant to a certain information need. Information Retrieval (IR) is the discipline that studies this problem, and IR models and approaches form the backbone for most search systems operational over the web. Risking over-generalization, IR approaches can be treated as statistical NLP systems that use large volumes of unstructured data to make a priori estimates of term relevances. Most practical IR systems use fairly simple models that treat each document as a bag of words (BOW) that are independent of each other, and thereby ignore word order and syntactic patterns. Clearly, this is quite unlike the way humans deal with text. More often than not, irrelevant documents retrieved by such simple approaches far outnumber the relevant ones. However, the lack of sophistication is accompanied by efficiency advantages, which is critical in the face of extraordinarily large volumes of data, as in the web. It should come as no surprise that finding strategies that lead to more effective and cognitively sound retrieval while retaining the efficiency edge constitutes an important research direction. This sets a platform for collaboration between Artificial Intelligence (AI) approaches and IR. An example of such collaboration is Textual Case Based Reasoning (TCBR).

Case Based Reasoning (CBR) is an AI paradigm, inspired by cognitive models of human memory. Operationally, CBR is a process of solving new problems based on solutions to similar problems encountered in the past. A case is a recorded episode of problem solving, and is often structured into a set of feature values to facilitate similarity matching with other cases. The similarity measures used to compare cases are usually specific to the domain. In (Richter 1998), CBR is seen as relying on four knowledge containers, which include the set of cases, the vocabulary used to describe the case structure and the similarity measure. In many real-world tasks like helpdesks and diagnostics, the records of problem solving are typically textual and not readily available in structured form. This motivates the sub-field of TCBR, which strives to handle cases directly in the textual form. The challenge here is to automate or semi-automate the process of acquiring the knowledge containers needed for effective problem solving. Broadly construed, this thesis aims to contribute novel approaches to address this challenge. A part of our work will specifically address the issue of supervised classification domains, where a select set of textual cases, referred to as training cases, are accompanied by a class or category label that identifies a broad sub-topic within the collection. In a collection of textual cases on sports news, for example, we can have classes such as *baseball*, *cricket*, *hockey* and *football*. A classification task is a variant of the retrieval task that uses the knowledge of training cases to assign class labels to unlabelled cases, referred to as test cases. In Section 1.1 and in the next chapter, we will identify the scope and assumptions behind our work, and position it in the context of other relevant approaches. The rest of this chapter outlines the motivation and objectives of this research.

1.1 Textual Case Based Reasoning in Context

TCBR attempts to strike a middle ground between simple and fast IR approaches founded on BOW and more knowledge rich approaches as inspired by AI techniques. The downside to most real world knowledge-based approaches is the knowledge engineering bottleneck, which in the TCBR context refers to the cost of acquiring domain-specific knowledge containers. Traditionally, CBR presupposes that cases are structured, such that the similarity between two cases can be obtained by computing similarity between their constituent features, and aggregating these feature-specific scores. In contrast, TCBR must do away with the assumption that cases are neatly structured. The absence of structure and a well-defined feature space makes comparison of textual cases harder. Even when a hypothetical feature space (using words as features, say) is constructed to facilitate comparison, it is difficult to ascertain that like is compared with like. This is because the surface meaning of text is often different from the deep or intended meaning, and only certain facets of the deeper meaning are meaningful for case comparison. A significant knowledge engineering effort is often involved to compensate for TCBR's non-reliance on readily structured cases. In most practical systems, we are interested in striking a tradeoff between the quality of retrieval and cost of knowledge acquisition.

In discussing knowledge acquisition in the context of TCBR, we will primarily focus on two knowledge containers: the relevance knowledge, which shows how strongly related features are to cases, and similarity knowledge, which captures how strongly features are related to each other. This terminology is derived from literature on Case Retrieval Networks (CRNs), a formalism to facilitate fast retrieval in CBR systems, first presented in (Lenz and Burkhard 1996). Figure 1.1 shows a CRN constructed for a simple domain having 4 cases. The cases are described by feature values, which are referred to as Information Entities (IEs) in the CRN literature. In our example, we have a vocabulary of nine words which act as IEs. Each case is linked to its constituent IEs by a relevance arc. IEs are linked to each other by similarity arcs which assume a real value in the range of 0 (in which case no arc is shown) to 1. In response to a query like "dog licking mirror", a two-step retrieval process is initiated. In the first step all IEs having non-zero similarity to the query IEs *dog*, *lick* and *mirror* are activated. In our example these IEs are *animal*, *cat*, *bite* and *glass*. This step is alternately referred to as query expansion. In the second step, all cases relevant to the expanded set of IEs are activated. Case nodes aggregate incoming activations from IEs, and are ranked according to the strengths of their activations. The activation of a case is thus a function of the similarity and relevance values defined in the network. While we have used binary relevance values in our example, real values can be used to model the degree of relatedness of an IE to a case. It is interesting to see that our example query leads to the retrieval of the case "animals biting glass", though the case shares no words in common with the query. Clearly, the effectiveness of retrieval is critically determined by the similarity and relevance knowledge, which have been relatively simple to encode in this toy domain, but are difficult to acquire in any realistic TCBR application. In their consolidated review paper on retrieval, reuse, revision and retention in CBR, Lopez De Mantaras, R. et al. (2005) note that "The approach (CRN) is efficient and flexible enough to handle incomplete case descriptions, but can incur a significant knowledge engineering cost in constructing the network." This very briefly summarizes the motivation behind the current thesis: proposing introspective machine learning techniques that can effectively automate acquisition of similarity and relevance for CRNs .

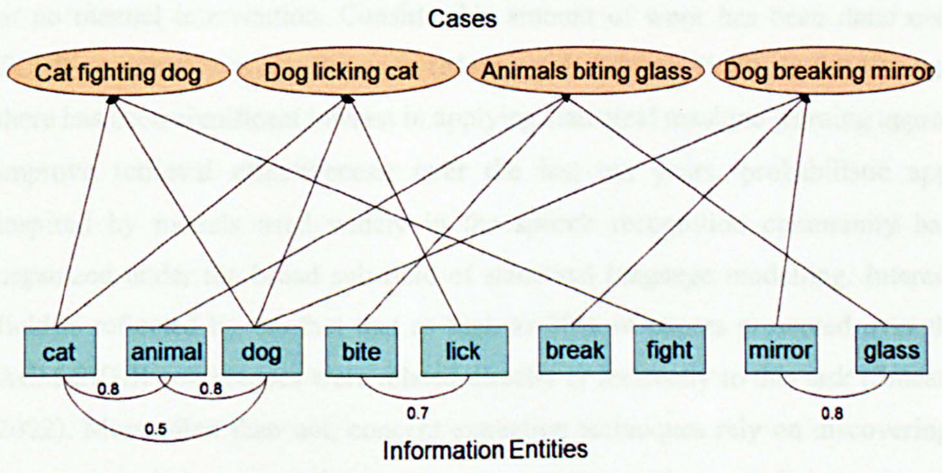


Figure 1.1 A Case Retrieval Network

Automated approaches for knowledge acquisition in TCBR systems are founded on the idea of moving from word-level representations to concept-level representations. Concepts can be interpreted in two ways. Firstly, concepts can refer to linguistic entities like phrases or other domain specific grammatical patterns that correspond to feature values and can be extracted from texts easily. The field of Information Extraction is based on this notion of concepts. The second interpretation of concepts is a statistical one, in which word co-occurrence patterns are used as the basis for inferring underlying concepts. We can then abstract out mathematical (often probabilistic) representations of these concepts, which are more robust indicators of the textual content than the words themselves. A simplistic example would be the grouping of near-synonymous words *car* and *automobile* into a single concept, based on the fact that there is a fair degree of commonality in the words they co-occur with. Once such concepts are learnt, a query on cars can retrieve cases dealing with automobiles as well, and vice versa. In addition to handling synonymy, statistical approaches are also effective in disambiguating polysemous words, i.e. words like *bank*, which assume different meanings based on the context in which they are used. The main strength of the statistical approach is that in most cases, concept extraction is automatic, with little or no manual intervention. Considerable amount of work has been done outside the TCBR community in this direction (Manning & Schütze 1999). In the IR community, there has been significant interest in applying statistical machine learning approaches to improve retrieval effectiveness; over the last ten years, probabilistic approaches inspired by models used widely in the speech recognition community have been organized under the broad sub-field of statistical language modelling. Interest in this field is reflected by the fact that as high as 30% of papers presented over the last 5 ACM SIGIR conferences were related directly or indirectly to this task (Smeaton et al, 2002). More often than not, concept extraction techniques rely on discovering hidden associations between words; so this area overlaps with research interests within the Text Mining community as well. In the recent past, several TCBR researchers have adapted these methods to their tasks, or presented novel extensions to cater to specific TCBR needs (Weber, et al. 2006).

1.1.1 The Thesis : A Quick tour

As a starting point for our research we explored the idea of exploiting Latent Semantic Indexing (LSI), a well established statistical concept induction approach, to the problem of acquiring similarity and relevance knowledge for TCBR tasks. LSI has been shown to improve retrieval effectiveness in IR in several independent studies (Deerwester, et al. 1990, Dumais 1993). In addition, LSI needs very little manual intervention, generates concepts that can be elegantly explicated in terms of underlying features as well as cases, and the acquired knowledge integrates easily into the CRN framework. While initial results were promising, we observed that in supervised classification tasks, a TCBR system based on knowledge inferred by LSI was outperformed by state-of-the art classifiers reported in machine learning literature. We hypothesized that this was because LSI relied solely on word co-occurrences and failed to take into account the class labels of training cases, while acquiring relevance and similarity knowledge. On the other hand, class labels are critical to concept learning in most other approaches. This motivated us to propose a supervised LSI technique called "sprinkling" that exploits class knowledge to bias the acquired similarity and relevance knowledge. Next, we consider more complex classification domains where the classes are related to each other. Two examples are hierarchical classes (e.g. Yahoo directory) and ordinal classes (e.g. ratings 1 to 5 in movie review, each rating treated as a class). Sprinkling is limited in that it fails to take into account class relationships. This motivated us to investigate approaches that can scale up to handle such complex classification problems. We next focussed our attention on the problem of mining similarity knowledge for CRNs. Most current statistical approaches to address this, like association rule mining (Wiratunga et al., 2005a) or distributional word clustering (Baker & McCallum 1998) are, at their roots, founded on the basic idea of estimating similarity based on the number of times the two features co-occurred in a given corpus. However, this is an inherently restrictive supposition. As an example, if words *web* and *browse* co-occur in one document, and words *internet* and *browse* in another, we can infer that *web* and *internet* are related to each other, even if they do not co-occur in any document. Such a relation is called a second-order association. We can extend this to orders higher than two; such associations are called higher order associations. While the significance of higher order co-occurrences was noted elsewhere (Lemaire & Denhiere, 2006), we have not come across any approach that exploited these associations to acquire feature similarity. This motivated us to explore how such associations can be mined efficiently from corpuses and how the effects of different orders can be aggregated to model similarity between features. We also explored extensions of this idea to supervised classification domains. As we will examine later, the effectiveness improvements obtained using knowledge rich retrieval techniques founded on statistical approaches like LSI or higher order associations, are accompanied by a slowing down of the retrieval. This is a critical concern from a practical standpoint. This motivates a novel retrieval formalism presented in this thesis, called the Fast Case Retrieval Network (FCRN). The motivation behind FCRN was to explore ways of eliminating redundant computations at retrieval time, leading to improvements in time performance. Finally, in the course of our experimental evaluations over diverse textual datasets, we realized that the applicability and effectiveness of approaches was often critically dependent on properties of the underlying dataset. In response to this observation, we investigated novel ways of visualizing and estimating complexity of textual casebases. Visualization provides a qualitative insight into the casebase, while complexity is a quantitative measure that indicates the level of difficulty in carrying out effective retrieval or classification over the dataset. While the scope of our research as reported in this thesis is restricted to making preliminary use of the visualization and complexity measure to explain our empirical findings, we believe that our ideas have the potential to be independently pursued to other applications such as facilitating knowledge acquisition from experts, and periodic maintenance of the casebase.

1.2 Research Objectives

This thesis explores techniques to acquire relevance and similarity knowledge for TCBR tasks. LSI was to be extended to acquire relevance knowledge for supervised classification tasks, and to handle casebases with diverse inter-class interrelationships. The intuition behind higher order associations between features was to be incorporated in an algorithm for mining similarity knowledge. Retrieval time efficiency implications of the proposed approaches were to be addressed, to demonstrate the feasibility of applying these techniques to handle real world applications. In order to obtain an insight into how diversity of casebases affects retrieval effectiveness, we also needed to make preliminary investigations into novel ways of visualizing and characterizing textual casebase complexity.

Specifically, we address the following five objectives.

1. Propose supervised extensions of LSI to mine relevance knowledge in classification domains.

2. Propose approaches that extend the scope of LSI to handle situations where class inter-relationships are critical, e.g. hierarchical and ordinal domains.

3. Propose supervised and unsupervised approaches to exploit higher order associations to mine feature similarity.

4. Propose a fast retrieval formalism that can use the acquired relevance and similarity knowledge to facilitate effective retrieval while minimizing retrieval time by cutting down on redundant computations.

5. Propose novel approaches to visualize and estimate complexity of textual casebases, so that they can be meaningfully compared.

1.3 Thesis Overview

In this chapter, we have highlighted challenges in acquiring knowledge for TCBR, which serve as motivation for the work reported in this thesis, and also outlined the specific objectives of our research. In the next chapter, we will examine more closely related work from the fields of TCBR, IR and machine learning that set the backdrop for our work. In particular, we will make a comparative study of techniques that abstract out concepts from a bag of words, and hence are potentially useful to knowledge acquisition in TCBR. We will also review retrieval formalisms in TCBR, and existing methods to visualize and compare textual casebases. This will help us in positioning our work in its context more crisply, and lay a foundation for the remaining chapters to build on.

Chapter 3 provides an overview of TCBR problems in specific relation to the supervised classification task. We present a simple approach to visualize textual casebases, and show how this can be extended to measure and compare complexity of casebases. To illustrate our ideas, we use these tools to characterize six experimental datasets that are used in our evaluations. We verify predictions from our complexity estimates against performances reported by standard classifiers. In later chapters, we will attempt to explain empirical findings from approaches proposed in this thesis in the light of these complexity estimates.

Chapter 4 presents an introduction to LSI, and shows how it can be used for acquiring similarity and relevance knowledge for TCBR applications. We specifically look at mathematical underpinnings of LSI that explain its ability to represent features as well as cases in terms of a common set of underlying concepts, making it particularly attractive to TCBR.

Chapter 5 introduces sprinkling, a supervised extension of LSI which is a novel contribution of this thesis. We also present Adaptive Sprinkling that extends this idea to diverse classification tasks.

In Chapter 6, we present a novel algorithm for mining feature similarities based on higher order associations between features. We also present extensions of this idea to supervised tasks.

Chapter 7 presents experimental evaluation of the ideas presented in Chapters 5 and 6. The empirical findings are critically analysed both across approaches and in relation to dataset complexities estimated in Chapter 3.

Chapter 8 proposes a novel formalism, FCRN, and shows how it can eliminate runtime computations to speed up retrieval. We also present extensions of FCRN to facilitate more flexible retrieval.

The conclusions in Chapter 9 summarise the contributions of the research, identify some of its limitations, and suggest possible extensions and directions of future research.

Chapter 2

Background

I don't paint things, I paint only the difference between things... Matisse 1908

This chapter aims at positioning our work in the context of related research in TCBR. In particular, we will address issues associated with knowledge acquisition in TCBR, scalable architectures to embed and effectively make use of the acquired knowledge, and ways of visualizing this knowledge. Since each of these constitutes broad TCBR sub-fields on their own, it will be ambitious to attempt a comprehensive and thorough comparison with related work. Rather, our goal will be to cover as much ground as motivates and sets in perspective the chapters that follow.

The organization of this chapter is as follows. The following section identifies key challenges in TCBR, and sets the big picture that motivates our research. In particular, we highlight the importance of statistical concept learners in automating knowledge acquisition for TCBR. Section 2.2 is intended to be a consolidated survey of concept induction techniques. Section 2.3 is a review of TCBR architectures in which concept learners can be embedded. In Section 2.4 we look at the role of visualization in TCBR. In each of sections 2.2, 2.3 and 2.4 we identify limitations of existing approaches, and these serve as motivations for the rest of the thesis. Section 2.5 concludes this chapter.

2.1 Introduction: Challenges in Textual Case Based Reasoning

A recent survey of TCBR systems by Weber et al. (2006) identifies four major challenges in CBR systems:

1. assessing similarity between textually represented cases
2. mapping texts to structured representations
3. adaptation of textual cases
4. automatically generating representations for TCBR.

While such a breakdown facilitates a preliminary grouping of existing TCBR systems, we feel that it may be misleading to consider any of these problems in isolation from the rest. In particular, challenges 1, 2 and 4 are closely intertwined. Similarity assessment (challenge 1) makes sense, only when a representation is chosen and methods identified to acquire structured case representations (challenges 2 and 4). Also, the richer the underlying representation, the more effective the TCBR system is, in problem solving. However, richer representations call for higher manual intervention in acquiring the necessary knowledge and structuring this knowledge to facilitate effective retrieval. Thus for any given domain, the choice of an underlying representation is governed by the domain-specific logistics of attaining a trade-off between (a) the quality of retrieval and (b) cost of knowledge acquisition. We illustrate this using the diagram in Figure 2.1, which is inspired by (Brown et al., 1998).

Grouped in the bottom left corner of Figure 2.1 are techniques that can be used to automatically build textual case representations, but such representations are not rich enough to allow sophisticated retrieval. By and large, these approaches are founded on Information Retrieval models, a concise survey of which is presented in (Rijsbergen 1979). An extreme situation is treating a case as a bag of words (BOW), and treating each distinct word in the casebase as a feature for indexing. Weights are assigned to features, based on how frequently they occur in a case (term frequency), and how strongly they discriminate a case from the rest of the cases (inverse document frequency). Each case is modelled as a vector in a vector space, with the features mapping onto the dimensions. The cosine similarity between two vectors is treated as a measure of similarity between the corresponding cases. Though very simple, this approach has received considerable attention in the TCBR community (Lenz et al., 1998b), possibly because it serves as the building block for more sophisticated systems.

As we shall see in Section 2.2, approaches that lead to richer representations can often be equivalently regarded as realizing a different weighting on BOW features.

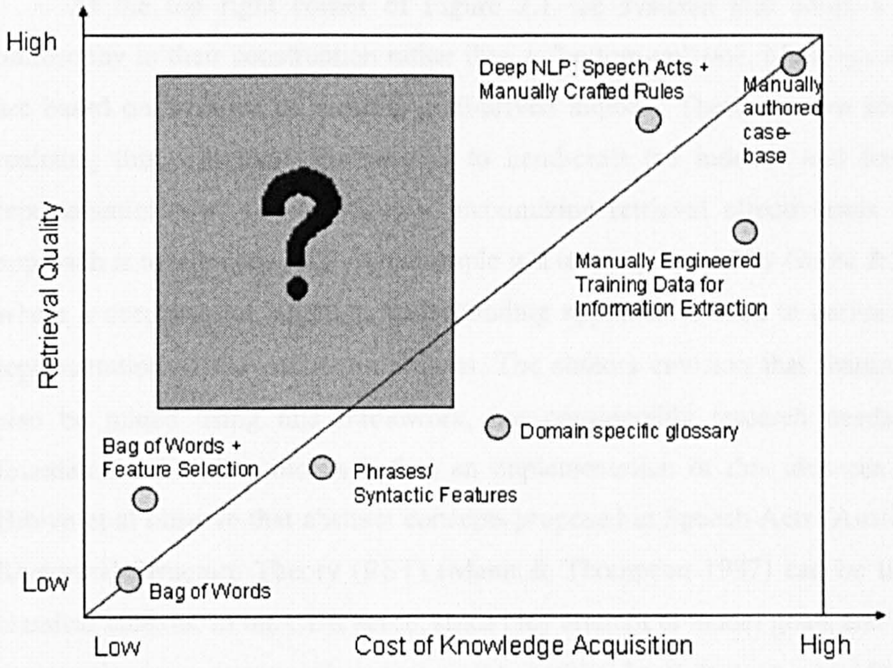


Figure 2.1: the stress-strain relationship between cost of knowledge acquisition and Quality of Retrieval

Moving slightly away from the BOW extreme in Figure 2.1 are approaches that attempt to identify meaningful "information entities" that are semantically richer than words on their own. Two examples are: phrases, made of contiguous words and word groups, made of related words that are not necessarily adjacent to each other. These entities are typically extracted using a combination of "partial" syntactic analysis (as opposed to deep NLP) based on part-of-speech-tags and statistical units like n-grams. In situations where a domain specific glossary is readily available, this can be a useful resource as well. It may be noted that often the resulting representations can again be mapped onto a vector space, with the information entities treated as features. An example of a TCBR system that belongs to this category is FallQ (Lenz & Burkhard

1997). The use of phrases has had a turbulent history, with several authors (Scott & Matwin 1999) reporting no significant performance improvements over BOW.

At the top right corner of Figure 2.1 are systems that adopt a "top-down" philosophy in their construction rather than a "bottom-up" one. More specifically, they are based on the idea of creating goal-driven indexes. There are two broad ways of realizing this. The first approach is to hand-craft the indexes and the underlying representation with the objective of maximizing retrieval effectiveness. The second approach is to use deep NLP. An example is a recent proposal by Gupta & Aha (2004), where a deep natural language understanding approach is used to derive a first-order representation of the unstructured texts. The authors envision that feature values can also be mined using this framework, but considerable research needs to go into foundational building blocks before an implementation of this idea can be realized. Brown et al observe that abstract concepts proposed in Speech Acts (Austin 1962) and Rhetorical Structure Theory (RST) (Mann & Thompson 1987) can be thought of as classical indexes, in the CBR sense, since they attempt to model goals and intentions of communicating agents, rather than using surface level text as a medium for such communication. The main criticism of the first approach is the prohibitively high cost involved in knowledge acquisition, making it infeasible for all but trivial demonstrations. In contrast, the second approach is limited by the difficulty in grounding the theories in sufficiently crisp rules to allow for automation, even within restricted domains.

The relative ineffectiveness of bottom-up approaches, and the practical bottlenecks associated with realizing top-down ones with low manual intervention motivates us to investigate the feasibility of a viable middle ground. This is shown as the grey area in Figure 2.1, sandwiched between the bottom left and top-right extremes. To date, there have been two distinct threads of research in TCBR to attain this middle ground. The first is based on Information Extraction (IE), and the second on modelling underlying concepts using statistical mining of term associations.

The first approach attempts to extract feature value pairs based on structural information that can be automatically extracted from documents (Riloff & Lehnert 1994). Examples of information entities that can be extracted with relative ease are dates, locations of events and names of people or organizations. In the terminology associated with the MUC series of conferences (MUC 91), these pieces of information are referred to as annotations. The annotated document may be used for further analysis based on domain specific knowledge, a significant part of which is acquired and encoded manually. In (Grishman 1997), templates are used to guide the knowledge acquisition process. A template is a set of slots and fillers, not very different from the feature-value representations used in CBR. In order to fill a template, the annotations are classified as belonging to one or more of the slots in a template. Often this classification is supervised, and relies on the presence of training examples in the form of unstructured cases that have been mapped to equivalent template representations by experts. As a further step, IE systems also attempt to capture domain specific rules (Grishman 1997) that allow the system to perform inferencing with the objective of filling in missing slots or expanding the user query. In the classification context, systems have been built to classify text that rely entirely on a system of handcrafted rules (Hayes et al., 1990). While there have been a few attempts to integrate IE with TCBR (Bruninghaus & Ashley 2001), to our knowledge, no TCBR system to date has used a combination of annotations and hand-crafted domain rules.

The second approach towards striking the middle ground in TCBR is using statistical techniques to facilitate the journey from a bag of words to a set of underlying concepts. The underlying hypothesis is that these set of concepts are better descriptors of the underlying content, than the surface word-level representation. A critical problem in dealing with text is the problem of word choice variability, or different surface representations that achieve the same communication goal. Two common problems highlighted in this context are synonymy and polysemy. Thus a textual representation in the form of bag of words is at best a noisy representation of the underlying knowledge content. The central thesis motivating statistical techniques is that textual cases are less "noisy" when modelled in terms of concepts rather than words. The main strength of this approach is that in most cases, the concept extraction is automatic, with little or no manual intervention. This has inspired a significant amount of research within Text Mining and IR communities, much of which has later been adapted for TCBR tasks. Instead of attempting to discuss each technique individually, we organize these techniques into a taxonomy and present a unified view in Section 2.2, which allows us to compare and contrast these approaches.

To summarize, there are two broad trends towards attaining the middle ground: the first is based on Information Extraction and relies strongly on linguistic knowledge; the second is based on abstraction of words to concepts, and is founded on statistical principles. In this thesis, we focus on the latter. However, it is important to note that these two philosophies of achieving a middle ground are not necessarily in competition with each other; in fact they can complement each other in more ways than one. Firstly, instead of operating directly over a bag of words, statistical approaches can use phrases or attribute values extracted by IE techniques as staring points for concept learning. Mao and Chu (2007) show that concept learners operating over domain specific keyphrases achieve significantly higher accuracy in classification tasks compared to using words on their own. Secondly, statistical learning techniques can facilitate automating Information Extraction tasks, or at the least assisting domain experts in such tasks as well. Several IE approaches model IE as a token classification task (Kushmerick et al. 2001). The text is split into several tokens, and standard classification algorithms are employed to assign these tokens to one of the slot fillers. Several other subtasks of IE like phrase extraction, identification of contextually related word groups and extraction of attribute value pairs employ statistical approaches well.

In Section 2.2 we review statistical approaches to concept learning, and argue that factor analytic approaches like LSI are favourably positioned amongst existing techniques, in alleviating the knowledge acquisition bottleneck associated with TCBR. We also make a critical note of LSI shortcomings that motivates much of our work reported in later chapters. In Section 2.3 we provide a comparative analysis of architectures that facilitate effective and efficient retrieval in TCBR. In particular, we highlight some advantages associated with Case Retrieval Networks in this context. We also identify weaknesses of CRNs in the presence of non-sparse textual representations, that motivates work reported in Chapter 8. Finally, we take a quick look at existing visualization approaches that may allow users and experts to interpret the acquired knowledge easily. We identify shortcomings of these approaches that inspire our preliminary research reported in Chapter 3.

2.2 From Words to Concepts

While the traditional BOW paradigm has been the easiest prescription for building most text retrieval applications, it is crippled with several limitations, most of which stem from the fact that BOW fails to exploit associations between words and fares badly in handling both synonymy and polysemy. Thus a query on "operating systems" may fail to retrieve documents on Windows XP or Linux if the words "operating" or "system" are not present in those documents, and polysemous words like "Jaguar" cannot be effectively disambiguated based on usage context. In classification tasks, using BOW results in very poor generalization over the knowledge present in training data. Furthermore, infrequent words that are representative of one case are often filtered out, if they do not occur frequently in the rest of the corpus. These limitations of BOW have motivated research into modelling textual content using concepts rather than words. From a statistical machine learning perspective, the thesis is that combinations of words can be abstracted out to form concepts, which are more representative of the underlying meaning thus facilitating more effective retrieval, or more discriminative with respect to the defined categories in the classification context.

Based on the representation of concepts, statistical techniques that mine concepts can be broadly grouped as follows:

1. **Word Clustering/Distributional Clustering.** In this approach, concepts are modelled as clusters of related words. One popular version is a probabilistic approach called Distributional Clustering, which was introduced in the early nineties for automated thesaurus creation, and later found application in text classification (Baker and McCallum 1998). In the classification context, the key idea is to extract features comprising words that are contextually similar and contribute similarly to classification. This has the positive effect of reducing data sparseness and redundancy, and facilitating selection of most representative features. Several distributional measures have been proposed for measuring similarity between two word distributions, the most notable of which is the KL divergence (Manning & Schutze 1999). In the TCBR community, researchers have extended this idea to the unsupervised case where class labels are not present (Wiratunga et al., 2006, Patterson et al. 2005). The basic idea behind these extensions is to substitute the class label by a separate set of seed words, against which the distributional similarities of other words are conditioned.

2. **Factor Analysis.** Documents are similar when they have similar words, but words are similar when they appear in similar documents. Approaches based on factor analysis attempt to find a mathematical solution to the above circularity. In the field of text retrieval, the most widely researched and used factor analytic technique is Latent Semantic Indexing (LSI) (Deerwester et al., 1990). LSI is founded on a vector space representation of documents, which can be mapped onto a term-document matrix. The key step in LSI involves subjecting this matrix to Singular Value Decomposition (SVD), a linear algebraic technique that extracts a set of orthogonal bases for this space. In essence, these basis vectors, also called concepts, are nothing but linear combinations of the original terms. These concepts are ranked according to their importance. Considering only the most important ones (based on some threshold) and ignoring the rest, we can obtain reduced dimensional representations of both words and documents. The revised representations using concepts rather than words have been shown to improve retrieval performance (Deerwester et al., 1990). The essential thesis

behind LSI is that by representing documents and words in the concept space, we can recover from "noise" due to word choice variability, and thus have more robust estimates of the underlying meaning. Further technical details are discussed in Chapter 4. One of the remarkable aspects of SVD is its ability to represent both terms and documents in the same "concept space". This distinguishes it from its historical precursors based on eigen-analysis that can handle only the term space or the document space at a time. While initially designed for unsupervised retrieval, LSI has also been applied to supervised classification tasks (Gee 2003, Zelikovitz 2004, Zelikovitz & Hirsh 2001), where LSI was agnostic to class-knowledge. The inability to exploit class knowledge in supervised tasks is one of the critical drawbacks of LSI. Ever since the seminal paper of Deerwester et al. (1995), several approaches and extensions of LSI have been proposed that are similar in spirit to the basic idea. Recently, a probabilistic version of LSI called PLSI has received much attention in the text mining community. Other relatives include Non-Negative Matrix Factorization (NMF) (Lee & Seung 2001) and Semi-Discrete Matrix Decomposition Reference (Kolda & O'Leary 1998).

3. **Rule Learners** Unlike probabilistic or factor-analytic approaches that generate numeric representations of concepts, rule learners produce symbolic concepts. These approaches have been widely used for supervised classification tasks. The two broad classes of rule learners are decision tree learners and inductive rule learners. A decision tree classifier is a tree whose internal nodes are labelled by the terms, each branch emanating from a node checks for the presence or absence of that term. Each leaf node is assigned a class label. A test document d_i is recursively tested for the weights that words in internal nodes have in d_i, until a leaf is reached, whose class label is assigned to d_i. A DT is constructed from training examples using a divide and conquer approach: (a) check if all training documents have the same class label; (b) if not, choose a feature and partition the set into two subsets, such that each subset has the same value for that feature (often only presence/absence is considered). The process is recursively performed on all subtrees, till each leaf is "pure", i.e. contains documents drawn from

one class, which is chosen as the label for that node. The key step here is to choose partitioning attributes judiciously; information theoretic measures such as Information Gain are often used to identify good choices. For our present discussion, we note that we can read out a "rule" induced by a DT, by treating a conjunction of all nodes from root to leaf (leaf excluded) as the rule antecedent, and the class label associated with the leaf as a consequent. Fully grown DTs often suffer from overfitting; methods to prune overly specific branches have been proposed (Breiman et al., 1984).

Inductive rule learners have the same goal as DT learners, but they tend to produce more compact rules. The basic idea is to start from a set of highly specific rules that cover all the training data. So, each training document leads to a rule which has words in the document in its antecedent, and the class label of that document as the consequent. Unsurprisingly, this leads to overfitting; the rule learner now generalizes these rules by removing or merging clauses, to maximize rule compactness, while retaining "coverage". A further step of pruning is now applied for "global optimization" to strike a balance between mimimizing error on the *entire* rule set and maximising generality. RIPPER(Choen & Singer 1999) and Information Extraction extraction approaches like TextRise (Nam and Mooney 2001) are founded on this idea. Unsupervised variants of rule learners have recently been studied. In Textual CBR, (Wiratunga 2006) proposed Propositional Semantic Indexing (PSI), which uses rule induction to extract new features that are logical combinations of existing features. FEATUREMINE (Zelikovitz 2003) extracts simpler and less granular rules based on pairwise comparisons of all feature-pairs.

4. Formal Concept Analysis (FCA). Founded on a theoretical framework conceived in the eighties (Wille 1982), FCA has only recently been applied to mining concepts in texts (Cimiano et al. 2003). FCA takes as input a term-document matrix, which is referred to as defining the context. The output is a set of concepts. Intuitively, a concept is essentially a grouping of a subset of documents, say D, with a subset of terms, say T, such that the only terms that documents in D share between them are T, and conversely, the only documents in which all terms in T appear are D ; this is called the closure property. The set of all concepts, when ordered by set inclusion, satisfies the properties of a complete lattice, which is called a Concept (or Galois) lattice. It may be noted that concepts have been alternately referred to as closed itemsets in data mining, and maximal bipartite cliques in theoretical computer science. One attractive feature of FCA is that clustering is not done separately on the word and document spaces. Rather, each concept is defined by a set of words *and* a set of documents, the former providing an explanation for the grouping of the latter, and vice versa. Thus FCA–generated concepts can be easily interpreted. However, one critical limitation is that it requires a binary-valued term-document matrix as a starting point. Thus relations captured using real valued weights or smoothed versions of this matrix cannot be accommodated. Furthermore, rigid closure requirements mean that it may fail to identify "approximate" concepts, which could be more meaningful and general than the extracted ones. When operating over large and sparse term document matrix, this often results in a large number of meaningless groupings of terms and documents. Furthermore, unlike LSI, the extracted concepts are not ranked according to their importance. It is also not clear how to extend this idea to accommodate class knowledge of training instances in supervised tasks.

5. Implicit Concepts Defined By Hyperplane Separators. In addition to the four concept mining approaches described above, there is a family of "black box" approaches that include neural networks and kernel methods like the Support Vector Machines (SVM). The latter, in particular, has been shown to yield state-of-the-art results in supervised text classification tasks (Joachims 1998). These approaches are founded on the vector space model, and attempt to learn decision boundaries that separate classes in the original feature space (as with neural networks (Mitchell 1997)) or a higher dimensional version of the original space, where classes become linearly separable (as with SVMs). While the other approaches described so far produce concept representations that can be accessed and exploited for various text mining applications, neural nets and SVMs represent concepts implicitly using clusters of objects (documents) bounded by hyperplane separators, which are linear combinations of features that separate classes from each other. Intuitively, a concept could be distributed across various regions of feature space, and each region is bounded by a complex decision boundary, a polygonal approximation of which is constructed by the separating hyperplanes. Thus, the representation of concepts is scattered across a set of geometrical surfaces; the concepts are implicitly modelled within the geometry of the space to solve the classification task, but cannot be easily interpreted, accessed or exploited for other tasks. This can be contrasted against factor analytic approaches like LSI, where each concept is a linear combination of features, that can be accessed and used for tasks like feature extraction or mining word similarity. Another limitation of neural nets and SVM is that they have been tailor-made to handle supervised tasks, and do not lend themselves comfortably to unsupervised extensions.

While the above list is representative of the established techniques, it is by no means exhaustive. In particular, it may be noted that all five approaches mentioned above start from BOW representations, and thus knowledge of the order in which words appear in the text is lost. One approach to address this problem is using syntactic phrases based on linguistic knowledge like WordNet (Miller 1995). The five approaches mentioned above can be easily extended to mine abstract statistical concepts over Bag of Phrases, instead of using BOW as the starting point.

For ease of analysis, we present below a taxonomy that groups approaches built using the above formalisms, based on nine axes. This allows us to evaluate the relative advantages and disadvantages of these approaches, and assess their suitability for a given task.

Axis 1: Class knowledge. Based on how strongly the approach relies on knowledge of class labels, techniques may be classified as:

Supervised. These techniques heavily rely on class knowledge, and some of them cannot be easily extended to unsupervised tasks. Examples are neural nets and SVM.

Rule learners like RIPPER, decision trees and distributional word clustering approaches also belong to this category.

Unsupervised. These techniques can learn concepts without relying on class labels of training data. Examples are factor analytic techniques like LSI, rule learners based on association rule discovery, and unsupervised extensions of distributional clustering approaches like (Patterson et al 2005, Wiratunga et al 2006).

In addition most supervised concept learners, can also be operated in a "semi-supervised" mode, whereby they use both labeled and unlabeled data for training. The strength of such approaches is that they can compensate for lack of sufficient labelled training instances, by using unlabelled ones, which are often available aplenty. Two approaches are Expectation Maximization (EM) and co-training (Feldman & Sanger 2007). In EM, as a first step, a model is learnt based on training data. Next, in what is called the E step, unlabelled documents are classified by the current model. In the M step, the model is trained over the combined corpus. E and M steps are repeated till convergence is obtained. Co-training is based on bootstrapping where unlabelled documents classified using parts of the training documents (say abstracts or meta-level tags) are used for training the classifier based on the remaining parts (say the body), and vice versa. Both EM and co-training have shown a reduction of around 60% in the amount of training data needed to produce the same classifier performance (Feldman & Sanger 2007).

Axis 2: Knowledge Source. Based on their source of knowledge, concept learners are of two types. While introspective techniques rely entirely on the given data (training data, in supervised cases), those that use background knowledge can exploit knowledge from external knowledge sources as well. Most techniques discussed above can be used to accommodate background knowledge, or disregard it; so this is more a classification of tools built on these learners, rather than the learners themselves. WordNet has been used to provide linguistic knowledge of word associations in TCBR (Chakraborti et al., 2003). Recently, the Web, and in particular the Wikipedia has been used to acquire knowledge of "semantic relatedness of words and phrases (Gabrilovich and Markovitch 2007). A third interesting way of incorporating background knowledge is transductive learning, proposed by Zelikovitz and Hirsh (2001) in the context of supervised classification tasks. The basic idea is to pool unlabelled documents along with labelled ones while doing LSI. In the TCBR context, the use of domain specific knowledge is critical to system performance. Mario Lenz (1998) identifies seven knowledge layers in TCBR, and surveys TCBR systems that incorporate domain specific glossaries and feature values. In the current work, we will focus on introspective approaches alone, so that we are free of any underlying assumptions about availability of background knowledge. We incur no loss of generality, since linguistic knowledge or knowledge of domain specific feature values and their associations, if available, can easily be integrated to augment the knowledge mined by the statistical approaches proposed in this thesis.

Axis 3: Knowledge Richness. This is related to Axis 2, but here we are concerned with the *representation* of features within the system, rather than the source from which they were derived. At one extreme are knowledge light systems that rely on bag of words; at the other extreme are knowledge rich systems that use domain specific feature values extracted semi-automatically, typically with significant manual intervention. Unsurprisingly, there are several possibilities between these extremes, as exemplified by the seven knowledge layers of Mario Lenz. Most statistical concept learning approaches make no underlying assumptions about the knowledge richness of features. However, not all approaches scale well over the large dimensionality associated with BOW (see Axis 9 below). Knowledge rich approaches help reduce dimensionality, and also allow for more meaningful concepts to be inferred. This is particularly important in the case of techniques like FCA which tend to generate noisy concepts over large sparse BOW representations. Furthermore, while most concept learners are aided by feature selection strategies using measures like Information Gain (Mitchell, 1997) to reduce dimensionality over supervised classification tasks, feature selection over unsupervised domains is less straightforward. In such situations, most learners are likely to benefit from availability of knowledge rich features. The disadvantage with knowledge rich features is in the additional manual effort involved in acquiring them. Given our focus on completely automated knowledge acquisition, we will assume BOW as our starting representation throughout this thesis. It is important to note, however, that all our approaches will benefit from better domain-specific feature engineering.

Axis 4: Explicitness/Interpretability. This is determined by the formalism used to describe statistically mined concepts. In vector space theoretic approaches, concepts are viewed as linear combinations of features (as in LSI), or as combinations of convex regions in the vector space bounded by linear decision surfaces (as in neural nets or SVM). In Probabilistic models, like distributional word clustering, PLSI, and probabilistic mixture models like Latent Dirichlet Allocation, concepts learnt are probability distributions over the feature space. Rule-based models, like decision trees and DNF learners model concepts as rules whose antecedents are logical combinations of features, and consequents are class labels or features in the supervised and unsupervised cases respectively. FCA may be alternately viewed as founded on the rule-based model, since concepts mined by FCA have their counterparts in closed itemsets mined by association rule mining algorithms. The question we ask here is: Can the knowledge acquired by the system be interpreted meaningfully by humans? While there is an element of subjectivity based on user profiles and representations that they are comfortable with, there is not much of a disagreement about the extremes: rule based learners are the easiest to interpret, while neural networks and SVM fare miserably. Between these "white-box" and "black-box" extremes are approaches that we call "grey box". Factor analytic approaches like LSI fall into this category. While the numbers involved in the linear word combinations may not be easily understood, LSI has the advantage that terms and documents are projected onto the same concept space, and this allows for easy visualization. Also the concepts are ranked in accordance with their importance, so a visual inspection of only a top few important concepts may reveal interesting patterns that might have otherwise required a large number of equivalent rules to elicit. It may be noted that in addition to interpretability, some authors (Hilario & Kalousis 1999) identify a second criteria called transparency, which refers to whether the principle behind the method (as opposed to the concepts it generates) is easily understood. SVM and related kernel approaches are particularly hard to grasp, whereas rule based techniques and word clusters are the easiest. LSI and neural nets are moderately hard. For the purpose of the current thesis, we do not lay much emphasis on transparency, as long as system generated concepts are accessible and lead to improvements in system effectiveness.

Axis 5: Ease of Incremental Update. This is measured by the number of past training examples that must be reprocessed to accommodate a new example. Given a new training instance, incremental concept learners can update the set of generated concepts, without having to run the learner on past examples all over again. While incremental learning is often desirable in real world situations where the learner has to handle a steady stream of incoming data, non-incremental learners are often more effective over batch data, since they can better exploit the global properties of the collection to arrive at "enlightened" (Hilario & Kalousis 1999) concept representations. Another limitation associated with certain incremental learners is their sensitivity to order of presentation of the training instances. SVM, neural networks and decision tree-based learners are all non-incremental. However (Utgoff 1989) and (Brodley & Utgoff 1995) present incremental variants of univariate and multivariate trees respectively. LSI has been typically used in the non-incremental mode, though this limitation has been successfully addressed by various fast SVD update strategies; (Berry et al., 1995) for example make a comparative study of six such update algorithms. It may be noted that unlike statistically inferred knowledge, linguistic and background knowledge do not require frequent updates.

Axis 6: Ease of Use. This is characterized by the number and complexity of model or runtime parameters that need to tuned by the user. Decision trees require very few parameter settings; and SVM has been shown to be fairly robust to parameters under certain conditions (Joachims 1998). Neural Networks are possibly the worst, since the number of internal nodes which determine the complexity of the learnt decision surfaces, as well as the learning rate, momentum and initialization of weights need to be tuned to suit the application in question. The case for LSI has no clear consensus. While no parameters are involved in the factor analysis process per se, many researchers have pointed to the difficulty in choosing the right number of concept dimensions. We argue that LSI cannot be singled out in this regard, since a disguised version of the same concern is shared by most other approaches as well. For example, the support and confidence thresholds in association rule mining, the number of word clusters and seed-words that are appropriate in distributional clustering, and the level of decision tree pruning to achieve the right generalization, are all parameters that need tuning and play a role not very different from the LSI dimensionality setting. Alternatively, we can regard this parameter setting as aiming to achieve the right bias-variance tradeoff (Hastie et al., 2001). Low LSI dimensionality corresponds to a high-bias low-variance learner that would generate simple highly constrained models that are insensitive to data fluctuations. Using large number of LSI dimensions, on the other hand, corresponds to a low-bias high-variance learner that can generate arbitrarily complex models, often running the risk of overfitting the data. Another aspect that determines the ease of use of a concept learner is the number of underlying assumptions. For example using association rules or FCA requires that one starts with an integer valued term document matrix, and some additional techniques are required to carry out the mapping from real-valued to integer-valued matrices, in a meaningful way. SVM is popular since it has the flexibility of handling a large number of dimensions, and its performance is not critically dependent on feature selection. Current implementations of LSI can scale up comfortably to handle very large number of dimensions.

Axis 7: Ease of integration with instance based techniques. This is an important criterion in relation to TCBR systems. CBR systems are instance based learners that support lazy learning, and incremental and local updates to knowledge. In addition they are easily interpretable and knowledge maintenance is facilitated by the availability of rich competence models. For TCBR tasks, we would prefer concept learners that yield representations that maintain these advantages. LSI has a distinct edge here, since it can generate revised vector space representations of the underlying cases, either as an approximation of the term-document representation in the original feature space, or as a reduced dimensional representation in the concept space. This will be discussed in more detail in Chapter 4. SVM and neural networks are the worst in this regard, since they yield no representations whatsoever, that can be exploited by instance-based techniques. FCA, decision trees and other rule-based systems are not naturally suited for CBR style representations, though the learnt concepts may be mapped indirectly to revised case representations, often with loss of information. In the PSI scheme (Wiratunga et al., 2005a) for examples, inferred rules were used for feature generalizations, which in turn led to revised cases.

Axis 8: Support for additional tasks like Word Similarity Mining. In the TCBR context, we are often interested in not just revised representations of concepts, but also an explicit knowledge of word (feature) similarity. This allows experts to independently examine the word similarity knowledge and suggest refinements. While techniques like LSI can be tailored easily to mine similarity knowledge, it is less straightforward to extract word similarity from rule-based learners like decision trees. SVMs are the worst in this regard as they permit no easy access to their underlying concepts, which are critical in determining how similar words are.

Axis 9: Efficiency. This is composed of two parts: training time and execution time. While actual training times are critically dependent on parameter settings which typically trade off effectiveness against training times, we can still compare approaches by expressing training times as a function of n, the number of training instances and p, the number of predictive features. Rule based techniques like ID3 have complexity of the order of $O(n^2p)$ in the worst case, whereas LSI's complexity mainly arises out of the SVD step which has worst-case complexity $O(\min(np^2, pn^2))$. However, SVD implementations can be speeded up considerably, by requiring that only the first few important concepts need to be considered, and by exploiting sparseness of the term-document matrices. Neural nets are the slowest in terms of training time. While complexity analysis over neural nets is generally problematic because of the difficulty in predicting the number of iterations required to converge, it has been shown that the worst case complexity is exponential. Hinton approximates training time on a neural network to be approximately $O(N^3)$ where N is the number of weights in the network. In comparison, SVMs are faster to train, the worst case complexity is $O(n^3)$, though further extensions reduce the average case complexity (Chin 1998). The execution times of different algorithms are less critical for the current comparison, since in TCBR the unwritten assumption is that all approaches will finally yield revised case representations, which will be processed using near neighbour approaches.

In Table 2.1, we summarize the strengths and weaknesses of the five concept learning approaches, using a subset of the above dimensions as basis for comparison. Some dimensions like use of background knowledge or knowledge richness, have been excluded for comparison, since they characterize TCBR systems as a whole, as opposed to underlying formalisms. In other words, no statistical approach has inherent restriction to the use of background knowledge, or of knowledge rich units like phrases instead of bag of words.

In this thesis, factor analytic approaches, in particular LSI, play a substantial role in automatic knowledge acquisition for CRNs. The choice of LSI is driven by its ability to generate rich representations for both documents and words in terms of a common set of underlying concepts thus facilitating acquisition of both relevance and similarity knowledge. Thus LSI integrates easily with instance based learners. Other advantages include availability of easy update strategies, efficiency of retrieval, and reliance on very few parameters to be tuned. However LSI is limited by its inability to exploit class knowledge in supervised classification domains; we address this limitation in Chapter 5. In Chapter 6, we present an approach to mine similarity knowledge that relaxes certain mathematical constraints imposed by LSI. We show that our new approach not only leads to better retrieval effectiveness, but is also better than LSI at explaining, as opposed to merely estimating, associations between features. In Chapter 8, we address run-time efficiency implications of LSI and present novel retrieval formalisms that facilitate fast retrieval over relevance and similarity knowledge mined using LSI.

Table 2.1 Comparison of concept learning approaches

	Word Clustering	**Factor Analysis**	**Rule Learners**	**FCA**	**Implicit Concepts (SVM)**
Class Knowledge	Originally supervised, recently unsupervised extensions proposed	Unsupervised (supervised extensions reported)	Supervised	Unsupervised	Supervised
Interpretability	Average	Average	Very Good	Good	Poor
Transparency	Average	Average	Very Good	Average	Poor
Ease of Use	Average	Average	Good	Average; Lack of support for real valued entries	SVMs very good, Neural nets the worst
Ease of Integration with instance based techniques	Average	Very Good	Average	Average	Poor
Support for Word Similarity Mining	Very Good	Very Good	Average	Average	Poor
Training Efficiency	Average	Average	Good	Average	Neural Nets slowest, SVMs average

2.3 Scalable TCBR Architectures

It is not enough to have a good representation of textual cases, we also need an architecture that embeds the different knowledge containers, and facilitates effective and efficient retrieval over these cases. We identify the following characteristics that are desirable for a TCBR architecture:

Efficiency: The architecture must allow for fast retrieval in the face of large number of cases, by avoiding exhaustive search over the casebase. In addition, TCBR domains are typically characterized by very high dimensionality, originating from the large number of words (alternately phrases or word groups) that define the feature space. So it is imperative for the system to be able to scale well to counter the "curse of dimensionality".

Retrieval Effectiveness: This is a broad goal, which encompasses several sub-goals. Firstly, the architecture must be flexible, so that the retrieval results can be tailored to specific search needs of the user. In IR, precision and recall are most widely used to evaluate retrieval effectiveness. The former measures the fraction of the retrieved results that are relevant to the query, while the latter estimates the proportion of all relevant documents in the collection that were retrieved. At one extreme, a TCBR system should be able to support a very focussed (precision-centric) search where all search terms or phrases appear in the retrieved documents. At the other end, it should also support a lenient (recall-centric) search where even documents that contain none of the query terms (say "gulf", "oil" and "war") but are still relevant (say on "Middle-east") are retrieved. Lenz (1999) identifies an additional pair of criteria, namely completeness and correctness. The former implies that every sufficiently similar case in memory will be found during retrieval, while the latter is ensured through a secondary selection over the retrieved cases. We will not lay strong emphasis on these criteria, primarily because in a practical setting, it is extremely difficult to formally evaluate systems according to these measures. Moreover, the measures in a sense mandate a two step retrieval, the first ensuring completeness and the second correctness, which may be restrictive when dealing with techniques that rely on precomputed indices to do a one-shot pruning of the search space. In certain situations it may suffice to obtain an appropriate ranking of cases, and the actual similarity scores do not matter; this aspect may be used to eliminate redundant computations, thereby improving efficiency of retrieval. To sum up, the efficiency of the system should not be at the cost of its effectiveness, though concessions may be made specific to the needs of the retrieval task at hand.

Few Underlying Assumptions: Approaches to improve retrieval efficiency often rely on assumptions such as triangle inequality, existence of ordered attributes or "skewed"ness of data distribution (Chavez et al., 2001). These assumptions restrict the applicability of such techniques. We would thus prefer architectures founded on formalisms with minimal underlying assumptions. In particular, the architecture must support efficient retrieval over commonly used distance metrics like the Euclidean Distance and the cosine similarity.

Explicit Knowledge Containers: The effectiveness of a TCBR system is typically governed by two main knowledge containers, similarity and relevance knowledge. Keeping in line with the CBR philosophy, explicit access to these knowledge containers is desirable. It may be noted that the role of a third CBR knowledge container, namely adaptation knowledge, has been of peripheral concern in practical TCBR systems, though some approaches towards acquiring adaptation knowledge for textual cases have been explored very recently (Gervas et al., 2005).

2.3.1 TCBR Architectures

We classify the mechanisms explored in CBR to facilitate efficient retrieval into the following broad categories.

Partitioning Methods. Organization of cases in the case-memory is based on similarities between cases. Cases similar to each other are grouped in the same "bucket" – this can be viewed as a generalization of hashing in one dimension. A k-d tree (Wess et al. 1993) is a *k*-dimensional binary search tree that groups cases into non-overlapping partitions, each partition consists of cases that are similar according to a given similarity measure. At retrieval time, only sub-trees likely to contain a prospective case are traversed, thus saving on similarity computation with cases distant from the query. To ensure that no relevant cases are missed out, similarity bounds are computed at run time to decide whether cases in adjacent partitions need to be considered. The main limitation of k-d trees is their assumption of ordered attributes; also they do not scale well when large number of dimensions are used for indexing. Inreca Trees (Bergmann 2002) are an extension of this idea that can handle unordered domains and allow for n-ary splits based on attribute values.

Pivot Based Methods. These approaches were inspired by algorithms conceived in early seventies (Chavez et al., 2001) to speed up near neighbour search in large metric spaces. Distances over a metric space obey the triangle inequality, and this property is exploited to eliminate redundant computations at retrieval time. The idea is to select a few documents as pivots, and pre-compute distances of all remaining documents to the pivots. The target case is then compared only with the pivot cases, and triangle inequality is used to eliminate distance computations to cases that can never satisfy the search criterion. Variants of this idea use a tree data structure where partitions of the casebase are assigned to pivots they are closest to, and these partitions are then split recursively using further pivots at each intermediate node. The cost of traversing the tree index is referred to as internal complexity, whereas the cost of exhaustive search at the leaf nodes is referred to as external complexity. The efficiency improvements obtained are critically dependent on the right trade off between these two complexity estimates. Two techniques in the CBR literature may be regarded as close relatives of pivot-based search. Smyth and McKenna (1999) present a footprint based retrieval algorithm where a set of footprint cases act as pivots. Footprint cases are those that provide a good "coverage" of the casebase. Intuitively these cases may be treated as a small fraction of the cases that can solve the same set of problems as the entire casebase. Each footprint case is associated with a set of related cases, that either solve the footprint case, or are covered by it. The authors propose a two stage retrieval. First, the target cases are compared against the footprint cases. In the second step, the cases related to the most similar footprint case are searched. Closely related is the idea of Fish and Shrink (Bergmann 2002) where similarities between a subset of cases is pre-computed using the relatedness of what the authors call "aspects". At query time, if a case is found to be far away from the target, several neighbours of that case can be shown to be ineligible as well, using the triangle inequality. This saves redundant computations which is especially significant in this case, since aspect-based similarities are computationally demanding. A limitation of most pivot-based approaches is their reliance on assumptions like triangle inequality.

Spreading Activation Spreading activation methods view case memory as an interconnected network of nodes that capture the association of cases with their attribute values. Target attribute values trigger a spreading activation in the network, resulting in activation of cases similar to the target. Spreading Activation based approaches have been proposed in Brown et al. (1994), Wolverton and Hyes-Roth (1994) and Lenz (1996). The CRN belongs to this category, and has been particularly favoured by the TCBR community because of several reasons. Firstly, CRNs are efficient. Lenz (1996) (also see Lenz et al. 1998) has successfully deployed CRNs over large casebases containing as many as 200,000 cases. The applicability of CRNs to real world text retrieval problems has been demonstrated by the FALLQ project (Lenz et al., 1997). Balaraman and Chakraborti (2004) have also employed them to search over large volumes of directory records (upwards of 4 million). More recently spam filtering has benefited from CRN efficiency gains (Delany et al., 2004). Secondly, they are flexible and allow different retrieval needs like high precision or high recall to be addressed within the same formal framework. Thirdly, they have no underlying assumptions about the nature of attributes or distance functions. Finally, both knowledge containers of CRNs, namely knowledge about how terms in a domain are related to each other (similarity knowledge), and knowledge about relatedness of terms to cases (relevance knowledge) are explicit and there is a neat separation between them, allowing them to be independently acquired, revised and manipulated. Since much of the later thesis is founded on the CRN architecture, we take a closer look at CRNs in Section 2.3.2.

Additional techniques. In addition to the above basic approaches, Stanfill and Waltz (1986) have reported significant speed-ups using massively parallel SIMD architectures. Their approach uses a brute force sequential search, but parallelizes the computations involved. Many commercial CBR systems use smart dynamic SQL queries to successively narrow down search space, thereby also exploiting fast indexing strategies already built into commercial DBMS systems. Both parallelization and database-centric optimizations can lead to better performance of most approaches discussed above.

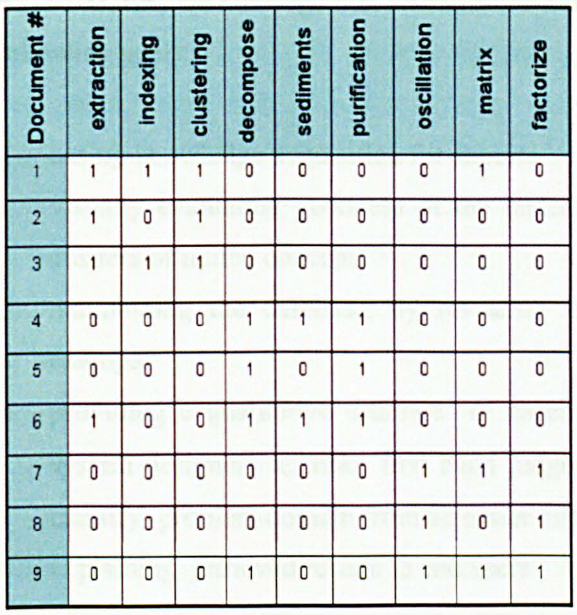
2.3.2 Case Retrieval Networks

In this subsection, we take a closer look at CRNs. To illustrate the basic idea we consider the example casebase in Figure 2.2(a) which has nine cases comprising keywords, drawn from three domains: CBR, Chemistry and Linear Algebra. The keywords are along the columns of the matrix. Each case is represented as a row of binary values; a value 1 indicates that a keyword is present and 0 that it is absent. Cases 1, 2 and 3 relate to the CBR topic, cases 4, 5 and 6 to Chemistry and cases 7, 8 and 9 to Linear Algebra.

Figure 2.2(b) shows this casebase mapped onto a CRN. The keywords are treated as feature values, which mapped to Information Entities (IEs). The rectangles denote IEs and the ovals represent cases. IE nodes are linked to case nodes by relevance arcs which are weighted according to the degree of association between terms and cases. In our example, relevance is 1 if an IE occurs in a case, 0 otherwise. The relevances are directly obtained from the matrix values in Figure 2.2(a). IE nodes are related to each other by similarity arcs (curved arrows), which have numeric strengths denoting semantic similarity between two terms. For instance, the word "indexing" is more similar to "clustering" (similarity: 0.81) than to "extraction" (similarity: 0.42). While thesauri like WordNet can be used to estimate similarities between domain-independent terms (Pederson et al. 2004), statistical co-occurrence analysis supplemented by manual intervention is typically needed to acquire domain-specific similarities.

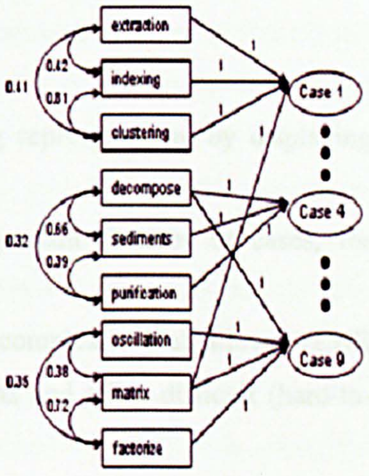
To perform retrieval, the query is parsed and IEs that appear in the query are activated. A similarity propagation is initiated through similarity arcs, to identify relevant IEs. The next step is relevance propagation, where the IEs in the query, as well as those similar to the ones in the query, spread activations to the case nodes via relevance arcs. These incoming activations are aggregated to form an activation score for each case node. Cases are accordingly ranked and the top *k* cases are retrieved.

A CRN facilitates efficient retrieval compared with a linear search through a casebase. While detailed time complexity estimates are available in (Lenz 1999), intuitively the speedup is because computation for establishing similarity between any distinct pair of IEs happens only once. Moreover, only cases with non-zero similarity to the query are taken into account in the retrieval process, thereby saving redundant computations.



Document #	extraction	indexing	clustering	decompose	sediments	purification	oscillation	matrix	factorize
1	1	1	1	0	0	0	0	1	0
2	1	0	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0
4	0	0	0	1	1	1	0	0	0
5	0	0	0	1	0	1	0	0	0
6	1	0	0	1	1	1	0	0	0
7	0	0	0	0	0	0	1	1	1
8	0	0	0	0	0	0	1	0	1
9	0	0	0	1	0	0	1	1	1

(a)



(b)

Figure 2.2 CRN for TCBR retrieval

CRNs are discussed in further detail in Chapter 8. Retrieval in CRNs tend to slow down when the sparseness of the original similarity relations is reduced, as is typical when concept learners like LSI are used to generate revised case representations. We propose a solution to this problem; the revised formalism is called the Fast Case Retrieval Network (FCRN) and forms the central theme of Chapter 8.

2.4 Visualising textual casebases

Deployment of a real world textual CBR system involves humans, either as experts or as users, and often as both. This makes it imperative to devise ways of effectively narrowing down the gap between the system and the human. In situations where the underlying similarity or relevance knowledge is mined using statistical techniques and encoded into the system as a set of numbers, we need effective ways of gaining access to the underlying knowledge. This is where visualization plays an important role. In the context of TCBR, we envisage that visualization can be potentially useful for realizing the following goals:

1. easing knowledge acquisition for experts
2. visually evaluating goodness of the underlying representation, by displaying clusters of mined concepts
3. maintaining the casebase, by revealing unimportant features or cases, for example
4. providing a qualitative estimate of casebase complexity that allows TCBR system designers to make first hand judgements and tell a difficult (hard-to-classify) problem domain from an easier one.
5. explaining retrieved results to end users

The first four are concerned with building and maintaining textual casebases, and are "off-line" activities in that they do not directly concern retrieval. In contrast, the fifth is an "on-line" activity, and is outside the scope of our research.

2.4.1 A Short Review of Related Work

Research in TCBR visualization has still not reached its critical mass. However, since most of TCBR concerns with regard to visualization are shared by researchers from text mining and CBR, it is worthwhile to take a close look at contributions from these fields.

One significant line of research focuses on grouping documents based on their similarity and displaying the similarity between the discovered groups (Feldman & Sanger 2007). As a first step, one of the classical clustering techniques, like partitional or hierarchical clustering is used to group documents having similar content. The second step is of displaying these clusters in a meaningful way. In case of partitional clustering, the clusters extracted are represented as nodes in a graph, and these nodes are linked together to reflect inter-cluster separation. An optimization technique like Simulated Annealing is used to arrive at the final graph configuration Also, each node representing a cluster is tagged with the most representative keywords of that cluster (Feldman & Sanger 2007). Hierarchical clusters are displayed using dendrograms, but they often get crowded with increasing number of documents. One solution is to use a hierarchical two-wise K-means algorithm such that each cluster is recursively partitioned into two subclusters. The tiling pattern arranges the leaf nodes of this tree along with keyword annotations for those nodes. While it provides insight into concepts associated with clusters packed close to each other, the arrangement fails to portray inter-cluster distances.

One disadvantage with most cluster visualization techniques is that the neighbourhood relations between adjoining clusters is lost. Self-organizing Maps address this limitation by mapping neighbourhood relations between high dimensional objects (cases) to a low dimensional topology. One prominent work founded on this idea is WebSOM (Feldman & Sanger, 2007). A very closely related theme is Multidimensional Scaling (MDS) (Hastie et al., 2001). MDS is a procedure to "rearrange" objects efficiently in a lower dimensional space, so as to arrive at a configuration that best approximates the distances observed between high-dimensional objects. The objects are iteratively moved around in the low-dimensional space, attempting to maximize the goodness of fit. CBR researchers have adopted a variant of this idea, metaphorically called Force Directed Graphs, which has been discussed in (Eades 1984), (Mullins & Smith 2001).

Parallel co-ordinates, conceived by Inselberg (Inselberg 1985) is a third way of representing high dimensional cases in two dimensions. In contrast to the Cartesian co-ordinate scheme where attributes are mutually perpendicular, parallel co-ordinates assigns a vertical axis to each attribute, and evenly spaces out these axes horizontally. The values that an attribute can take are plotted on the corresponding axis. Any given case is represented as a polygonal line laid out across the axes, such that each line segment connects two attribute values of that case. Viewed as a whole, such a plot is expected to reveal some coherent patterns arising out of correlations between cases, as well as features. One limitation in the context of text is the large number of dimensions which lead to overcrowding of axes. Moreover, because of data sparseness typical with text, most of the plot would be wasted portraying relatively trivial associations. For example, two cases may appear similar because 90% of the features are absent in both.

It may be noted that most of these approaches may be used either to display document clusters or word clusters.

2.4.2 Limitations of Existing Visualization Schemes

From the TCBR standpoint, we note the following limitations of existing visualization mechanisms. Firstly, most approaches display either the feature space or the document space, but lack the ability to display both documents and words in the same space. Since document similarities can be accounted for by the similarities of their words, showing document and word clusters in relation to each other has better explanatory power, and enhances the usefulness of the visualization to experts. Secondly, most techniques are not very helpful in identifying redundant words or documents that do not contribute to casebase competence. Finally, it is not straightforward to gauge the complexity of the casebase using most existing visualization schemes. Ideally, we would like intuitive visual indicators such that complexity can be compared meaningfully across representations of the same casebase, and across different casebases. We present an approach to address these limitations in Chapter 3. We also present a quantitative measure of casebase complexity that is directly inspired by our visualization scheme.

2.5 Chapter Summary

We have taken stock of key TCBR challenges and surveyed the broad landscape of concept learning techniques that can potentially aid in automated knowledge acquisition. We have identified key dimensions that facilitate meaningful comparison of the suitability of these approaches with respect to the TCBR knowledge acquisition task. Factor analytic techniques like LSI have been shown to have several advantages in mining relevance and similarity knowledge for CRNs. However, their inability to incorporate class knowledge in supervised classification tasks is a significant drawback. Also, similarity knowledge mined using LSI lacks transparency, and is limited in effectiveness by its adherence to a set of mathematical constraints that can be relaxed for TCBR. Overcoming these limitations constitutes a key motivation of this thesis. We have also reviewed retrieval formalisms that can embed the acquired knowledge to facilitate efficient retrieval. In particular, CRNs were studied as a special case of spreading activation formalisms. We have seen that techniques like LSI can result in loss of sparseness which can, in turn, have adverse effects on efficiency of retrieval. This motivates a further contribution of our thesis in terms of retrieval formalisms to facilitate efficient retrieval in the face of non-sparse similarity and relevance knowledge. Finally, we look at visualization approaches that can help the knowledge engineer in having better qualitative insights into the characteristics of the domain, and facilitating maintenance tasks. We identify limitations of existing visualization techniques that need to be addressed. In the following chapter, we address these limitations, and present qualitative as well as quantitative ways of characterizing textual casebases.

Chapter 3

Characterizing Textual Casebases

To be blind is unfortunate indeed but to be without a staff is even worse, for the staff does much of the eyes' work. Sri Sri Thakur Anukulchandra (Indian saint)

The objective of this chapter is to examine factors that affect the effectiveness of TCBR approaches, with a special emphasis on classification domains. One important goal is to study inherent properties of textual classification datasets that determine how a CRN based on bag-of-words performs in relation to state-of-the-art classifiers like SVM. The analysis will also lead us to the challenge addressed in subsequent chapters; namely, that of acquiring knowledge automatically for CRNs with the goal of elevating their effectiveness to make them comparable to, or outperform, competing classifiers. This is important in the light of the comparative study in the last chapter, where we noted several strengths of instance based approaches relative to other classifiers. To ground our discussion, we will focus on six experimental datasets, which we use consistently through later chapters for evaluation. These datasets are reflective, if not representative, of the diversity encountered in dealing with TCBR datasets in supervised classification tasks.

We need tools in helping us probe into the nature of textual casebases. To date, there appears to be no consensus in the TCBR community about what constitutes "adequate" characterization of a casebase; most often the issue is closely coupled with the task at hand. Classification tasks have inspired work on evaluating classification complexity in non-textual CBR (Massie 2006), whereas very recently unsupervised measures of textual casebase complexity have been proposed (Lamontagne 2006, Massie et al. 2007). In this chapter, we adopt a two-step approach for characterizing casebases. In the first step, we render a visualization of the casebase that reveals the broad clustering patterns in terms of the underlying cases and features. The visualization provides a bird's eye view of the dataset. In the second step, we use the visualization to formulate a compression-based measure that quantifies the complexity of the casebase. While the measure itself can be used for unsupervised casebases, we will focus more on a version of it that captures the complexity of the classification problem associated with each dataset. We evaluate correlation of accuracy results obtained from different classifiers against these complexity measures to seek explanations on why certain approaches work well over some datasets, but not on some others.

Visualization and complexity evaluation proposed in this chapter are used only to the extent of facilitating better insight into the results of our experimental evaluation. However, we have seen in the previous chapter that visualization is significant in its own right in that it facilitates several TCBR tasks. Evaluating casebase complexity is important in facilitating the off-line tasks identified in Section 2.4, in that it provides a quantitative basis for assessing the suitability of a representation. While visualization and complexity evaluation have often been treated in isolation, our current understanding is that they often share similar goals, and may exploit similar mechanisms to realize these goals as well.

The rest of the chapter is organized as follows. Sections 3.1 and 3.2 introduce the datasets and classifiers used for our experiments. Section 3.3 presents a novel approach called "stacking" to visualize textual casebases. Section 3.4 shows how this approach can be extended to evaluate casebase complexity in unsupervised and supervised settings. Section 3.5 shows how our complexity metrics for the six datasets correlate with the accuracy figures reported by the classifiers described in Section 3.2. We highlight insights from this analysis that lead us to intuitions behind the novel contributions reported in subsequent chapters. Section 3.6 summarizes the chapter and takes stock of its main contributions.

3.1 Datasets Used

For experimental evaluation reported in this thesis, we use six text classification datasets. Four of these are email routing datasets created from the 20 Newsgroups (Mitchell 1997) corpus. The remaining two are Spam filtering datasets.

For creating the four routing datasets, one thousand postings of discussions, queries, comments etc. from each of the 20 Usenet groups covered by the 20Newsgroups datasets were chosen at random and partitioned by the newsgroup name (Mitchell 1997). Four sub corpuses were created:

1. SCIENCE from four science related groups
2. REC from four recreation related groups
3. HARDWARE from two hardware problem discussion groups, one on MAC and the other on PC
4. RELPOL, from two groups, one concerning religion, the other politics in the middle-east

Thus HARDWARE and RELPOL are binary classification problems, while SCIENCE and REC are four-class problems. The HARDWARE domain is interesting in that there are many terms like "drive" or "bus" which are shared by both PC and MAC, and hence fail to discriminate between the two classes. However "drive" combined with "vlb" indicates PC, whereas "drive" combined with "syquest" indicates Mac. It shows that class labels of training data must play an important role while inducing the co-occurrence patterns that can help us in disambiguating between the two classes. The RELPOL domain, on the other hand, presented challenges of a different kind. While bag of words do not yield impressive results, unsupervised approaches that yield clusters of features are useful in digging out concepts like "Palestine war" or "holocaust" that play a critical role in improving classification accuracy. Compared to

HARDWARE, class knowledge has a relatively smaller contribution towards accuracy improvements in RELPOL.

The two spam filtering datasets are

1. USREMAIL (Delany & Cunningham 2004a) which contains 1000 personal emails of which 50% are spam
2. LINGSPAM (Sakkis, et al., 2003) which contains 2893 messages from a linguistics mailing list of which 27% are Spam.

The datasets were split into equal sized disjoint training and test sets. Each split contains 20% of documents randomly selected from the original corpus, and is stratified in that it preserves the class distribution of the original corpus. Fifteen such train-test splits (alternately called trials) were obtained for each of the six datasets mentioned above. It may be noted that the documents were pre-processed by removing stop words (noise words) like functional words which are frequent throughout the collection and ineffective in discriminating between classes. Punctuations and special characters (quotes, commas and full stops) were also removed. Some special characters like "!", "@", "%", "$" were retained because they have been found to be discriminative for some domains (Sakkis, et al. 2003) Remaining words are reduced to their stem by using Porter's algorithm (Porter 1980). We use Information Gain (Mitchell 1997) to perform feature selection and use a maximum of 1000 top features for evaluation.

3.2 Classifiers Used

Sebastiani (2002) carried out a comparative experimental study of several text classifiers over five different versions of the Reuters collection, as well as OHSUMED and 20 Newsgroups collections. In summarising his findings, he reports that boosting-based classifier committees, Support Vector Machines, example based methods (which we refer to as instance based methods in this thesis) and regression methods deliver the best performances. They are closely followed by neural networks and online classifiers. Based on this study, we select a cross section of classifiers that are broadly representative of the best performing classifiers. The first approach is the most basic instance based approach that uses a CRN to realize a kNN-based retrieval. The second is the Extended Case Retrieval Network (ECRN), a novel approach that exploits a neural network style training algorithm to acquire knowledge for CRNs. The third is SVM, and the fourth is LogitBoost, a popular boosting-based ensemble approach. We then briefly look at using LSI for acquiring knowledge for CRNs, an approach that has been covered in detail in the following chapter. We also include for comparison Propositional Semantic Indexing (PSI) which has been proposed by Wiratunga, et al. (2005a) to acquire and explicate knowledge in TCBR applications.

1. kNN using CRNs. Instance based classification using *k* Nearest Neighbours is based on the idea of retrieving *k* cases most similar to a query case, and using the class labels of the similar cases to arrive at a prediction for the query (Sebastiani 2002). The effectiveness of this approach is largely determined by the representation of cases and appropriateness of the distance measure used to evaluate similarity between cases. We have seen in Section 2.3 that a CRN can be used to realize fast kNN based retrieval. The relevance and similarity knowledge of CRN determine the case representation and the relationship between features that comprise the vocabulary. The simplest CRN would be one that uses no knowledge of similarity between features, and that which uses binary relevance values, with a relevance value assuming a value 1 when a feature is present in a case, a value 0 otherwise. Thus, such a CBR system is no different from a basic IR system that is founded on the vector space model over BOW, and forms our baseline system for comparisons. The choice of binary valued representations is governed by the fact that unlike most IR systems operating over very large datasets, we use relatively smaller datasets over which frequency-based measures like tf-idf are not very robust. There are several distance measures proposed in literature (Manning & Schütze, 1999), of which the Euclidean distance and the cosine similarity are most popular. If cases are regarded as points in a vector space, the Euclidean distance between two cases is the geometrical (or straight-line) distance between their corresponding points. On the other hand, the cosine similarity is measured by computing the cosine of the angle between the two vectors representing the two cases. Two cases are maximally similar if their vectors are perfectly aligned, leading to a cosine similarity of 1. The original work on CRNs does not show how it can be used to realize the Euclidean and the cosine measures. In Appendix A.1, we show that such an extension can be made while retaining CRN's efficiency advantages..

2. **ECRN.** We proposed Extended Case Retrieval Nets (ECRN) (Chakraborti et. al., 2004) to integrate sub-symbolic learning mechanisms as exploited by neural networks, into the CRN. ECRN was motivated by the architectural parallels between CRN and a Multi Layer Feed-forward Network (MLFN). The name ECRN stems from the fact that in addition to the IE nodes and the case nodes present in a CRN, we now have a third layer pertaining to class labels of the textual cases. Henceforth we refer to these layers as IE, case and class layers respectively. The number of nodes in the IE layer is equal to the number of distinct features obtained after feature selection. Figure 3.1 shows a schematic of the architecture.

To initialize the ECRN, the relevance weights connecting IE nodes to case nodes are assigned binary values as in the baseline CRN described above. The weights connecting the case-layer to the class-layer are assigned binary values based on whether the case belongs to that class or not. During training, weights are modified to improve classification accuracy of the system. The training cases are fed into the IE layer one at a time and the classification output of the system is compared against the expected outcome. The desired output is a binary value 0 or 1 depending on the class to which the input case actually belongs. An error signal is computed between the observed and desired outcomes. This error is fed back to alter the set of weights that connect the IE layer with the case layer nodes. This is done by using a variant of the back-propagation algorithm used widely in training Multi-layer Feed-Forward

Networks (Mitchell 1997). A sigmoidal weighted aggregation (typical of back-propagation) is used at each node. Now the revised set of weights is used to classify the set of training documents again. If the mean squared error of the output layer is less than the previous iteration, the current set of weights is retained; otherwise we revert back to the old set of weights. The iterative process terminates once there is no improvement in accuracy: the set of weights obtained are retained for use in the classification phase. Once the ECRN has been trained for classification over training data, we evaluate its classification performance over test data. For this the new case to be classified is preprocessed – the IEs pertaining to the case are activated and the activation is propagated through the case nodes and the class nodes. This can be viewed as a two step process: in the first step, the nodes pertaining to cases that are similar to (nearest neighbours of) the incoming document are activated – in the second step, the activated cases vote for their respective classes via the connections to the output layer, and the results are aggregated in the output nodes. The class with the strongest activation is returned as the result.

In the context of the current chapter, it suffices to treat ECRN as a neural network approach for text classification. However, from a broader standpoint, it is worthwhile to take note of significant differences between a traditional neural network approach and the ECRN approach. Firstly, hidden nodes and weights of neural networks do not carry any specific interpretation with respect to the domain. This has led to the "black-box" view typically ascribed to neural networks. In contrast, the relevance arcs and all nodes in ECRN have definite correspondence to knowledge in the TCBR system. Furthermore, with traditional Neural Networks, the weights would have random initializations to start with. However since weights in ECRNs map onto relevances, we could launch the ECRN training using relevance weights meaningful to the domain. In (Chakraborti et. al., 2004), we present empirical evidence to show that this leads to significant reductions in the training time of ECRN, when compared to random initialization. A summary of empirical results obtained with ECRN is presented in Appendix A3. The idea of using algorithms like Back-propagation over a neural network initialized with prior domain knowledge is used in KBANN (Mitchell 1997). However in the framework of KBANN, the prior knowledge is a domain theory consisting of non-recursive propositional Horn Clauses; in contrast ECRN uses a knowledge-light initialization based on binary relevance values. This helps us preserve the CRN topology while allowing for relevance weight learning.

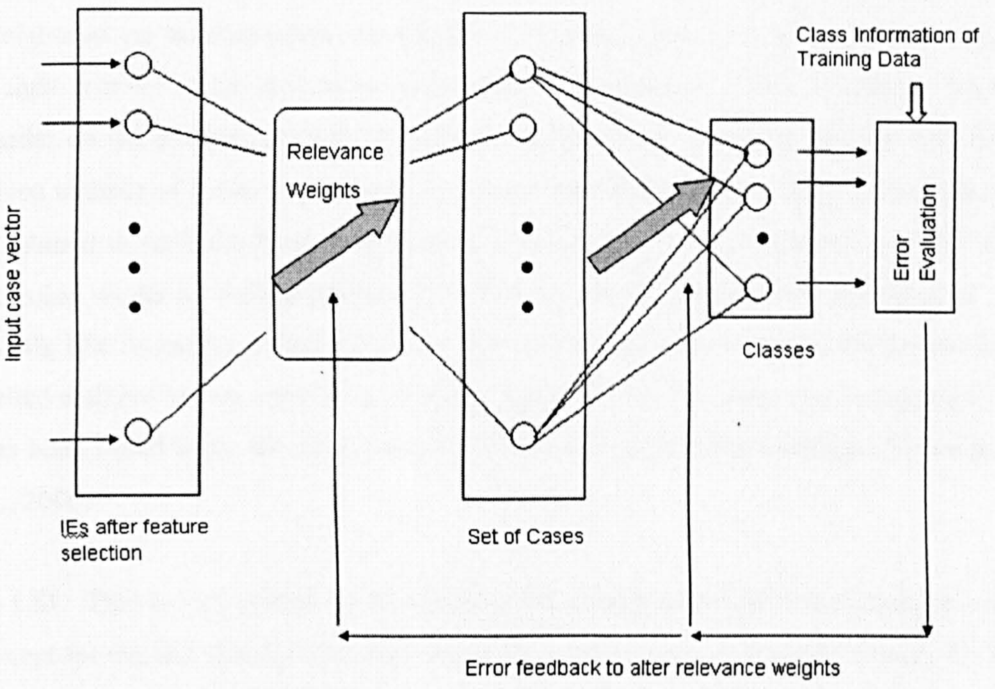

Input case vector
IEs after feature selection
Relevance Weights
Set of Cases
Classes
Class Information of Training Data
Error Evaluation
Error feedback to alter relevance weights

Figure 3.1 A Schematic of the Extended Case Retrieval Network (ECRN)

3. SVM SVMs are founded on the basic idea of inducing hyperplane separators that was briefly discussed in the last chapter. SVMs have been reported to outperform most other off-the-shelf classifiers in several experimental studies over diverse text classification applications (Drucker et al., 1999, Dumais et al., 1998, Joachims 1998). Joachims(1998) argues that SVMs have two advantages in the context of text classification. Firstly, feature selection is not needed, as SVMs are fairly robust to overfitting and can scale upto very high dimensionalities. Secondly, they need almost no parameter tuning, as there is a theoretically motivated, "default" choice of parameter settings, which has been shown to provide the best effectiveness.

4. **LogitBoost** Boosting is based on the idea of using an ensemble (committee) of diverse "weak" learners that complement each other. In the first step, a prediction model is induced from training data and added to the committee. In the next step, the weights of the training cases are changed, so that the hard-to-classify cases get higher weight relative to the rest of the cases. The next member of the committee focuses harder on the difficult parts of the instance space. These two steps are repeated for a given number of iterations, leading to several diverse classifiers, whose predictions are combined to yield the final classification. The diversity of the classifiers explains why boosting works so well in practice (Frank et al., 2002). LogitBoost (Friedman et al., 2000), like its earlier sibling Adaboost, is based on the statistical estimation procedure called additive logistic regression. We have chosen it for our comparative study since it has been found to be the most accurate of multi-class boosting methods (Friedman et al., 2000).

5. **LSI.** This is very similar to the baseline CRN realizing a kNN based classification, except for the fact that the relevance knowledge for the CRN is acquired using LSI. We have briefly introduced LSI in the previous chapter and seen how it uses ideas from factor analysis to model textual cases in terms of their constituent concepts, where a concept can be viewed as a linear combination of features. All of the next chapter is devoted to an in-depth treatment of LSI with special emphasis in Section 4.3 on how it can be used to acquire similarity and relevance knowledge for CRNs.

6. **PSI**. PSI (Wiratunga et. al, 2005a) has goals similar to LSI in that it attempts to acquire an indexing vocabulary to describe textual cases, while minimizing noise due to word choice variability. However, unlike LSI, PSI extracts new features as logical combinations of existing keywords. The underlying thesis is that such logical

combinations correspond more closely to natural concepts and are more transparent than linear combinations mined by LSI. PSI uses association rule mining to arrive at logical combinations of features that are highly discriminatory. As a further step, boosting is used to reduce redundancies in the mined feature set, by retaining only those feature combinations that have minimal overlap with the rest. Experiments on classification domains show that PSI-derived case representations have superior retrieval performance compared to the original keyword-based representations.

3.3 Visualizing Textual Casebases

Let us consider a set of textual cases, each case consisting of a set of features. For simplicity, we treat words in the text as features; the ideas presented can easily be extended to deal with more complex features. The domain is unsupervised, so no class knowledge is available. Also, we will restrict our attention to the problem side of cases, for the moment. To illustrate our ideas, we model the documents in the toy Deerwester collection (Deerwester, et al. 1990) as cases. This is shown in Figure 3.2(a). An alternate representation is in the form of case-feature matrix shown in Figure 3.2(b); elements are 1 when a feature is present in a case, 0 otherwise. It is straightforward to map this matrix onto an equivalent image, shown in Figure 3.3(a), where the values 0 and 1 are mapped to distinct colours, a lighter shade denoting 1. We refer to this as the "casebase image" metaphor. All images in this chapter were obtained using Matlab. However the image as it stands, is not very useful. Firstly, it conveys very little information about underlying patterns in terms of word or document clusters. Secondly, the image is highly sensitive to how the words and documents are arranged in the matrix; this is clearly undesirable. Thirdly, and we shall explore this in more detail later, the image tells us very little about the complexity of the underlying casebase.

To address these limitations, we propose an algorithm that does a two-fold transformation on the case-feature matrix to yield a matrix where similar cases (and similar features) are stacked close to each other. The output is a matrix, which when visualized as an image, captures the underlying regularities in the casebase. Figure 3.4 shows a sketch of the algorithm. The broad idea is as follows. The first case row in the original matrix is retained as it is. Next, we compute the cosine similarity of all other cases to the first case, and the case most similar to the first case is stacked next to it, by swapping positions with the existing second row. If more than one case is found to be equally similar, one of them is chosen randomly. In the next step, all cases excepting the two stacked ones are assessed with respect to their similarity to the second case. The case that maximizes a weighted combination of similarities to the first and second case (with higher weight assigned to the second case) is chosen as the third case, and stacked next to the second row. The process is repeated till all rows are stacked. In Step 2 of the algorithm, the same process is repeated, this time over the columns of the matrix generated by Step 1.

c1: ***Human*** **machine** ***interface*** **for Lab ABC** ***computer*** **applications**
c2: **A** ***survey*** **of** ***user*** **opinion of** ***computer system response time***
c3: **The** ***EPS user interface*** **management system**
c4: ***System*** **and** ***human system*** **engineering testing of** ***EPS***
c5: **Relation of** ***user*****-perceived** ***response time*** **to error measurement**

m1: **The generation of random, binary, unordered** ***trees***
m2: **The intersection** ***graphs*** **of paths in** ***trees***
m3: ***Graph minors*** **IV : Widths of** ***trees*** **and well-quasi-ordering**
m4: ***Graph minors*:** **A** ***survey***

(a)

	human	survey	interface	trees	computer	graph	user	minors	system	time	EPS	time
c1 :	1	0	1	0	1	0	0	0	0	0	0	0
c2:	0	1	0	0	1	0	1	0	1	1	0	1
c3:	0	0	1	0	0	0	1	0	1	0	1	0
c4:	1	0	0	0	0	0	0	0	1	0	1	0
c5:	0	0	0	0	0	0	1	0	0	1	0	1
m1:	0	0	0	1	0	0	0	0	0	0	0	0
m2:	0	0	0	1	0	1	0	0	0	0	0	0
m3:	0	0	0	1	0	1	0	1	0	0	0	0
m4:	0	1	0	0	0	1	0	1	0	0	0	0

(b)

Figure 3.2 Documents in the Deerwester Collection

The weighted similarity evaluation is critical to the working of this algorithm and merits a closer look. The general rule for selecting the (*k*+1) row (case) is to choose the one that maximizes

$$\sum_{i=1}^{k} w_i \, sim(c_i, c) \text{ such that for all } 1 \le i < k, \; w_{i+1} > w_i \qquad (3.1)$$

where k is the number of already stacked rows, c_i is the ith stacked case, c is a case whose eligibility for $(k+1)$th position is being evaluated, $sim(c_i, c)$ is the cosine similarity between cases c_i and c, and w_i is the weight attached to the similarity of c with the ith stacked case. In our implementations, we used

$$w_i = 1/(k - i + 1) \tag{3.2}$$

The basic intuition behind this weighting scheme is that we want to ensure a gradual change in the way cases are grouped. This has implications for facilitating a meaningful display of clusters, and also for the complexity evaluation discussed in Section 3. If only $sim(c_k, c)$ were considered for the stacking process (which is equivalent to assigning 0 to all w_i, i = 1 to k-1) we may have abrupt changes resulting in an image that fails to reveal natural clusters. We note that for efficiency reasons, our implementation uses an approximation of (2), where we take into account only the previous 10 stacked cases and no more, since the weights associated with very distant cases are negligible and have no significant effect on the ordering.

Figure 3.3(a) shows the image corresponding to an arbitrary arrangement of the documents in the Deerwester matrix. Figure 3.3(b) shows the image after the rows are stacked. Figure 3.3(c) is the final image after column stacking. It is interesting to see that the two broad topics within the collection, namely *Human Computer Interaction (HCI)* and *graphs* are clearly visible in Figure 3.3(c) as two "chunks" of contiguous light shades. Also, there is a gradual transition in shades from *HCI* to *graphs*. This is useful in identifying "bridge words" that can serve to connect two topics; an example is word 9 ("survey") in Figure 3.3(c) which is common to *HCI* and *graphs*. We can also visually identify cases that are in the topic boundaries and deal strongly with more than one topic. This is useful for aiding casebase maintenance tasks such as identification of noisy cases and redundant features (Massie 2006). We have designed a simple interface that allows users to "navigate" the image, and visualize the "topic chunks", and words that describe those chunks.

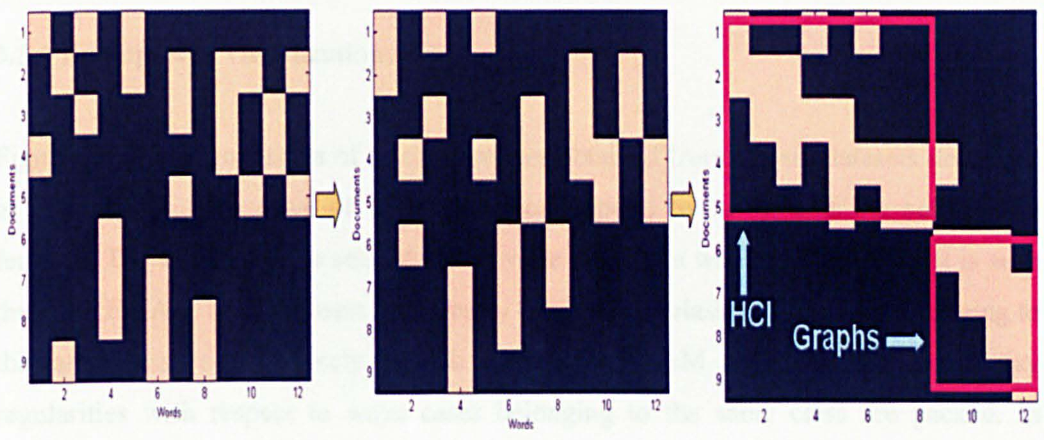


Figure 3.3 Images from Deerwester collection (a) arbitrarily stacked (b) after row stacking (c) after column stacking

```
Step 1 (Stack Rows)
Input : Case-Feature Matrix M
Output : Case-Feature Matrix $M_R$ which is M stacked by rows
Method:
Instantiate first row of $M_R$ to first row of M
for k = 1 to (noOfRows-1) /*the index of the last case (row) stacked*/
    for j = (k+1) to noOfRows /* check through all candidate cases*/
        $wsim_j = 0$; /* $wsim_i$ weighted similarity of ith case */
        for i = 1 to k /* already stacked rows*/
            $wsim_j = wsim_j + wsim_j*(1/(k-i+1))*sim(c_i,c_j)$ ;
        end
    end
    choose j that maximizes $wsim_j$ and interchange rows (k+1) and j
end
Step 2 (Stack Columns)
Input : Case-Feature Matrix $M_R$ generated by step 1
Output : Case-Feature Matrix $M_C$ which is $M_R$ stacked by columns
Method: same as in Step 1 except that columns are interchanged (based on feature similarity computed
as cosine similarity between columns) instead of rows.
```

Figure 3.4 The Stacking Algorithm

3.3.1 Examples of visualizations

Figure 3.5 shows snapshots of stacked images obtained from the six datasets described in Section 3.1. The rows of each image correspond to cases, and the columns to features. The case rows are shaded to show the classes to which they belong. It is seen that USREMAIL has very neat separability between the classes with cases belonging to the same class packed closely to each other. LINGSPAM and RELPOL also display regularities with respect to ways cases belonging to the same class are packed. In contrast, HARDWARE is clearly a complex domain, with very little separability between classes, and very few pronounced topic chunks. As noted in Section 3.1, this is because of the presence of large number of features which are shared by classes Apple and Mac. To increase effectiveness of classification in HARDWARE, one approach is to combine features to extract new features which are more discriminative of the two classes.

3.4 Complexity evaluation of textual casebases

Complexity of a casebase is independent of classifiers and derived directly from the casebase characteristics that are critical to estimating the difficulty of performing effective retrieval or classification on that dataset. In this section, we explore how the image metaphor can be exploited to obtain a measure of the casebase complexity. For completeness, we will digress into the more general problem of complexity evaluation over textual casebases, and then show how we can arrive at a complexity measure that suits our needs.

There are two reasons why complexity evaluation is useful. Firstly, we can predict difficulty of domains (datasets) for a given choice of representation (feature selection/extraction and similarity measures). Secondly, we can compare across different choices of representation over a fixed domain and choose the representation

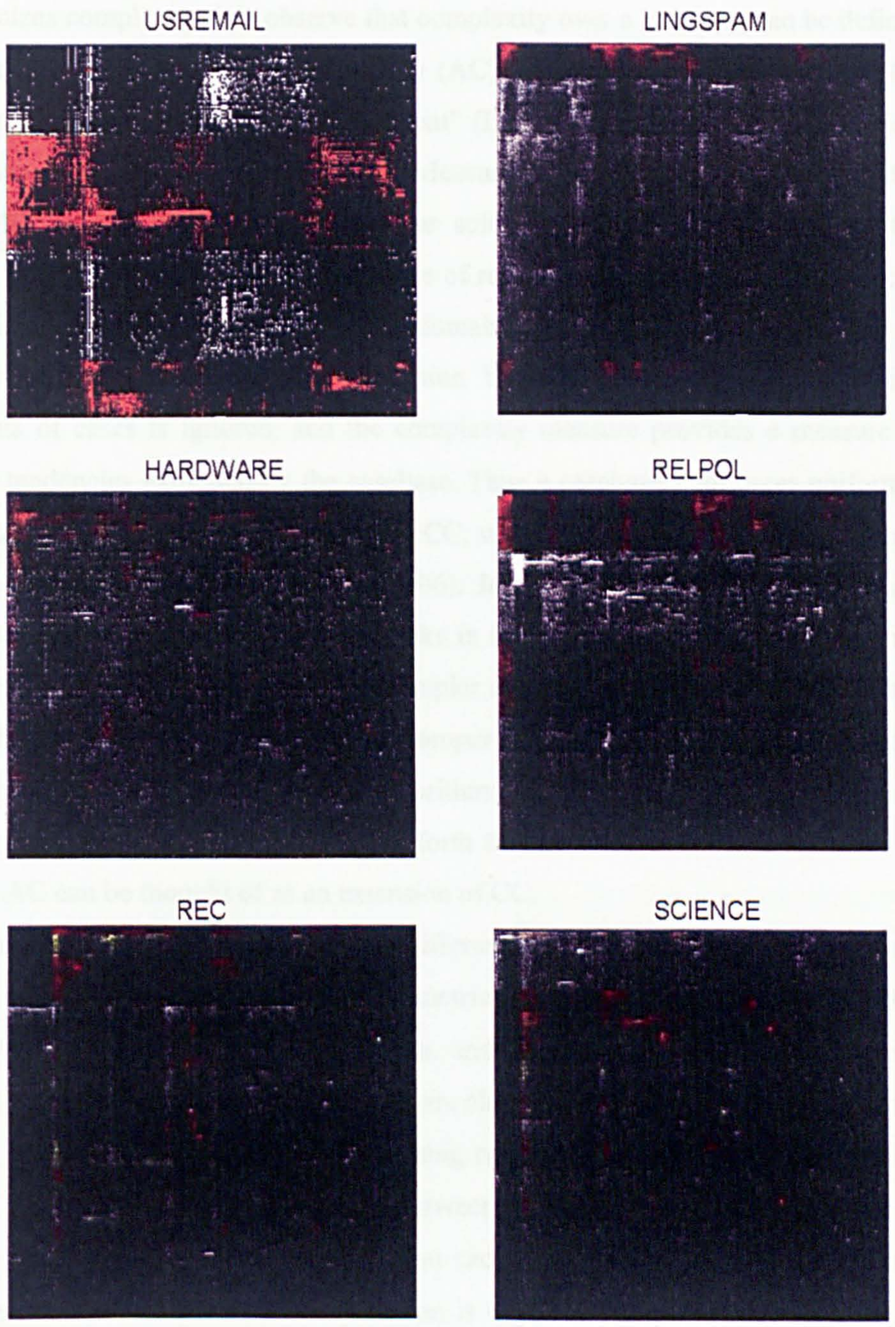


Figure 3.5 Stacked Images obtained from the six datasets

that minimizes complexity. We observe that complexity over a casebase can be defined in two ways, namely Alignment Complexity (AC) and Collection Complexity (CC). The former measures the degree of "alignment" (Lamontagne 2006) between problem and solution components of textual cases. Measuring this helps us in answering the question "Do similar problems have similar solutions?" and thereby assessing the suitability of CBR (or alternatively the choice of representation) to that task. A special case of this problem is seen in classification domains, where the solution is replaced by class label. In measuring CC, the distinction between the problem and solution components of cases is ignored, and the complexity measure provides a measure of clustering tendencies exhibited by the casebase. Thus a casebase with cases uniformly distributed over the feature space has a high CC; whereas, one with more well-defined clusters has a lower CC (Vinay et al., 2006). Intuitively, since the stacked image captures regularities arising from topic chunks in the casebase, we would expect that, all else being equal, stacked images from simpler domains will be more compressible, and thus have higher compression ratios, compared to ones from complex domains. This is because image compression algorithms typically exploit regularities to minimize redundancy in storage. We carry forth this intuition into our discussions of AC, since AC can be thought of as an extension of CC.

Alignment can be interpreted in two different ways. The first interpretation is a local one; an example is the case cohesion metric formulated by (Lamontagne 2006). Here we look at a case, say *C*, in isolation, and determine two sets: set S_1, which comprises cases whose problem components are closest to the problem component of *C* (based on a threshold), and a set S_2, comprising cases whose solution components are closest to the solution of *C*. The overlap between S_1 and S_2 is used as a measure of alignment of *C*. This is a local metric, in that each case is evaluated on its own, and assigned a measure. The second interpretation is a global one based on how well the clusters derived from problem components of cases correspond to clusters derived from solution components. In other words, a global measure is different from a local one in that it is not evaluated "bottom-up" by aggregating complexities obtained by looking at each case in isolation. Rather the clustering patterns at a broad level are used as the basis for evaluating complexity. Also, a global measure cannot be extended comfortably to yield case-specific complexity. In this paper we adopt this second global interpretation of alignment.

Compression approaches used to measure CC can be extended to measuring AC. For measuring alignment, we construct two case-feature matrices: one based on problem components of cases, the other based on solution components. These two matrices are stacked as described in Section 3.3, to yield two images I_P and I_S respectively. I_P and I_S are now independently compressed to obtain compression ratios CR_P and CR_S respectively. The higher the compression ratio, the more pronounced the clustering patterns. We note that generating I_P and I_S involve reordering the cases, and we can read out the new order in which cases are arranged based on problem and solution side clustering. For measuring alignment, we compare the ordering of cases in I_P and I_S. One way of doing this is to create a fresh solution side image I_{SP} by stacking solution components of cases using the problem side ordering of cases as read out from I_P. We would intuitively expect I_{SP} to be less compressible than I_S, unless the casebase is perfectly aligned. Compressing I_{SP} yields a new compression ratio CR_{SP}. The Global Alignment MEasure (GAME) is given by CR_{SP}/ CR_S. A higher value of GAME indicates a better alignment. An alternate measure can be obtained by considering I_{PS}, the problem side image with solution ordering imposed on it, instead of I_{SP}. However, our choice of I_{SP} over I_{PS} was governed by the observation that in CBR, while we are keen on ensuring that similar problems have similar solutions, it is not of primal importance that similar solutions necessarily originate from similar problems. Using I_{SP} takes care of this asymmetry.

GAME can be extended to classification domains where the class label is treated as a solution. In this case, our interest is in determining whether near-neighbours in the problem side ordering (as obtained from I_P) belong to the same class. We obtain a string of class labels corresponding to the problems as they appear in the problem side ordering. This allows us to do away with the image compression and resort to a simpler string based compression instead. As an illustration, let us consider a two class problem of 10 cases in the email domain, where cases C_1 through C_5 belong to class S (for SPAM) and C_6 through C_{10} belong to L (for LEGITIMATE). Let us assume that the problem side ordering of the cases after stacking is $C_1C_2C_6C_4C_5C_7C_3C_9C_{10}C_8$. Replacing each case identifier with its class label, we obtain the class string SSLSSLSLLL. The most easily classifiable casebase would have a string SSSSSLLLLL, and the most complex would have SLSLSLSLSL. A compression algorithm that exploits contiguous blocks (but not compound repeating patterns like SL) would thus be ideal; Run Length Encoding (Rosenfeld and Kak, 1982) is one such scheme. Using this, the complexity is a direct function of the number of the flips (changes from one class label to another, N to S or S to N in the above example). We define GAME complexity measure for classification as

$$\text{GAME}_{\text{class}} = \log\left(\frac{flips_{\max} - flips_{\min}}{flips - flips_{\min}}\right) = \log\left(\frac{(n-1) - (k-1)}{flips - (k-1)}\right)$$

where k is the number of classes, n is the number of cases ($n > k$), *flips* is the number of transitions from one class to another in the class string, $flips_{\min}$ is the value of *flips* for the simplest possible casebase having n cases and k classes, and $flips_{\max}$ is the value of *flips* for the most complex casebase. We note the most complex casebase presupposes a uniform class distribution; we then have $flips_{\max} = (n\text{-}1)$. A higher values of $\text{GAME}_{\text{class}}$ corresponds to a simpler domain; the most complex domain has $\text{GAME}_{\text{class}} = 0$. Thus we expect positive correlation of $\text{GAME}_{\text{class}}$ to accuracy results derived from classifiers. The logarithm has a dampening effect on the large values that could result when $n >> k, flips$. As a further detail, a small constant (say 0.01) should be added to the denominator to avoid division by zero when $flips = flips_{\min}$.

An important issue that merits more attention is the choice of starting case in the stacking process, and its influence on the visualization and complexity measure. A theoretically sound way of choosing the starting case would be to perform stacking several times, using a distinct case each time as a starting case. When we have exhausted all possibilities, we choose the arrangement that yields the highest compression ratio. Further research is needed to find efficient ways of pruning the search space to make this process less computationally expensive. Our experiments have shown that visualizations are not widely affected by the choice of starting cases, except for the shuffling in the order in which clusters are displayed.

3.5 GAME$_{class}$ for the six datasets

Figure 3.6 shows the GAME$_{class}$ values obtained over the 15 trials in each of the six datasets. Of the binary problems, LINGSPAM and USREMAIL have high GAME$_{class}$ values indicating that they are simpler compared to HARDWARE which has a low GAME$_{class}$ value. Table 3.1 suggests that GAME$_{class}$ predictions are supported by accuracy figures recorded by different classifiers. The current formulation of GAME$_{class}$ allows for more meaningful comparisons between problems when they have the same number of classes. So we compared the binary and four-class problems separately. The correlation coefficient of the GAME$_{class}$ score against classification accuracies over the four binary problems are shown in Table 3.2. We note a strong positive linear correlation of GAME$_{class}$ to all four classifiers. It is pointless to do correlation over the four-class datasets since we have just two of them; however we observe that GAME$_{class}$ declares SCIENCE to be more complex than REC, and this is confirmed by all classifiers. SVM being inherently a binary classifier was not applied on the multi-class datasets, though we plan to experiment with multi-class SVM in future.

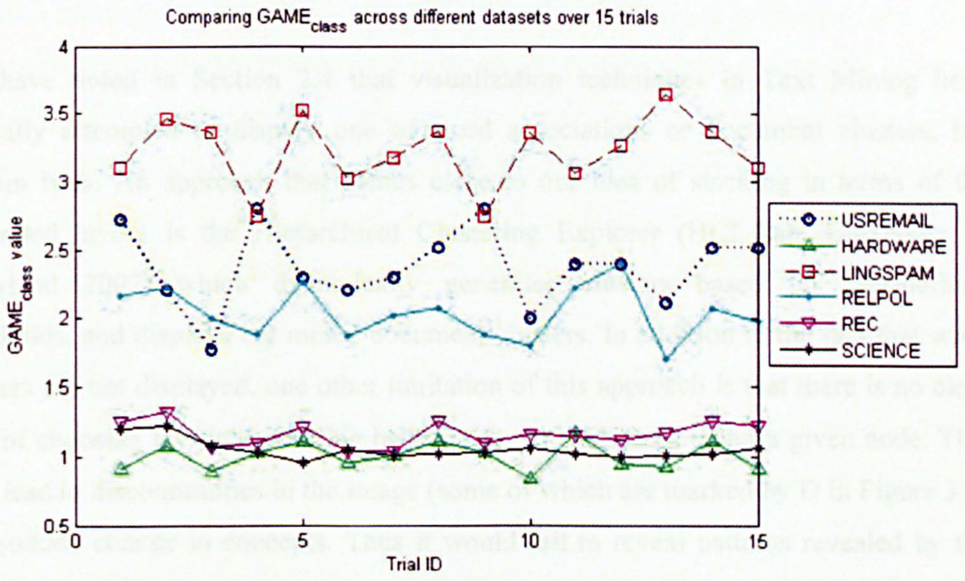


Figure 3.6 GAME$_{class}$ values across different datasets

Table 3.1 GAME$_{class}$ and Accuracies obtained by different classifiers

	REC	SCIENCE	HARDWARE	RELPOL	USREMAIL	LINGSPAM
GAME$_{class}$	1.1629	1.0492	1.0028	2.0358	2.3728	3.2222
kNN (CRN)	62.79	54.89	59.51	70.51	59.23	85.09
LSI	79.32	72.55	66.30	91.17	94.67	97.37
ECRN	69.91	80.18	80.12	93.26	96.50	98.17
SVM	--	--	78.82	91.86	95.83	95.63
LogitBoost	87.15	73.77	77.99	79.67	92.67	95.80
PSI	66.28	76.2	80.1	91.2	94.83	95.8

Table 3.2 Correlation of classifier accuracies with GAME$_{class}$

	kNN (CRN)	LSI	ECRN	SVM	LogitBoost	PSI
ρ	0.7685	0.9176	0.9360	0.9023	0.8820	0.9330

3.6 Discussion of Related Work

We have noted in Section 2.4 that visualization techniques in Text Mining have typically attempted to display one of word associations or document clusters, but seldom both. An approach that comes close to our idea of stacking in terms of the generated layout is the Hierarchical Clustering Explorer (HCI Lab, University of Maryland 2007) which dynamically generates clusters based on user-defined thresholds, and displays the mined document clusters. In addition to the fact that word clusters are not displayed, one other limitation of this approach is that there is no clear way of choosing the right ordering between several sub-trees under a given node. This may lead to discontinuities in the image (some of which are marked by D in Figure 3.7) and sudden change in concepts. Thus it would fail to reveal patterns revealed by the weighted stacking approach. An approach that comes close to showing both words and documents in the same space is WEBSOM (Feldman et al., 2007). WEBSOM fails to preserve the structure of cases as a set of feature values, and is unwieldy for casebase maintenance. Furthermore, our approach has the relative advantage of being free from convergence problems faced by WEBSOM.

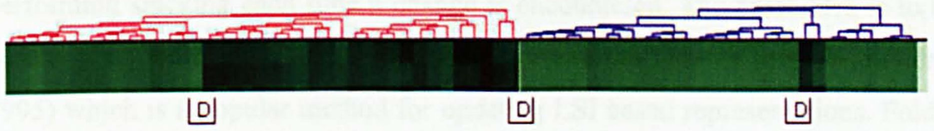


Figure 3.7 A snapshot of hierarchical visualization (courtesy HCI Maryland website)

It would be interesting to explore parallels between "topic chunks" revealed by the stacked image, and concepts as mined by Formal Concept Analysis (FCA) (Diaz-Agudo et al. 2001). While FCA has been applied to TCBR tasks, the inherent sparseness of textual data leads to generation of a large number of concepts that are brittle and unintuitive. Relaxing the strict closure requirements of FCA could possibly lead to "approximate concepts". Our intuition is that a topic chunk, when interpreted as a blurred rectangular version of the actual light shades in close proximity, may be a close analog to such an approximate concept. It is worth noting that this blurring operation can be viewed as smoothing of cases based on neighbourhood of each cell, thus achieving feature generalization. Blurring makes sense only on the stacked image since we are assured that neighbouring cells are likely to correspond to similar cases and features; it is meaningless on the original image where the arrangement is arbitrary. In the next chapter, we show that lower dimensional representations generated by LSI can be regarded as blurred versions of the original casebase. This parallel opens up avenues for exploring alternatives to LSI that tailor the blurring process to cater to specific TCBR goals. Of particular interest in this context is the idea of image transforms proposed by Hoenkamp (2003).

As a final point, we note that casebases are seldom static, so the importance of efficient update strategies that can handle additions, deletions or updates of cases (or features) cannot be over-emphasized. Though we have not experimented with dynamical collections, our current prescription is a lazy strategy that makes quick incremental but approximate updates whenever a change happens, and relegates the job of making accurate changes at a later "bulk update" stage. This saves the overhead of performing stacking each time a change is encountered. The basic idea is to trade off accuracy for efficiency, and is similar in sprit to the idea of folding-in (Berry et al. 1995) which is a popular method for updating LSI based representations. Folding-in is briefly described in Section 4.2.2 in the following chapter.

3.7 Chapter Summary

In this chapter we have emphasized the importance of characterizing textual casebases in explaining the performance of retrieval or classification techniques that operate over them. Towards this direction, we have presented novel approaches for visualizing and evaluating complexity of textual casebases. The visualization gives a bird's eye view of the domain and helps the expert or knowledge engineer make qualitative judgments on its characteristics. We have studied the performance of text classifiers founded on well-studied principles over six textual datasets, and analysed the correlation of our complexity measure against accuracy figures reported by these classifiers. The six datasets that were examined in detail in this chapter will be used in later chapters to evaluate novel approaches to acquire relevance and similarity that aim at improving retrieval effectiveness. This will help us derive better explanations of experimental results in comparison with other classifiers, and particularly in terms of the inherent dataset characteristics.

Chapter 4

Latent Semantic Indexing for Knowledge Acquisition in CRNs

Classification is hard, especially for unlabelled cases.[1]

In Chapter 2, we have seen several limitations associated with bag of words representation for textual cases. To summarize, BOW fails to recover from polysemy and synonymy, and hence cannot resolve disparities due to variability in word choice. In a study (Furnas et al., 1987), it has been reported that different people use the same keywords for expressing the same concepts only 20 % of the time. As noted before, there are two distinct ways of addressing this problem, and most practical solutions use a combination of both approaches. The first approach is to use carefully handcrafted knowledge sources like domain specific ontologies and rule-bases, or linguistic resources like thesauri and Wordnet (Miller 1995). The second approach is to use statistical learners to infer "latent" word associations from a document corpus. Given our emphasis on reducing manual knowledge engineering overheads, we are more interested in the second approach. We have briefly introduced a statistical approach called LSI in Chapter 2 and argued that it has certain advantages over other approaches in relation to our TCBR goals. To recapitulate, these advantages include: ability to position documents and words in the same space and hence generate both similarity and relevance knowledge, easy integration with instance based learners because of its

[1] Inspired by Neil Bohr's quote: "Prediction is difficult, especially about the future."

grounding on the vector space formalism, rich underlying representation of concepts in terms of words or documents allowing for good visualization support, availability of easy update strategies, efficiency of retrieval, easy integration of background knowledge in the form of additional documents to boost retrieval effectiveness, and reliance on very few parameters to be tuned. One critical limitation is that LSI fails to exploit class knowledge in supervised classification tasks. We address this limitation in depth in Chapter 5. In this chapter, we elaborate on the mechanics of LSI with the goal of laying a foundation for the following chapters. We also illustrate how LSI can be used to mine knowledge for TCBR tasks, and how the acquired knowledge can be integrated into the CRN.

The rest of the chapter is organized as follows. In Section 4.1 we attempt to provide an intuitive insight into the mathematics of LSI. In Section 4.2 we relate this mathematical understanding to the context in which LSI is actually applied. We also examine the rationale behind using a dual mode factor analysis, from different standpoints. Section 4.3 shows how LSI can be used to mine similarity and relevance knowledge for CRNs.

4.1 Two Mode Factor Analysis

LSI was proposed as a technique for concept extraction by Deerwester. The starting point for LSI is a term document matrix (alternately case feature matrix). The objective is to determine a set of underlying "factors" or concepts, that best explain the relationship between the terms and documents. This is not very different from the goal of most factor analytic research from the sixties to the nineties. What distinguishes LSI from most earlier approaches is its "two mode factor analysis" which allows it to express both words and documents in terms of the same underlying concepts. For the sake of completeness, we provide a brief introduction to linear algebraic techniques for single-mode factor analysis in Section 4.1.1. In Section 4.1.2 we introduce the Singular Value Decomposition, which is at the heart of LSI.

4.1.1 An Introduction to the Mathematical Foundation of LSI

To start with, we have a term document matrix; each element in that matrix is a weight showing the relevance of the term to the corresponding document. The first significant step in Linear Algebra is to view a matrix such as this as an operator. This means that the matrix can act upon a vector (when it is multiplied with that vector), and relocate it to a different position. For example, the square matrix[2]

$$M = \begin{pmatrix} 2 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 2 \end{pmatrix}$$

can act on the vector

$$\vec{A} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

and move it to a new location given by $M\ \vec{A}$:

$$M\ \vec{A} = \begin{pmatrix} 3 \\ 0 \\ 5 \end{pmatrix}$$

In the underlying geometry of the space, the action of a matrix M can be viewed as a combination of translation and rotation of $\vec{A}$ in the general case. We are interested in charactering a matrix M formally in terms of its properties that govern its action on vectors; the concept of eigenvectors does precisely that.

We consider all vectors $\vec{x}$ that, when acted on by M, stretch themselves to a different location $\lambda\ \vec{x}$, where λ is a scalar, but do not undergo any rotation. Thus

[2] Note that, unlike the example presented, all entries in a term document matrix are usually non-negative.

$$M\vec{x} = \lambda\vec{x} \tag{4.1}$$

The vectors satisfying (4.1) are called eigenvectors, and each of these eigenvectors is associated with a corresponding value of λ referred to as an eigenvalue. We rewrite (4.1) as $(M - \lambda I)\,\vec{x} = 0$, where I is an identity matrix of dimensions matching M; this is called the characteristic equation. Solving it in our example, we have the following three eigenvectors

$$v_1 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, v_2 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \text{ and } v_3 = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$$

associated with the eigenvalues $\lambda_1 = 1$, $\lambda_2 = 1$ and $\lambda_3 = 4$ respectively.

We now study the effect of M on any arbitrary vector $\vec{x}$

$$\vec{x} = \begin{pmatrix} 4 \\ -1 \\ 4 \end{pmatrix}$$

We can express $\vec{x}$ as a linear combination of v_1, v_2 and v_3. The revised position $M\vec{x}$ is now given by

$$\begin{aligned} M\vec{x} &= M(1v_1 + 2v_2 + 3v_3) \\ &= Mv_1 + 2Mv_2 + 3Mv_3 \\ &= \lambda_1 v_1 + 2\lambda_2 v_2 + 3\lambda_3 v_3 \end{aligned}$$

The interesting aspect of this rewrite is that we can see that the total effect of M on $\vec{x}$ is expressed as a weighted combination of effects due to each eigenvector. Eigenvectors having very small eigenvalues associated with them have a small effect on the operation of M on $\vec{x}$. In the example above, the eigenvector associated with the eigenvalue $\lambda_3 = 4$ will have a more pronounced effect in characterizing M as an operator compared to the two other eigenvectors each associated with eigenvalue 1. This intuition is critical to our treatment of SVD below.

Moving on to a few more definitions, a family of a finite number of vectors is said to be linearly independent if none of them can be expressed as a linear combination of the remaining ones. The rank of a matrix M (not necessarily square) is the number of linearly independent columns (or rows) in it. It can be shown that the rank of a square matrix equals the number of its non-zero eigenvalues, counted with multiplicity.

We now look at an important result in factor analysis. For a given square real valued $m \times m$ matrix M with linearly independent eigenvectors, we can obtain a factorization

$$M = U \Lambda U^{-1}$$

such that the columns of U are the eigenvectors of M, and Λ is a diagonal matrix whose diagonal elements are eigenvalues of M arranged in decreasing order. This result is due to the Matrix Diagonalization Theorem. This result applies to square matrices, but not to rectangular ones like the term-document matrix.

4.1.2 The Singular Value Decomposition

Previous attempts at factor analysis applied the idea to term-term matrices or document-document matrices, which are square. This is referred to as single-mode factor analysis. In contrast, a two mode factor analysis starts off with a rectangular term document matrix M of dimensions $m \times n$ (corresponding to m terms and n documents), and rank r. The key apparatus is the singular value decomposition (SVD) of M, which is given by:

$$M = U \Sigma V^T$$

where

U is an $m \times m$ matrix whose columns are orthogonal eigenvectors of $M M^T$.

V is an $n \times n$ matrix whose columns are orthogonal eigenvectors of $M^T M$.

The eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_r$ of MM^T are the same as eigenvalues of M^TM. The square root of these r eigenvalues, called singular values, are arranged in descending order along the diagonal of the matrix Σ, all other elements of which are set to 0.

We have seen before that small eigenvalues contribute less to the effect of the action of a matrix M on vectors. Extending this intuition to SVD, it is interesting to see the effect of considering only the top k singular values, and discarding the rest (flipping them to 0). Thus the matrix Σ is shrunk to a $k \times k$ diagonal matrix Σ_k. We also delete the columns corresponding to low (and zero) singular values in U and V to obtain $\hat{U}$ and $\hat{V}$ respectively. $\hat{U}$, Σ_k and $\hat{V}$ can now be combined to yield

$$\hat{M} = \hat{U}\Sigma_k \hat{V}^T \tag{4.1}$$

$\hat{M}$ is a k-rank approximation to M. This result is pivotal to our discussion of LSI that follows in the next section. In the rest of the thesis, we will refer to M as a case-feature matrix (with cases as rows and features as columns), excepting situations where we refer to equation 4.1 which is formulated with cases as columns and features as rows.

4.2 Latent Semantic Indexing

SVD as formulated in Equation 4.1 is at the heart of LSI. In this section, we examine the use of SVD for arriving at better textual representations. We show how LSI can be made useful in practice, and take specific note of issues that provide a context to motivate research reported in the following chapters.

4.2.1 SVD for LSI

The following are a few distinct directions from which SVD is interesting from the point of view of text retrieval.

Firstly, we note that SVD achieves dimensionality reduction. Let M be a case-feature matrix, with each row representing a case. Geometrically, the rows of $\hat{U}$ and $\hat{V}$ are co-ordinates of points corresponding to cases and features mapped onto a k-dimensional space. Typically, the axes are scaled using the k singular values to assign more importance to dimensions that are associated with high singular values. These reduced dimensional representations can then be compared against each other using the dot product or the cosine measure.

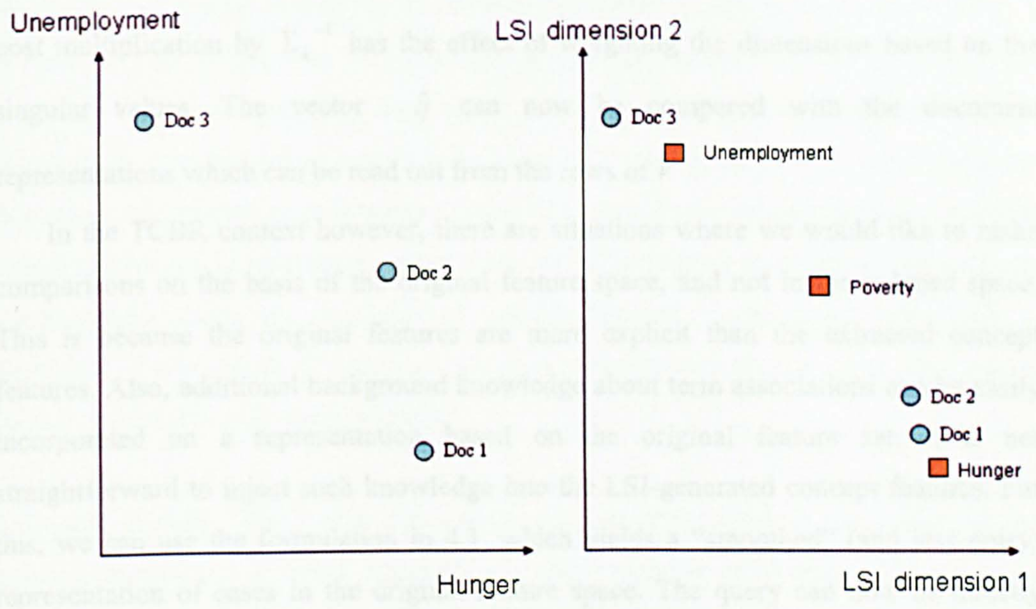
Secondly, it can be shown that $\hat{M}$ is the best k-rank approximation to M in the least-squares sense. The quality of an approximation M_A is measured by the Frobenius Norm of the "discrepancy" matrix $X = M - M_A$, which is given by:

$$\| X \|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij}^2}.$$

The lower the value of $\| X \|_F$, the better the matrix M_A is, as an approximations to M. Viewing the low rank approximation problem as one of constraint optimization, it can be shown that, of all approximate matrices that satisfy the constraint that their rank is at most k, $\hat{M}$ is the one that registers a minimum value for $\| X \|_F$. This conforms to our earlier intuition that removing very small singular values does not significantly affect M. The important thesis behind LSI is that the small singular values correspond to noise due to word choice variability (polysemy and synonymy). $\hat{M}$ is a less sparse representation compared to M that broadly retains the patterns of term association to documents, but at the same time "smoothes" it out to eliminate noise.

Thirdly, we note that the correspondence between low singular values and noise due to word choice variation is not accidental. Considering a square matrix M with two identical columns, we can eliminate one of these and still retain the same rank. This is a trivial case of feature selection. If instead, M had nearly identical columns, it would mean that the two corresponding features would have co-occurred similarly with documents. This would be true for closely related features like "Middle East" and "oil" which might appear in very similar contexts in a large document corpus. In such a case, it is intuitive that we can still go ahead with replacing the two columns corresponding to the two features by a new feature (column) that averages or smoothes out the two original features. This is exactly what SVD achieves when it constructs a low rank approximation. In this context, we make a critical distinction between the "true rank" and "effective rank" of a matrix. While the true rank takes into account all non-zero singular values, effective rank discards the very small ones. Thus replacing two closely related features by a single new feature changes the true rank but maintains the effective rank of the matrix. *The ability of SVD to identify "latent" co-occurrence patterns is the main reason for its improved effectiveness in retrieval tasks compared to the plain vector space model based on bag of words.* Also, the new features which are referred to as "concept" features are expected to be more robust indicators of meaning in comparison to the original feature set. This can be viewed as a step of feature extraction. It is important to note that extracted features can be expressed as a linear weighted combination of original features. There is another notable consequence of feature extraction: although LSI deals reasonably well with synonymy, (Deerwester 1990) observe that the solution it offers to polysemy is at best partial. This is also confirmed by the results of their experiments. The problem lies in the fact that LSI forces a term to have a single representation in the concept space; thus a word with multiple meanings is represented as the weighted average of the different meanings. It is possible that none of the "real" meanings is close to the average, leading to a serious distortion. We will revisit this idea using a concrete example in Section 4.3.

Fourthly, both terms and documents are treated in a uniform way by LSI. The concept features act as new dimensions, in terms of which both terms and documents are represented. In Figure 4.1(a), which is an adapted version of Figure 18.3 from the online version of (Manning, C. et al., 2008 expected), we show an example of vectors spaces before and after LSI. Figure 4.1(b) shows how representations of words and documents obtained by LSI can be positioned in the new concept space. This allows us to visualize term and document clusters in the same space, and obtain interpretable descriptors of these clusters based on neighbouring words.



(a) Original Feature Space (b) Terms and documents in the LSI concept space

Figure 4.1 LSI in an example domain

4.2.2 LSI: Beyond SVD

In this section, we focus on additional issues that need to be addressed to make LSI work in practice.

Query Transformation. For LSI to be practically useful in a retrieval task, it is obviously not enough to represent documents in a low dimensional space; we need to map the query to that space as well so that it can be meaningfully compared with the documents. This mapping is given by:

$$\hat{q} = q' U_k \Sigma_k^{-1} \tag{4.2}$$

where q is the original query and $\hat{q}$ its representation in the k-dimensional space. The post multiplication by Σ_k^{-1} has the effect of weighting the dimensions based on the singular values. The vector $\hat{q}$ can now be compared with the document representations which can be read out from the rows of V .

In the TCBR context however, there are situations where we would like to make comparisons on the basis of the original feature space, and not in the reduced space. This is because the original features are more explicit than the extracted concept features. Also, additional background knowledge about term associations can be easily incorporated on a representation based on the original feature set; it is not straightforward to inject such knowledge into the LSI-generated concept features. For this, we can use the formulation in 4.1, which yields a "smoothed" (and less noisy) representation of cases in the original feature space. The query can now be directly compared against these representations. A relevant case that shared no features with the query in the original vector space, may now register a non-zero similarity with the query. This is because the smoothed representation of the case generated by LSI is likely to reduce the sparseness of the original representation and assign positive relevances to words contextually related to those present in the case. This increases the likelihood of the case being retrieved in response to a semantically related query that uses a different choice of words. This will be illustrated with an example in Section 4.3.

The choice of dimensionality One critical factor determining LSI performance is the choice of *k*, the number of singular values that need to be considered. There is no elegant solution to determining the value that works best. Usually the effectiveness of retrieval is evaluated on a subset of documents over which relevance judgments are available, and the value of *k* that works best is used for the entire collection. In supervised classification domains, an appropriate value of k can be obtained using cross validation over the training set. Using a very low value of k may result in filtering out useful information along with noise; using a very high value may lead to retaining noise that ought to be filtered out. In terms of evaluation measures, using a low value of k leads to high recall; thus documents even remotely relevant to the query will get retrieved. However precision may suffer, since not all retrieved documents may be relevant. In contrast, using a very high value of k may result in low recall and high precision. An extreme case is using a value of k same as the original number of terms, which results in LSI representations that are no different from the original vector space. Alternatively, using the measures of completeness and correctness introduced in Section 2.3, low values of k favour complete retrieval over correct ones; high values favour correctness over completeness.

Efficient Update When new documents or terms are added to the collection, or existing documents or terms removed, the LSI generated representations need to be revised to accommodate these changes. Three broad update strategies have been proposed in LSI literature: (a) recomputation (b) folding-in (c) SVD update. Recomputation involves a brute force SVD computation all over again to generate fresh representations, each time a change happens. This is the most straightforward and the most inefficient of the three approaches. Folding in terms and documents is a more efficient approach, where the new documents (or terms) are mapped to the existing representation using a formulation similar to query transformation shown in Equation 4.2. This obviates the need to compute SVD from scratch, and hence is much more efficient compared to recomputation. The downside is that folding-in may not generate accurate representations as would be obtained with a fresh SVD on the revised collection. For one, the orthogonality of $\hat{U}$ and $\hat{V}$ are no longer guaranteed. This issue is addressed by the approach called SVD-update, which obtains a revised lower rank approximation which is comparable in accuracy to the one that would be obtained with recomputation, and also preserves the orthogonality of $\hat{U}$ and $\hat{V}$. SVD update is faster compared to recomputation since it involves only incremental changes to the existing representation. However, it is much slower, though more accurate, compared to folding in. The choice between SVD update and folding-in is thus driven by the tradeoff between accuracy and time performance. Berry et al. (1995) observe that SVD update is preferable when changes are more frequent. While update strategies are peripheral to the theme of this thesis, we note that these approaches can be directly applied to the LSI improvisations suggested in the following chapter. Most realistic applications have dynamic collections, hence efficient updates is of critical concern from a practical standpoint.

Using LSI for Classification LSI can easily be extended to supervised text classification tasks. The revised representations of the labelled training documents in the lower dimensional space are obtained as described before. Alternatively, we can use the reduced rank approximations of the training documents in the original feature space. The test documents are mapped to either of these spaces, and a weighted k-Nearest Neighbour algorithm is used to identify the nearest neighbours. A class label is assigned to the test document based on weighted majority vote. Either cosine similarity or Euclidean distance can be used as to compute similarities. As has been observed in Chapter 2, a disadvantage with this straightforward extension of LSI to classification is that class labels of the training documents play no role in constructing the revised document representation.

Space and Time Efficiency LSI generated representations are more compact compared to the original vector space because of reduction in the feature space size. However, this does not really translate to reduction in storage space requirements, since advantages due to compaction are offset by the fact that LSI destroys the sparseness of the original representation. Furthermore, storage requirements are compounded by the fact that relevance values generated by LSI are real numbers, while the original term document matrix is typically binary valued. The loss of sparseness has adverse effects in terms of retrieval time as well. While the use of an inverted index as common in IR applications (Rijsbergen 1979), leads to significant speedups over the original representation by restricting the number of cases compared, it is relatively ineffective over the LSI generated representation. In Chapter 8, we address the issue of sparseness in depth and propose an efficient CRN-based retrieval formalism to facilitate fast retrieval over non-sparse representations.

4.2.3 Why does LSI work? An Empirical Justification

In Section 4.2.1, we have presented intuitive explanations for improvements in performance observed with LSI. In this section, we will look at some interesting empirical evidence (Kontostathis & Pottenger 2006) that suggests that LSI's effectiveness can be attributed to its ability to model higher order term co-occurrences. If two words co-occur in at least one document in the collection, they are said to share a first order co-occurrence between them. Examples are "filtering" and "indexing" in the casebase shown in Figure 4.2(a). Furthermore, we note that "matrix" and "clustering" co-occur in one document, and words "matrix" and "differential" in another; thus we can infer that "clustering" and "differential" are related to each other, even if they do not co-occur in any document. Such a relation is called a second-order association, and the word sequence "clustering – matrix -- differential" defines a second order path between "clustering" and "differential". We can extend this idea to orders higher than 2. While we revisit the issue of higher order co-occurrences in more detail in Chapter 5, here we take note of the main conclusions reached by Kontostathis and Pottenger (2006) that will be relevant to our discussion in the following three chapters.

There is a strong correspondence between the number of higher order co-occurrence paths between two words and the similarity between them as inferred by LSI. The authors derive an LSI based term-term similarity matrix as described in

Section 4.3.2, for several well known corpuses in IR, like MED, CRAN and LISA. They also independently mine the number of co-occurrence paths (upto the 6th order) corresponding to each pair of terms. The first five orders of association show a bearing on the term similarity values. While the individual contribution of very high order paths is expected to be small, this is compensated by the fact that there are so many of them. In the LISA collection comprising about six thousand documents, the authors found around 50,000 pairs with first order associations between them, around 10 million with 2nd order co-occurrence paths and over 60 million with 3rd order paths.

Terms with high LSI similarities are those that have a moderate number of co-occurrences with other terms, and not those that share a huge number of high order relationships with other terms. This suggests that the latter is treated as noise by LSI. In contrast, second order pairs with many connectivity paths between them are associated with high LSI similarity values; those with a moderate number are associated with negative values. The authors infer that this points to the fact that second order associations are critical to the "latent semantics" emphasized by LSI.

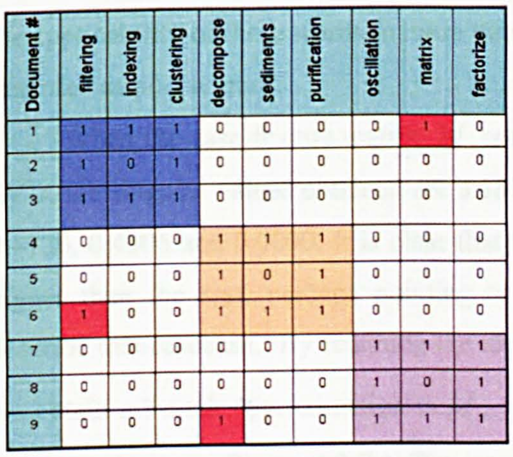
The authors mathematically prove that a connectivity path (at least one of several higher orders) exists between any pair of terms with a non-zero LSI similarity.

4.3 Using LSI for Knowledge Acquisition in CRNs

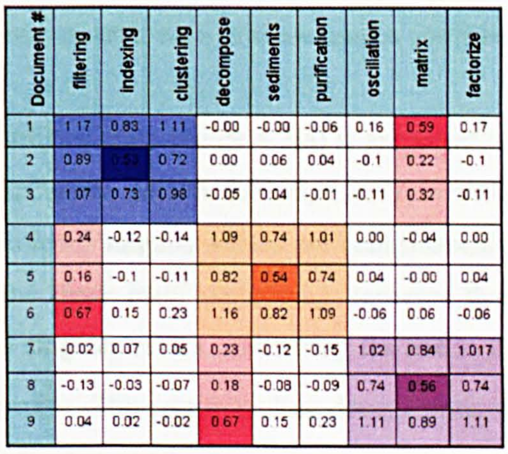
A core motivation for our discussion so far has been to facilitate automated acquisition of knowledge for TCBR systems. In this subsection, we illustrate using examples how LSI can be used to acquire relevance and similarity knowledge for CRNs.

4.3.1 Using LSI for Relevance Knowledge Mining

For mining relevance knowledge, we use the formulation in (4.1), which provides a lower rank approximation $\hat{M}$ corresponding to the original case feature matrix M. As we have discussed in Section 4.2, the rows of $\hat{M}$ are the new representations of the cases in terms of the original feature set. These case representations can in turn be mapped to a CRN, where each element in the matrix $\hat{M}$ defines the relevance of a term to a document.



Document #	filtering	indexing	clustering	decompose	sediments	purification	oscillation	matrix	factorize
1	1	1	1	0	0	0	0	1	0
2	1	0	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0
4	0	0	0	1	1	1	0	0	0
5	0	0	0	1	0	1	0	0	0
6	1	0	0	1	1	1	0	0	0
7	0	0	0	0	0	0	1	1	1
8	0	0	0	0	0	0	1	0	1
9	0	0	0	1	0	0	1	1	1



Document #	filtering	indexing	clustering	decompose	sediments	purification	oscillation	matrix	factorize
1	1.17	0.83	1.11	-0.00	-0.00	-0.06	0.16	0.59	0.17
2	0.89	0.53	0.72	0.00	0.06	0.04	-0.1	0.22	-0.1
3	1.07	0.73	0.98	-0.05	0.04	-0.01	-0.11	0.32	-0.11
4	0.24	-0.12	-0.14	1.09	0.74	1.01	0.00	-0.04	0.00
5	0.16	-0.1	-0.11	0.82	0.54	0.74	0.04	-0.00	0.04
6	0.67	0.15	0.23	1.16	0.82	1.09	-0.06	0.06	-0.06
7	-0.02	0.07	0.05	0.23	-0.12	-0.15	1.02	0.84	1.017
8	-0.13	-0.03	-0.07	0.18	-0.08	-0.09	0.74	0.56	0.74
9	0.04	0.02	-0.02	0.67	0.15	0.23	1.11	0.89	1.11

Figure 4.2 Relevance values mined using LSI

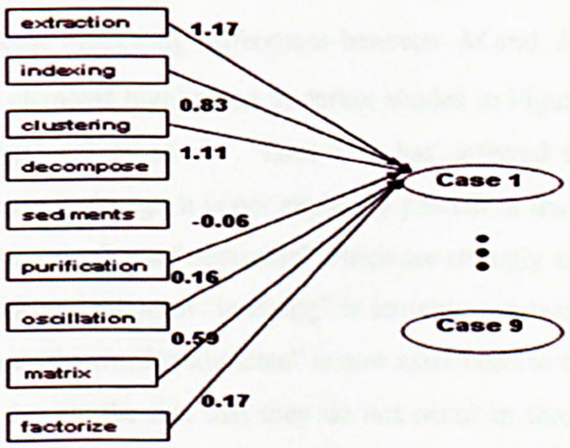


Figure 4.3 A CRN constructed with the acquired relevance weights

Figure 4.2 (a) shows an example casebase, which is same as the one in Figure 2.2. We have nine documents and nine terms representing three broad concepts: CBR, chemistry and Linear Algebra. The elements in the matrix that pertain to these concepts are shown in shades of grey. The three elements shown in darker shades are interesting because they identify a departure from the norm. These elements relate to the use of the Linear Algebra term "matrix" in Case 1 which deals with CBR; the use of the CBR term "filtering" in Case 6 which relates to chemistry; and the use of the chemistry word "decompose" in Case 9 which belongs to Linear Algebra. In as far as they highlight the applicability of these words in more than one context, each of these usages could be regarded as polysemous.

When the case-feature matrix M constructed from Figure 4.2(a) is subjected to SVD, the singular values obtained are 3.1873, 2.6940, 2.6185, 1.0786, 0.8071, 0.7094, 0.4730, 0.4306 and 0.0000. It is clear that the top 3 singular values are conspicuously bigger than the rest, perhaps pointing to the three main underlying concepts that describe this casebase. By retaining the top 3 singular values, and setting the rest to 0, we obtain a 3-rank approximation to M, say $\hat{M}$. The corresponding document-term matrix is shown in Figure 4.2 (b). The new relevances of features to cases can be read out from this matrix.

We now examine some interesting differences between M and $\hat{M}$. We focus on how the values of the elements highlighted in darker shades in Figure 4.2(b) have changed from their original values in M. Thus LSI has inferred that the term "indexing" is relevant to case 2, though it is not explicitly present in that case. This is because case 2 has terms "filtering" and "clustering" which are strongly associated with the underlying concept CBR and the term "indexing" is strongly representative of that concept. For similar reasons, the word "sediments" is now associated to case 5, and the word "matrix" to case 8, despite the fact that they do not occur in those cases. This illustrates the ability of LSI to exploit co-occurrence patterns to infer implicit semantic associations within a casebase. We also note that the relevance of polysemous terms to their cases have been attenuated by LSI. For example, the relevance of "decompose" to case 9 has been diminished from 1 to 0.67. A possible explanation for this is as follows: The meaning of the term "decompose" as used in Case 9, is different from the "average meaning" of the term. It may be noted that LSI assigns a unique location (a set of co-ordinates) to each term (and each document) in the concept space. This location can be intuitively regarded as capturing the global meaning of the term, as averaged from several local meanings. In our current example, since the word "decompose" has been more often used in the context of chemistry than in Linear Algebra, the global meaning will show a greater belongingness to the concept of chemistry than to Linear Algebra. Since Case 9 is predominantly about Linear Algebra, LSI attempts to strike a balance between the following two conflicting requirements: the first that "decompose" actually occurs in this case and therefore should be considered relevant, and the second that the average meaning of "decompose" is conceptually not aligned to the main theme of the case. Figure 4.3 shows how the LSI generated relevance values can be implanted into a CRN.

We can extrapolate our discussion so far to see why $\hat{M}$ is better suited to facilitate retrieval of relevant documents compared to M. When retrieval is performed over $\hat{M}$, Case 2 can be retrieved in response to a query on "indexing", even though it does not have that term. This is because Case 2 has terms "filtering" and "clustering" which are conceptually related to "indexing", thus resulting in a non-zero relevance of "indexing" to case 2, and in consequent retrieval of case 2. We can regard this as an "implicit" query expansion (Manning et al, 2008 (expected)), where the query terms are augmented with additional terms that are semantically similar.

Figure 4.4 (a) shows an image from the USREMAIL domain, obtained using the stacking approach described in Chapter 3, and Fig, 4.4(b) shows its lower rank approximation generated by LSI. It is interesting to observe that the LSI image is relatively blurred; also the compressed LSI image is approximately 73% the size of the original compressed image.

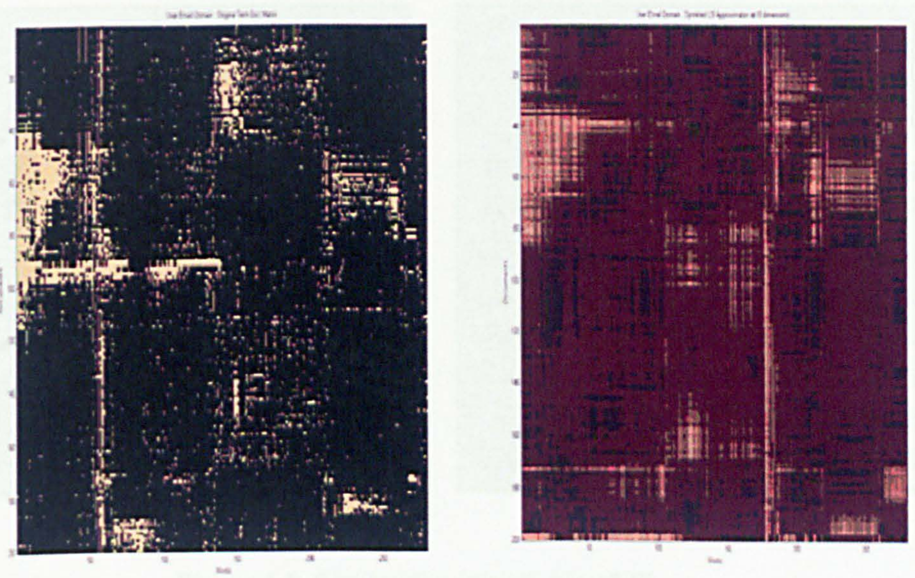


Figure 4.4 Stacked images from USREMAIL before and after LSI

4.3.2 Using LSI for Similarity Knowledge Mining

The method used for acquiring relevance values can easily be adapted for acquiring similarity knowledge as well. Each column of the approximation matrix $\hat{M}$ corresponds to the representation of a feature in terms of its relevance to the cases. Computing word similarities is thus simple: we take a dot product (or cosine similarity) between the corresponding columns of $\hat{M}$. The word similarities thus obtained can be compared against similarities derived from the original term document matrix M. Figure 4.5 shows term similarities before and after LSI, for the example document collection Figure 4.2(a). It is seen that LSI destroys the sparseness of the original similarity matrix.

	filtering	indexing	clustering	decompose	sediments	purification	oscillation	matrix	differential
filtering	4	2	3	1	1	1	0	1	0
indexing	2	2	2	0	0	0	0	1	0
clustering	3	2	3	0	0	0	0	1	0
decompose	1	0	0	4	2	3	1	1	1
sediments	1	0	0	2	2	2	0	0	0
purification	1	0	0	3	2	3	0	0	0
oscillation	0	0	0	1	0	0	3	2	3
matrix	1	1	1	1	0	0	2	3	2
differential	0	0	0	1	0	0	3	2	3

(a)

	filtering	indexing	clustering	decompose	sediments	purification	oscillation	matrix	differential
filtering	3.86	2.28	3.10	1.12	0.93	1.07	-0.13	1.21	-0.11
indexing	2.28	1.56	2.09	-0.05	0.04	-0.07	0.06	0.91	0.07
clustering	3.10	2.09	2.80	-0.04	0.10	-0.03	-0.04	1.13	-0.03
decompose	1.12	-0.05	-0.04	3.74	2.26	3.08	1.08	0.90	1.08
sediments	0.93	0.04	0.10	2.26	1.56	2.10	-0.05	0.03	-0.05
purification	1.07	-0.07	-0.03	3.08	2.10	2.84	-0.01	0.02	-0.01
oscillation	-0.13	0.06	-0.04	1.08	-0.05	-0.01	2.87	2.29	2.87
matrix	1.21	0.91	1.13	0.90	0.03	0.02	2.29	2.32	2.30
differential	-0.11	0.07	-0.03	1.08	-0.05	-0.01	2.87	2.30	2.87

(b)

Figure 4.5 Similarities mined using LSI

A second way of obtaining similarity knowledge is to use revised representations of words in the k-dimensional space and compute cosine or dot product similarity between these lower dimensional representations. We have seen that the matrix $\hat{U}$ in the decomposition of equation 4.1 contains the revised term representations. These are scaled using the matrix Σ_k to obtain the term matrix $\hat{U}\ \Sigma_k$, so that dimensions are weighted by the importance of concepts. The rows of $\hat{U}\ \Sigma_k$ are the new co-ordinates for terms. We can compute the dot product (or cosine similarity) between term representations derived from $\hat{U}\ \Sigma_k$. It can be shown that these two ways of computing word similarities are equivalent in that they produce the same similarity matrix (Deerwester et al., 1990).

4.4 Chapter Summary

In this chapter, we have looked at the mathematical foundations of factor analytic approaches, in particular LSI, to induce concepts from a collection of textual cases.

Linear algebraic techniques like matrix decomposition to extract eigenvectors are central to the theme. SVD, which is the heart of LSI, is an extension of the idea which realizes a dual mode factor analysis, which allows it to create a representation of both terms(features) and documents(cases) in terms of a common set of underlying concepts. We have presented intuitive arguments that explain why the revised representations yield improved retrieval effectiveness. Several issues of practical concern like choice of dimensionality, space and time efficiency, approaches for efficient updates, and using LSI for supervised classification tasks have been briefly covered. Finally, we have illustrated using examples how LSI can be used to mine relevance and similarity knowledge for CRNs in TCBR.

Chapter 5

Supervised Latent Semantic Indexing

You've got to stop looking at the big picture. Gunnar Grimnes

In the previous chapter, we discussed how LSI can be used to acquire relevance knowledge for CRNs. In this chapter, we focus on using LSI for acquiring relevance knowledge in supervised classification tasks. LSI has been used before in classification tasks. In (Gee 2003) LSI has been applied to spam classification, and performances competitive with Naïve Bayes classifier reported. Similarly, in a study by (Zelikovitz & Hirsh 2001), LSI-based classifiers have been extended to accommodate background knowledge. However, an inherent limitation of LSI when applied to classification is that it fails to exploit class knowledge of training documents. If taken into account, class knowledge can lead LSI to promote inferred associations between words representative of the same class, and attenuate word associations otherwise. In this chapter, we present approaches to incorporate class knowledge into LSI to produce revised document representations.

Section 5.1 discusses limitations of LSI in the context of supervised classification tasks. We present a novel theoretical framework for understanding LSI performance in classification tasks. In Section 5.2, we propose the idea of sprinkling, which integrates class knowledge into LSI. Sprinkling is a simple extension of LSI based on augmenting the set of features using additional terms that encode class knowledge. We present an intuitive analysis of why sprinkling works, and also identify factors that play a critical role in determining its effectiveness. Sprinkling is "naïve" in that it accords equal importance to all classes. Furthermore, when the resulting case representations are used for classification by more than one classifier, sprinkling fails to exploit the strengths and weaknesses of these classifiers. Section 5.3 presents a principled approach called Adaptive Sprinkling (AS) to address these issues; Chapter 7 presents empirical results to demonstrate the effectiveness of AS over diverse classification tasks including those where classes share ordinal or hierarchical relationships. Section 5.4 positions our work in the context of other related works, and identifies avenues for future work. Section 5.5 summarizes the main contributions of this chapter.

5.1 LSI in Classification Tasks

In Section 4.2.2, we showed that applying LSI in supervised classification tasks is a straightforward extension of its more commonplace application for retrieval. To make our discussion self-contained, we revisit this extension here briefly. We are given a collection of labelled training cases, and LSI is used to construct lower dimensional representations of these cases. An incoming unlabelled test case is treated as a query, and positioned in the space of training cases. This allows us to retrieve the *k* training cases most similar to the test case; since these training cases are labelled, a weighted k nearest neighbour (w-kNN) algorithm can be used to arrive at a class assignment for the test case. It may be noted that w-kNN is not the only approach that can be used in conjunction with LSI; since LSI generated representations are founded on the vector space model (VSM), any technique founded on the VSM like the Support Vector Machine can be used to carry out the classification.

Most practical systems using LSI for supervised classification use wkNN over LSI generated representations as explained above. However, this approach has several shortcomings, all resulting from the fact that LSI fails to take into account class labels of training documents while constructing revised case representations. Let us consider the simplistic example in Figure 5.1(a) which shows cases originating from two classes distributed over two features. It is clear that Feature 2 is more useful than Feature 1 in discriminating between classes 1 and 2. However, the variance or spread of the cases across Feature 1 is more pronounced compared to Feature 2. Thus, assuming that features 1 and 2 coincide with the orthogonal features extracted by LSI, Feature 1 will be assigned more importance (as indicated by a higher singular value associated with it) compared to Feature 2. This can be understood intuitively in the light of our discussions in Section 4.2.1, where it was observed that LSI yields the best lower rank approximation in the least squares sense. LSI shows a preference for Feature 1, since projecting the cases onto Feature 1, will result in lesser "loss of information" than projecting them onto Feature 2.

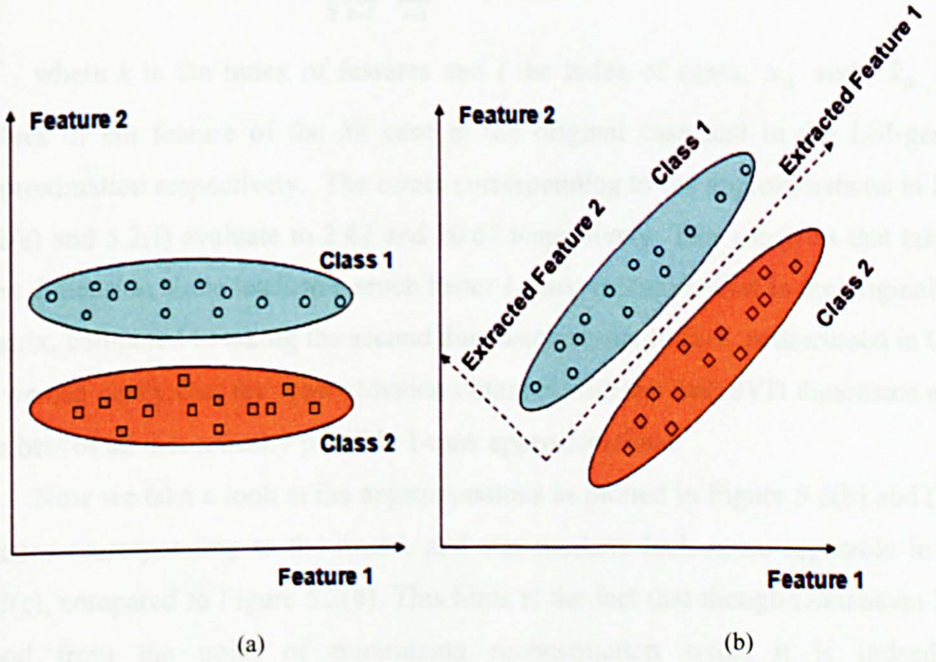


Figure 5.1 Hypothetical Case Bases to illustrate LSI's preference for dimensions

To better understand this, we take a simpler example of ten cases in Figure 5.2 that mimic the case distribution of the classes in Figure 5.1. The feature values of the five cases in Class 1 are (1,1.2), (1.8 ,1.3), (2.5,1.1), (3.6,1.2) and (4.5, 1.4). The five cases in Class 2 are (1.2, 2.3), (1.8, 2.2), (2.4, 2.4), (3.6, 2.1) and (4.4, 2.3). The case feature matrix is shown in Figure 5.2(d). We perform an SVD on this matrix. The singular value attached with the first LSI dimension is 10.67, while SVD associated with the second dimension is 2.42. When only the first dimension is retained, we obtain the approximation shown in Figure 5.2(e) Figure 5.2(f) shows the approximation when only the second dimension is retained.

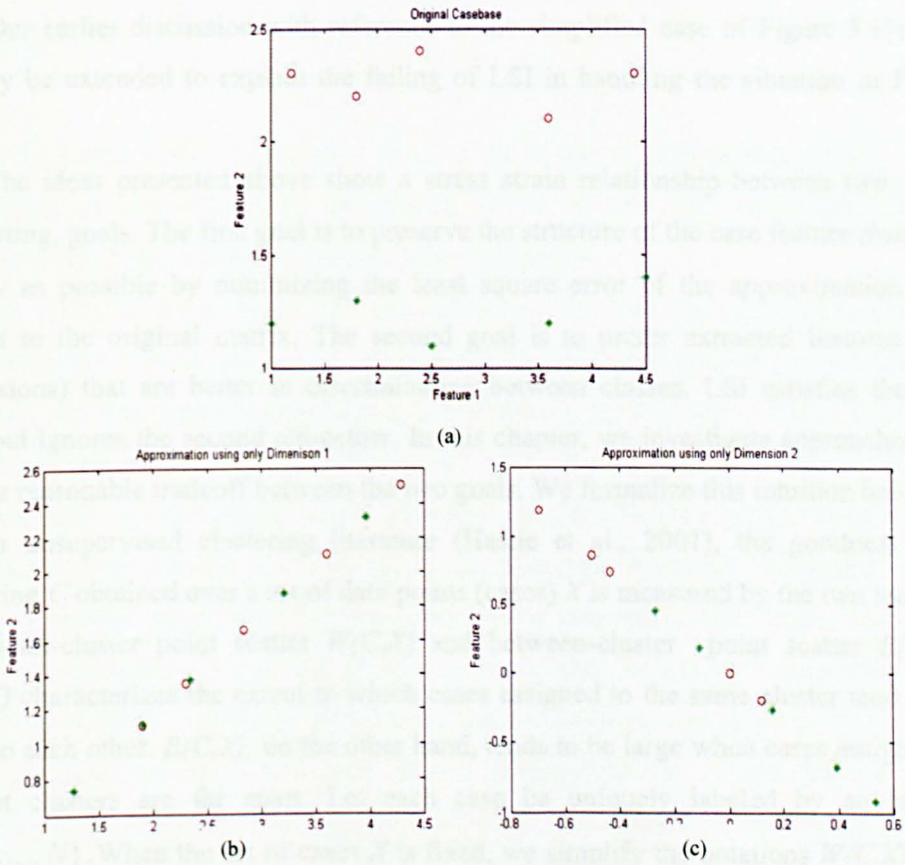
Based on the formulation in Section 4.2.1, we compute the Frobenius norm of differences between an approximation X and the original matrix as:

$$E(X)=\sqrt{\sum_{k=1}^{2}\sum_{i=1}^{10}(x_{ik}-\hat{x}_{ik})^2}$$

where k is the index of features and i the index of cases, x_{ik} and $\hat{x}_{ik}$ are the values of kth feature of the ith case in the original case and in the LSI-generated approximation respectively. The errors corresponding to the approximations in Figures 5.2(e) and 5.2(f) evaluate to 2.42 and 10.67 respectively. This confirms that taking the first dimension alone leads to a much better 1-rank approximation to the original 2-rank matrix, compared to taking the second dimension alone. In fact, as discussed in Chapter 3, we can verify that the approximation obtained with the first SVD dimension alone is the best of all theoretically possible 1-rank approximations.

Now we take a look at the approximations as plotted in Figure 5.2(b) and (c). The classes corresponding to the round, and star markers look more separable in Figure 5.2(c), compared to Figure 5.2(b). This hints at the fact that though Dimension 2 is not good from the point of minimizing reconstruction error, it is indeed more discriminative of features compared to Dimension 1.

The above example shows that LSI is handicapped in its absence of class knowledge and thus may extract features which are not the best from a classification standpoint. Extracted features corresponding to the top k singular values correspond to class structure only when features from cases belonging to different classes do not overlap. However, more often than not, textual cases pertaining to different classes are likely to share features because of polysemous and context driven usage of words and large word choice variability. Furthermore, infrequent features with high discriminatory power may be treated by LSI as noise, and filtered out.



(a) (b) (c)

$$
\begin{pmatrix}
1.0000 & 1.2000 \\
1.8000 & 1.3000 \\
2.5000 & 1.1000 \\
3.6000 & 1.2000 \\
4.5000 & 1.4000 \\
1.2000 & 2.3000 \\
1.8000 & 2.2000 \\
2.4000 & 2.4000 \\
3.6000 & 2.1000 \\
4.4000 & 2.3000
\end{pmatrix}
\quad
\begin{pmatrix}
1.2683 & 0.7405 \\
1.9084 & 1.1143 \\
2.3434 & 1.3683 \\
3.2072 & 1.8727 \\
3.9655 & 2.3154 \\
1.8964 & 1.1073 \\
2.3003 & 1.3431 \\
2.8349 & 1.6553 \\
3.5991 & 2.1015 \\
4.2828 & 2.5007
\end{pmatrix}
\quad
\begin{pmatrix}
-0.2683 & 0.4595 \\
-0.1084 & 0.1857 \\
0.1566 & -0.2683 \\
0.3928 & -0.6727 \\
0.5345 & -0.9154 \\
-0.6964 & 1.1927 \\
-0.5003 & 0.8569 \\
-0.4349 & 0.7447 \\
0.0009 & -0.0015 \\
0.1172 & -0.2007
\end{pmatrix}
$$

(d) (e) (f)

Figure 5.2 An Example

The features extracted by the LSI seldom coincide with the original set of features; Figure 5.1 (b) shows a more realistic situation where the extracted LSI features are different from the original. We observe that the extracted feature 2 is more discriminating than the extracted feature 1, which is declared to be more important by LSI. Our earlier discussion with reference to the simplified case of Figure 5.1(a) can directly be extended to explain the failing of LSI in handling the situation in Figure 5.1(b).

The ideas presented above show a stress strain relationship between two, often conflicting, goals. The first goal is to preserve the structure of the case feature matrix as closely as possible by minimizing the least square error of the approximation with respect to the original matrix. The second goal is to prefer extracted features (LSI dimensions) that are better in discriminating between classes. LSI satisfies the first goal, but ignores the second altogether. In this chapter, we investigate approaches that strike a reasonable tradeoff between the two goals. We formalize this intuition below.

In unsupervised clustering literature (Hastie et al., 2001), the goodness of a clustering *C* obtained over a set of data points (cases) *X* is measured by the two metrics: the within-cluster point scatter *W(C,X)* and between-cluster point scatter *B(C,X)*. *W(C,X)* characterizes the extent to which cases assigned to the same cluster tend to be close to each other. *B(C,X)*, on the other hand, tends to be large when cases assigned to disjoint clusters are far apart. Let each case be uniquely labeled by an integer $i \in \{1,\ldots,N\}$. When the set of cases *X* is fixed, we simplify the notations *W(C,X)* and *B(C,X)* to *W(C)* and *B(C)* respectively, which are given by

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} d(x_i, x_i')$$

$$B(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')\neq k} d(x_i, x_i')$$

where $d(x_i, x_i')$ is the distance between two cases x_i and x_i' as computed using a standard distance measure like the Euclidean distance. K is the number of clusters, and C(i) returns the class label of the ith case. We can define the total point scatter T as

$$T = W(C) + B(C)$$

It may be noted that T is constant given a set of cases, irrespective of the cluster assignment. A good clustering assignment would be one that maximizes *B(C)* or minimizes *W(C)*, for a given number of target clusters K.

We will now try to adapt this formulation to deal with the supervised case. The class labels are known and can be equivalently thought of as defining a clustering C. However, the data points corresponding to the cases x_i are no longer fixed, but decided by the choice of our representation. One such representation is the lower-rank approximation to the original cases generated by LSI. Feature generalization using Association Rule Mining as in Propositional Semantic Indexing (Wiratunga et al., 2005a) generates yet another representation in the original feature space. Instead of evaluating goodness of clustering given a representation, we are now interested in the dual of the problem, intuitively stated as: Given a clustering C as enforced by the class labels of cases, how good is a given choice of representation in ensuring that cases belonging to the same class are close to each other, and cases from disjoint classes are far apart. In other words, we are interested in finding X, which represents an assignment of data points x_i in the feature space, that leads to minimizing *W(C,X)* and maximizing *B(C,X)*, for a given C. Since C is held constant, we simplify the notations *W(C,X)* and *B(C,X)* to *W(X)* and *B(X)* respectively. In our discussion henceforth, we focus on minimizing the function *G(X)* = *W(X)/B(X)*, though alternative formulations are possible.

It is easily seen that given this problem definition, the "best" representation would be one that collapses all cases belonging to a certain class to a single data point, thus resulting in each case in the casebase mapped to one of the *K* distinct points corresponding to each class. This has the effect of minimizing W(X) to 0. However, this trivial solution is useless for all practical purposes, since it has very poor generalization over an unseen test case; also, from a k-NN perspective, the representation would be quite useless for retrieval since all knowledge of differences between cases within a given class have been lost. Thus the system cannot say which of the several cases within a given class is most similar to a given test case. Hence certain constraints need to be added to the above definition to tighten up the problem definition and make it useful in practice. These constraints come from the need to preserve the structure of the original case-feature matrix, and patterns within it.

We have already defined the error in approximating the original case feature matrix in Chapter 4 using the Frobenius Norm. We use this to define the second optimization criterion, which we call called *E(X)*; we are interested in minimizing *E(X)*. An extreme case of using the original case feature matrix suggests itself, in that it makes *E(X)* = 0. However, it has poor generalizability and is unsurprisingly unimpressive in its classification effectiveness. On the other extreme, it can be seen that the situation described above of collapsing all cases belonging to the class to one datapoint results in a very large *E(X)*, suggesting a large departure from the original case-feature matrix structure. To sum up, thus, *our goal is to opt for a representation X that strikes a reasonably good tradeoff between satisfying the conflicting goals of minimizing G(X) and minimizing E(X).*

We refer to this as the structure versus class-knowledge dilemma, and in conjunction with the well studied bias-variance trade-off (Mitchell 1997), will serve as a useful tool in devising our algorithms, and explaining empirical results from experimental evaluations.

5.2 Sprinkling

In the last section, we have identified two limitations of LSI in a classification setting. Firstly, clusters defined by extracted features correspond neatly to class structure only when there are not many overlapping terms in cases from different classes. Secondly, infrequent words with high discriminatory power are watered down. In this section we address these issues by directly incorporating class knowledge into the representation used as input by SVD, which forms the heart of LSI.

The basic idea is simple: we generate a set of artificial terms corresponding to the class labels of the training cases. These artificial terms are then appended to the feature set of the training cases. We refer to this process as 'sprinkling'. The case-feature matrix of Figure 5.3(a) has three classes each having three cases as shown. We obtain the augmented case feature matrix of Figure 5.3(b) by adding three new features corresponding to the three class labels. These features can be thought of carriers of class knowledge.

Figure 5.4 shows a schematic of the processes involved in using the idea of sprinkling for classification. To start with, LSI is carried out on the augmented case-feature matrix. Noisy dimensions corresponding to low singular values are dropped and a lower-rank approximation of this matrix is obtained as usual. The approximation matrix containing the revised case representations has the same dimensionality as the augmented matrix. However, we do not know the class labels of the incoming test documents. Therefore, to make training document representations compatible with the test document, the columns corresponding to additional sprinkled features are dropped from the augmented matrix. This step is referred to as "unsprinkling" in Figure 5.4. Test documents are now classified using weighted kNN using an Euclidean distance metric in the usual manner. From a knowledge acquisition perspective, the relevance values in a CRN can be read out from the case representations in the case-feature matrix after unsprinkling.

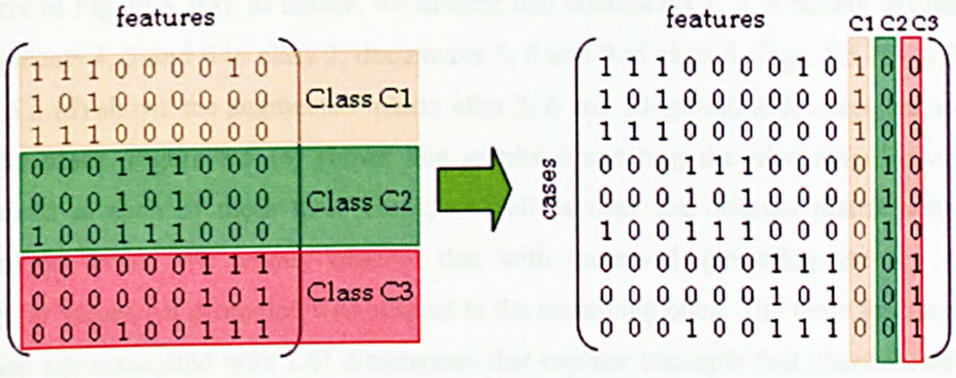


Figure 5.3 Sprinkling

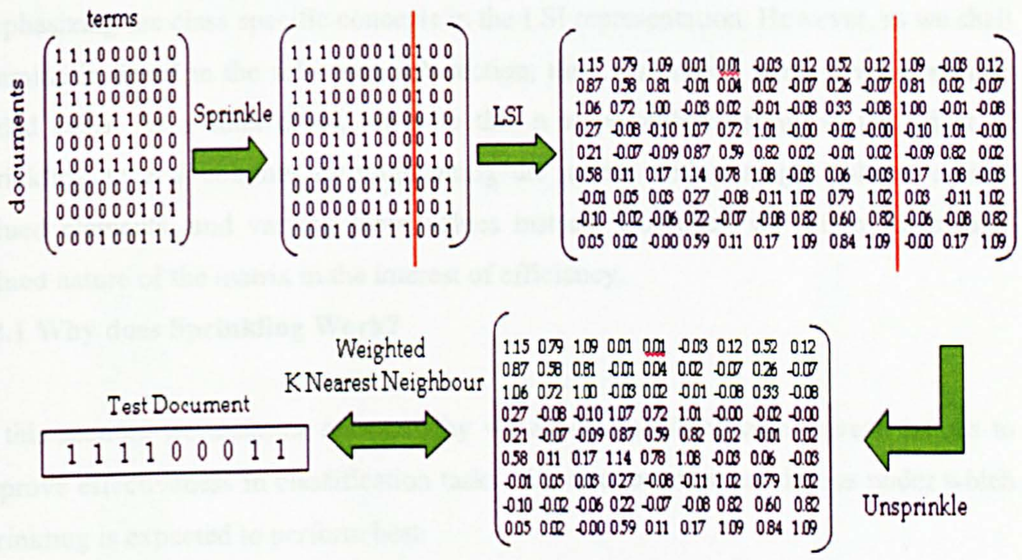


Figure 5.4 Classification using sprinkled LSI

While we have sprinkled only one additional feature per class, in principle we can add more than one artificial feature for each class. This gives rise to interesting possibilities. We observe that larger the number of sprinkled features per class, the more the contribution of the class knowledge of the training cases in the generated LSI representations, and hence in the consequent classification process. It is illustrative to study the singular values associated with the LSI dimensions obtained after the SVD of the augmented case feature matrix, and observe how these singular values change as a function of the number of sprinkled features. Figure 5.5(a) shows the case feature matrix in Figure 5.3(a); as before, we assume that documents 1, 2, 3 belong to class 1; documents 4, 5 and 6 to class 2; documents 7, 8 and 9 to class 3. Figs. 5.5 (b), 5.5 (c) and 5.5 (d) shows the augmented matrix after 3, 6 and 18 sprinkled features are added to the cases. Figure 5.5 (e) shows line graphs connecting the nine singular values obtained in each of these three cases, as well as over the original matrix with no sprinkled terms. We readily observe that with increased sprinkling the top three singular values get promoted with respect to the remaining ones. The three top singular values are associated with LSI dimensions that capture concepts that characterize the three classes. Thus adding larger number of sprinkled terms has the effect of emphasizing the class specific concepts in the LSI representation. However, as we shall examine in detail in the following sub-section, this "distortion" is not always without added costs. As a final detail, we note that it is possible to simulate the effect of sprinkling several columns by augmenting the matrix with a single column of real valued elements, and varying these values instead. However, we retain the binary-valued nature of the matrix in the interest of efficiency.

5.2.1 Why does Sprinkling Work?

In this section, we examine reasons why we expect sprinkled case representations to improve effectiveness in classification tasks. We also examine conditions under which sprinkling is expected to perform best.

Structure versus Class Knowledge Argument

In Section 5.1, we highlighted that a good choice of representations must strike a tradeoff between minimizing *G(X)*, defined as the ratio of within-cluster point scatter and between-cluster point scatter, and minimizing *E(X)*, which quantifies the distortion of the new representation with respect to the original one.

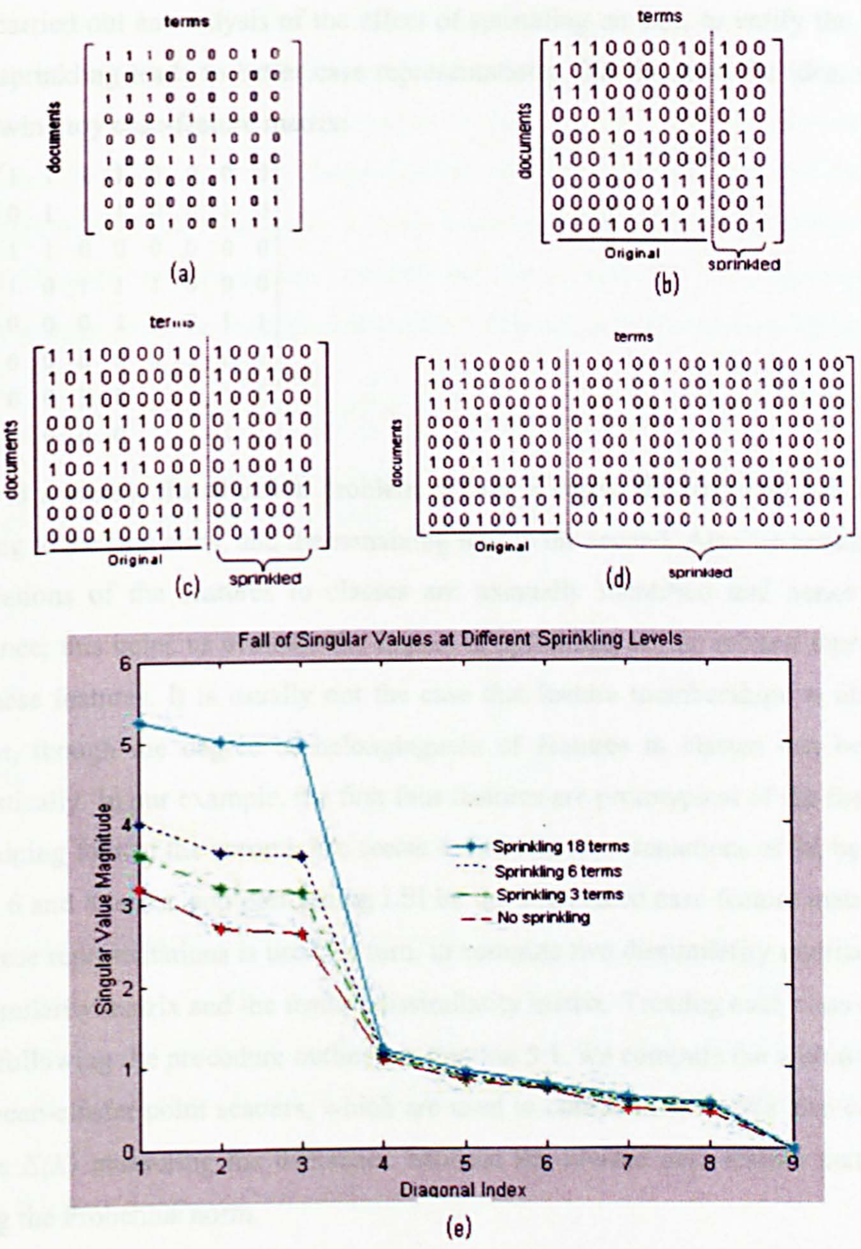


Figure 5.5(a) Original Term Doc Matrix (b) Matrix After Sprinkling 3 terms (c) Matrix After Sprinkling 6 terms (d) Matrix after Sprinkling 18 terms (e) Fall of Singular Values before and after sprinkling

We carried out an analysis of the effect of sprinkling on LSI, to verify the hypothesis that sprinkling leads to better case representations. To illustrate the idea, we use the following toy case-feature matrix:

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

This is a binary classification problem. Of the 8 cases, the first four are assumed to belong to the first class, and the remaining four to the second. Also we assumed that the affiliations of the features to classes are manually identified and hence known in advance; this helps us evaluate the impact of sprinkling on the revised representations of these features. It is usually not the case that feature memberships to classes are a given, though the degree of belongingness of features to classes can be estimated statistically. In our example, the first four features are prototypical of the first class, the remaining four of the second. We create 4 different representations of *M*, by sprinkling 2, 4, 6 and 8 terms, and performing LSI on the augmented case feature matrices. Each of these representations is used, in turn, to compute two dissimilarity matrices, the case dissimilarity matrix and the feature dissimilarity matrix. Treating each class as a cluster and following the procedure outlined in Section 5.1, we compute the within-cluster and between-cluster point scatters, which are used to compute *G(X)*. We also compute the value *E(X)* measuring the difference between the revised case-feature matrix and *M*, using the Frobenius norm.

Figure 5.6 shows that with increased number of sprinkled terms *G(X)* falls conspicuously. However, this is accompanied by the fact that sprinkled LSI distorts the original term document matrix D to a class-enriched LSI approximation D_S. We note that D_S is no longer the best k-rank approximation to the D in the least-square sense. The vertical axes of the graphs in Figure 5.6 show the mean square of errors between D

and D_S, given by $E(X)$. The number alongside each marker point in the line graphs shows the number of sprinkled terms. We readily observe that the reduction in $G(X)$ achieved by sprinkling is at the cost of losing information on D, as indicated by an increase in $E(X)$. Thus very large number of sprinkled terms may be detrimental to classification performance, as it may over-emphasise class-knowledge. Ideally we would like a trade off between ``under-" and ``over-sprinkling", that gives us the best of both worlds: improve class-discrimination while not overlooking specific patterns in D.

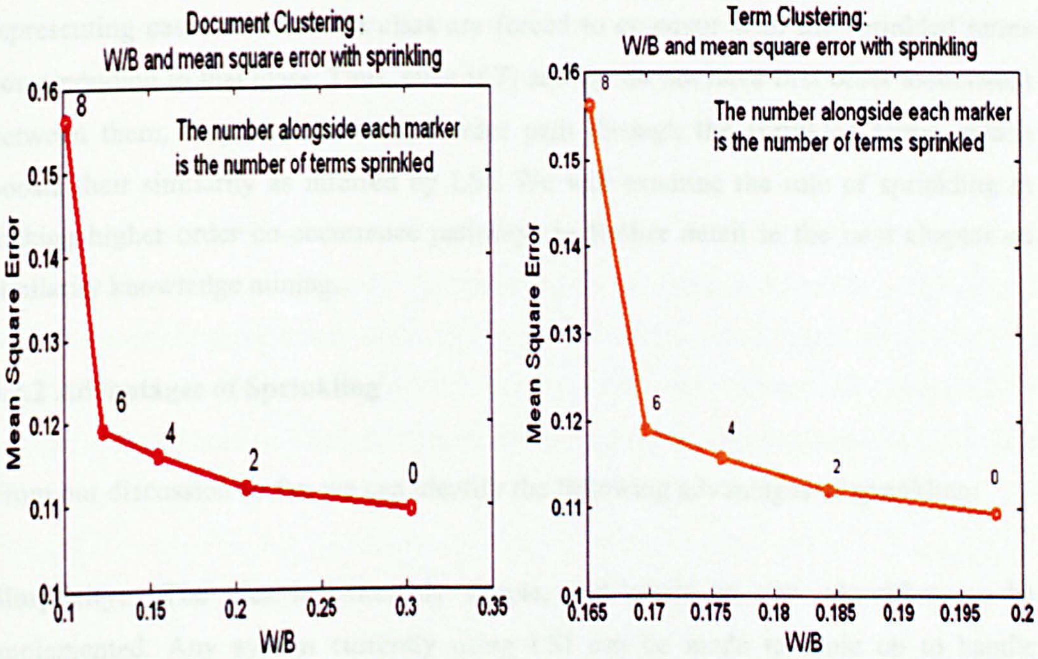


Figure 5.6 Illustrating the tradeoff between reducing W/B ratio ($G(X)$) and reducing the mean square error distortion with respect to the original case feature matrix ($E(X)$).

The Higher Order Co-occurrence Argument

Sprinkling aims to make explicit any implicit associations between terms indicative of underlying classes. Since sprinkled features are essentially class labels, including them helps to artificially promote co-occurrences between existing terms and classes. More specifically, the performance of sprinkled LSI in classification tasks can be explained using empirical observations made in (Kontosthathis & Pottenger 2006), which were summarised in Section 4.2.3. Their work reveals a close correspondence between LSI and higher order associations between terms. Kontostathis and Pottenger (2006) provide experimental evidence to show that LSI boosts similarity between terms sharing higher order associations. In the light of this observation, it is interesting to note that sprinkled terms boost second-order associations between terms related to the same class, hence bringing them closer. This is because two terms T_1 and T_2 representing cases of the same class are forced to co-occur with the sprinkled terms corresponding to that class. Thus, even if T_1 and T_2 do not have first order association between them, they share a second-order path through the sprinkled terms, which boosts their similarity as inferred by LSI. We will examine the role of sprinkling in making higher order co-occurrence pathways in further detail in the next chapter on similarity knowledge mining.

5.2.2 Advantages of Sprinkling

From our discussion so far, we can identify the following advantages of sprinkling:

Simplicity: The idea is extremely simple, and needs no new algorithms to be implemented. Any system currently using LSI can be made to scale up to handle supervised domains by a preprocessing step that adds columns corresponding to the additional terms to the case feature matrix. The basic apparatus of SVD, and incremental update algorithms associated with SVD remain unaffected as a result of this change.

Representation Richness: Sprinkling generates revised case representations that integrate well within instance based learners like k Nearest Neighbours. More generally, the generated representations can be used by any mechanism founded on the vector space model; in Chapter 7 we present empirical results suggesting that SVMs benefit by using sprinkled LSI representations instead of the original LSI representations. Secondly, sprinkled LSI representations can be used for unsupervised tasks like case retrieval, and retrieval quality is expected to benefit from the incorporation of class knowledge. Thirdly, as with LSI, sprinkling can lead to revised feature representations and feature similarity knowledge that incorporates class knowledge. Finally, we note that though sprinkling has been discussed in the context of LSI, the idea can be decoupled from LSI and exploited by other learners as well. In the following chapter, we show how sprinkling can be used in conjunction with an approach that mines similarity knowledge using higher order association between features.

Efficiency: Sprinkled features are typically far fewer in number compared to the size of the original feature space. In our experiments reported in Chapter 7, significant improvements were obtained when as low as 8 artificial terms were sprinkled to an original representation comprising 1000 features. Thus the overheads in terms of SVD computation are minimal. Empirical evaluations supporting this are reported in Section 7.2.

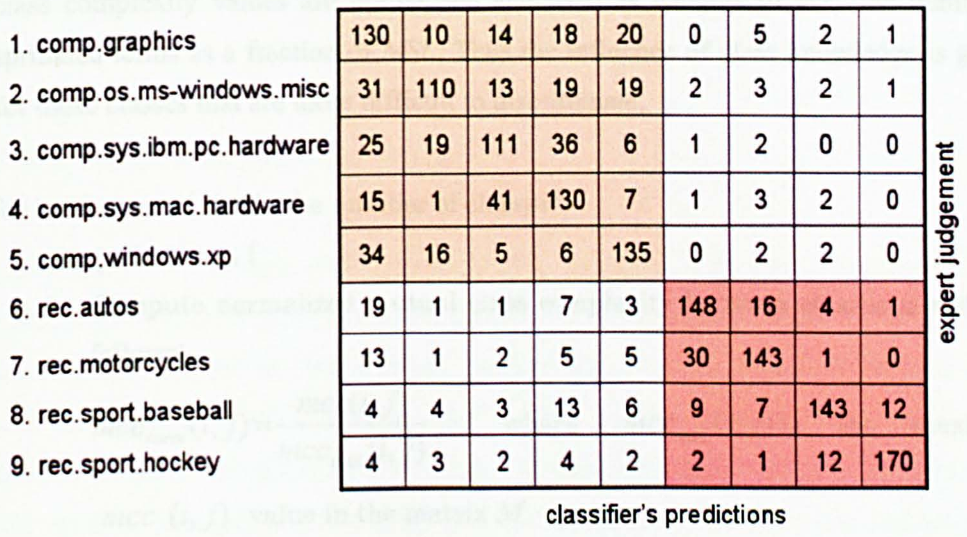
5.3 Adaptive Sprinkling

The basic sprinkling approach treats all classes equally. This is a limitation for the many multi-class problems with explicit relationships between classes. Two examples are hierarchical classes and ordinal classes. An example of hierarchical classification is the Yahoo directory, which is a manually created and maintained library of web sites organized into categories and subcategories. Subcategories (say *dogs*) have is-a relationships with their parents (say *mammals*). Ordinal classes are common in sentiment analysis domains. In a movie review domain, we may have ratings 1 to 5, each rating treated as a class. Reviews rated 1 are similar to those rated 2 in that they both express a negative polarity, but dissimilar to those rated 5 which carry a positive polarity. Ignoring inter-class relations in both the hierarchical an ordinal scenarios may have an adverse effect on the effectiveness of the sprinkled representations. Furthermore, even in scenarios where classes have no explicit relationship between them, some classes are more easily separable than others, so the number of sprinkled terms should depend on the complexity of the class decision boundary. A second limitation of sprinkling is revealed when the case representations generated by sprinkled LSI are used by other classifiers like SVM. Classes found confusing by a kNN classifier could be different from those found confusing by SVM, and ideally the sprinkling process should adapt to classifier needs. Adaptive Sprinkling (AS) is motivated by the need to address the aforementioned limitations of sprinkling.

There are two broad ways of incorporating knowledge of inter-class relationships into case representations generated by sprinkling. The first is the explicit approach, where the similarity between classes is captured using an explicit formulation that captures the inter-class similarity. One can model the relationship between classes in a movie review domain by assuming that the distance between classes is a linear function of the absolute difference in the ratings (Mukras et al., 2007). Similarly, for hierarchical classes, we can use one of the several distance measures like the Wu Palmer distance, the Resnik distance or the path-length distance (Pederson et al., 2004), which can be used to compute similarity between any two classes. One way of incorporating the explicit inter-class similarity knowledge into classification is to use it to bias the feature selection process; an example is (Mukras et al., 2007) where the Information Gain approach is adapted to take into account class relationships in ordinal datasets. There are several disadvantages to this approach of explicitly modelling interclass relationships. Firstly, most explicit formulations fail to take into account asymmetry between classes. Thus in the movie review domain, the similarity between classes 1 and 2 will be reckoned to be the same as that between classes 2 and 3. This may not be a reasonable assumption, given the fact that classes 1 and 2 are likely to share more vocabulary given that they are both negative in their sentiment orientation, while class 3 is likely to be more diverse in its choice of words.

Thus the interclass similarity as formulated by an explicit relation may have little or no grounding on the actual overlap of vocabularies between cases originating from different classes, and thus fails to recognize that certain pairs of classes are harder to separate than some others. Secondly, it is hard to arrive at a right choice of distance measure. In the case of the movie review domain, why should we favour a linear relationship over say one that uses an exponential decay instead? Similarly, which of the several approaches to model hierarchies is expected to work the best? Thirdly, the method fails in situations where classes have no explicit relationship between them. This is the most common classification scenario, where the classifier chooses between one or more of several unrelated classes. Henceforth we refer to such problems as orthogonal classification tasks. The name signifies that the classes are considered orthogonal with respect to each other, in the absence of any known relationship between them. As we noted before, in this case we would like the sprinkled terms to have a bearing on the inter-class complexity.

The second approach of influencing generated case representations using the knowledge of inter-class relationships is the implicit one, and this is our main contribution in this section. The key idea behind AS is to exploit confusion matrices generated by classifiers like kNN and SVM. Confusion matrices implicitly capture a wealth of knowledge about how classes are related to each other. A confusion matrix compares a classifier's predictions against expert judgements on a class-by-class basis. The non-diagonal values in this matrix are indicative of classes that the classifier finds hard to separate; the lower the values, the more easily separable the classes. Figure 5.7 shows a confusion matrix created from nine classes in the 20 NewsGroup text collection (Mitchell 1997) using the k-NN classifier. The classes shown are arranged in a hierarchy. The two broad trees are *comp* for computing and *rec* for recreation. Referring to classification errors in the example confusion matrix of Figure 5.7 we readily infer that classes 1 and 9 are easy to tell apart, while classes 1 and 2 are harder to discriminate. AS is based on the intuition that relatively more sprinkled terms are to be allocated between hard-to-discriminate classes. Interestingly, we found that confusion matrices also implicitly carry information about explicit class relationships as in ordinal and hierarchical classes. For example, in Figure 5.7, we see that the two shaded regions correspond to confusion between classes within the *comp* and *rec* subtrees. The confusion between classes from the two disjoint subtrees is smaller.



1. comp.graphics	130	10	14	18	20	0	5	2	1
2. comp.os.ms-windows.misc	31	110	13	19	19	2	3	2	1
3. comp.sys.ibm.pc.hardware	25	19	111	36	6	1	2	0	0
4. comp.sys.mac.hardware	15	1	41	130	7	1	3	2	0
5. comp.windows.xp	34	16	5	6	135	0	2	2	0
6. rec.autos	19	1	1	7	3	148	16	4	1
7. rec.motorcycles	13	1	2	5	5	30	143	1	0
8. rec.sport.baseball	4	4	3	13	6	9	7	143	12
9. rec.sport.hockey	4	3	2	4	2	2	1	12	170

expert judgement

classifier's predictions

Figure 5.7 A Confusion matrix from the hierarchical 20 NewsGroups domain

AS determines the number of sprinkled terms for each class from the confusion matrix. The confusion matrix is generated using the same classifier that will operate on the revised representations. Let q_{ij} be an element of the confusion matrix Q, showing the number of documents of class c_i being misclassified as class c_j. We define probability $P(j|i)$ as the probability of class c_i being misclassified as class c_j. This can be estimated from the entries in the confusion matrix as:

$$P(j \mid i) = \frac{q_{ij}}{\sum_k q_{ik}}$$

We then define the ``mutual complexity" between classes c_i and c_j as

$$mcc(i,j) = \frac{P(i \mid j) + P(j \mid i)}{2}$$

The asymmetric confusion matrix Q is now transformed into a mutual complexity matrix M, which is symmetric. The pseudo-code in Figure 5.8 shows how sprinkled terms can be generated based on the matrix M. The maximum sprinkling length *MSL* is empirically determined. In our experiments we used $MSL = 8$. We note that the mutual class complexity values are normalised and used as weights to vary the number of sprinkled terms as a fraction of *MSL*. Thus the influence of class knowledge is greater for those classes that are more difficult to discriminate.

```
for i = 1 to m-1 { /*m is the number of classes*/
    for j = i+1 to m {
    1. Compute normalized mutual class complexity between classes $c_i$ and $c_j$ as follows:
       $mcc_{norm}(i,j) = \frac{mcc(i,j)}{mcc_{\max}(i,j)}$ where $mcc_{\max}(i,j)$ is the maximum
       $mcc\ (i,j)$ value in the matrix $M$
    2. $s = \lfloor MSL \times mcc_{norm}(i,j) \rfloor$
    3. Sprinkle $s$ terms in all documents belonging to class $c_i$ and $s$ others in all
       documents belonging to $c_j$.
    }
}
```

Figure 5.8 Pseudo-code for Adaptive Sprinkling

In Chapter 8, we present experimental evaluations over orthogonal, ordinal and hierarchical classification problems. We compare the original confusion matrix with the one obtained by trying the same classifier on the case representations generated by AS, and also examine the feature similarities before and after sprinkling.

There are several advantages with using confusion matrices as implicit knowledge sources for mining inter class relations. Firstly, the knowledge engineering effort is reduced, since we can do away with modelling explicit associations between classes. Secondly, the same approach can be used to handle the three categories of classification problems : orthogonal, ordinal or hierarchical. We do not need to tailor the algorithm to suit these tasks. Finally, unlike explicit approaches, AS exploits inter-class complexity specific to a classifier to arrive at richer and more effective revised case representations.

5.4 Discussion and Related Work

There have been several efforts in the past to extend LSI to text classification tasks. Zelikovitz and Hirsh (2001) consider using a set of background texts in addition to the training data for use in classification. Background texts are hopefully relevant to the text classification domain and are used to find training examples that could not be found by a simple comparison between the text example and training set. For example, if a piece of background knowledge is found similar to both a training example and a test example, the training example is considered similar to the test case, even if they do not share any terms. The approach is especially suitable when the training data set is small. The significant difference with sprinkling is that instead of using an extended corpus for operation, we attempt to integrate additional knowledge using a synthetic set of terms that reflects the underlying class structure.

Sun et al. (2004) recently presented a technique called SLSI that is based on iteratively identifying discriminative eigenvectors from class-specific LSI representations. SLSI involves *km* SVD computations, corresponding to *k* iterations over *m* classes, making it computationally expensive. In their study, no significant improvement of SLSI over baseline SVM was reported. In Chapter 8, we will present empirical studies demonstrating that sprinkled representations perform significantly better than SVM over most of our experimental datasets.

Wang et al (2005) have an objective similar to ours; they present a theoretical model to extend LSI to capture classification knowledge considering two matrices: a term-document matrix and a document-class matrix. While the authors do not present any experimental validation for their algorithms, they observe that the algorithm slows down in situations where a document can belong to more than one class. In contrast, sprinkled terms can comfortably represent affiliations of documents to more than one class. This should have no conspicuous adverse effect on time performance.

Wiener et al. (1995) approach the problem of text classification using LSI by conceiving of a local LSI in addition to the global (or the usual) LSI. In local LSI, a separate LSI representation is created for each category. The local representations are compared separately with an incoming test document. There are two main disadvantages to this approach. The first is that since each local LSI representation is created separately, the resulting similarities are not easily comparable. The second is the computational overhead of making and maintaining several LSI representations and the run-time overheads in processing the query separately against each local representation.

Zelikovitz (2004) proposes transductive LSI for text classification. Rather than performing SVD only on the training data, they use an expanded term-document matrix that includes the test data as well. The classification accuracy improves because more data is used. This work can, in effect, be viewed as a special case of (Zelikovitz and Hirsh, 2001) where the test data is treated as background knowledge. However, neither of these approaches takes into account class labels.

When compared to Adaptive Sprinkling, a general shortcoming of all of the above mentioned approaches is that they fail to take into account relationships between classes. A second relative strength of our approach is that it is simple and can easily be integrated into existing LSI implementations. Unlike most of the approaches above, the time complexity of AS is independent of the number of classes. In all our benchmark experiments reported in Chapter 8, computing SVD over an augmented term-document matrix takes less than 5% additional time compared to SVD on the original matrix.

Finally, it is important to note that sprinkling, though presented in the context of LSI, is a fairly general strategy that can be used to benefit other approaches that attempt to learn relevance or similarity knowledge. In the next chapter, we see how the idea can be used in conjunction with a similarity knowledge mining algorithm to result in feature similarity knowledge that respects class knowledge of the training documents. We also note that while we have used sprinkling to incorporate class knowledge, it would be interesting to see if the basic idea can be extended to incorporate other types of knowledge as well. An interesting possibility is the encoding of background knowledge as sprinkled features. An example would be to use sprinkling to encode feature similarities as obtained from linguistic resources like WordNet (Miller, 1995) or case associations as mined from Web resources like Wikipedia (Gabrilovich & Markovitch, 2007). This may lead to scenarios where we may not only append artificial features to cases, but also artificial cases to features. We envisage that the simple idea of sprinkling may give birth to a framework for comprehensively integrating introspective and background knowledge.

5.5 Chapter Summary

In this chapter, we have presented techniques to incorporate class knowledge into LSI with the goal of improving effectiveness in supervised classification tasks. The first approach is sprinkling. The basic idea is to augment cases with additional features based on class labels, and do SVD on the augmented case feature matrix, so that the dimensions extracted by LSI are influenced by class knowledge. Sprinkling is an extremely simple approach that can easily be integrated with any existing LSI application. Also it is efficient and involves minimal computational overheads. The second approach Adaptive Sprinkling extends sprinkling by incorporating knowledge of inter-class complexity derived from confusion matrices. This expands the scope of our approach to handle diverse classification problems involving orthogonal, ordinal or hierarchical relationships between classes.

Chapter 6

Learning Similarities from Higher Order Co-occurrences

I link therefore I am. S.J. Singer

Similarity knowledge for CRNs can be acquired either introspectively, or using background knowledge. The former relies on inferring similarity relations directly from patterns hidden within the given collection of texts, while the latter uses external linguistic resources like Wordnet or Roget's thesaurus, or web resources like Wikipedia. The approach presented in this chapter is primarily an introspective one, in that we rely on statistical properties of the collection and ignore linguistic relationship (like syntactical categories) between features.

The rest of the chapter is organized as follows. Section 6.1 sets out the motivation for our research. Section 6.2 explains the concept of higher order associations, along with algorithms to mine the same. Section 6.3 describes our model of word similarities. Section 6.4 shows how the parameters of this model can be determined automatically. In Section 6.5, we present a novel approach of influencing the similarity values based on class knowledge, along with empirical results. Section 6.6 shows some examples of mined associations. Section 6.7 discusses our work in the context of earlier relevant research. We sum up our main contributions in Section 6.8.

6.1 Motivation

Most early IR and TCBR systems were built on the assumption that features are independent of each other. While this assumption simplified the design of systems and facilitated efficient retrieval, the major downside was its very poor retrieval effectiveness. This lead to questioning the status of a word as a standalone unit of information, inspiring a family of techniques that use concepts instead of words as their building blocks. We have reviewed a cross section of such approaches in Chapter 2. While being effective in facilitating better retrieval, statistical concepts are often not easy to explain to humans. In this section we will retain words as our choice of Information Entities for building CRNs, but do away with the assumption that they are unrelated to each other. In other words, we use statistical approaches to model similarity between words, with the goal of improving retrieval effectiveness. Unlike concept learners that induce concepts which implicitly group related words together, our approach is to explicitly model word associations that can be alternately regarded as defining concepts over the feature space implicitly.

Typically statistical approaches model similarity between two words based on the number of documents where these words co-occur. Notwithstanding a significant amount of both philosophical and pragmatic debate on whether co-occurrence is a robust basis for semantic similarity (Jarmasz and Szpakowicz 2003), this simple approach works fairly well in the presence of large and representative collections (Terra and Clarke, 2003). Also, unlike domain-independent linguistic resources like WordNet or Roget's Thesaurus, this approach can be used for estimating domain specific word similarities. In this chapter, we show that we can do even better. We incorporate the notion of higher-order co-occurrence into our model of word similarity. The concept of higher order associations was introduced in Section 4.2.3, and is summarized here to make the discussion self-contained. The basic idea is to use indirect associations between words, in addition to direct ones. For example if words *car* and *chassis* co-occur in one document, and words *automobile* and *chassis* in another, we can infer that *car* and *automobile* are related to each other, even if they do not co-occur in any document. Such a relation is called a second-order association. We can extend this to orders higher than two. Several interesting examples showing the importance of second order associations have been reported in studies on large corpora. Lund and Burgess (1996) observe that near-synonyms like *road* and *street* fail to co-occur in their huge corpus. In a French corpus containing 24-million words from the daily newspaper *Le Monde* in 1999, Lemaire and Denhiere (2006) found 131 occurrences of *internet*, 94 occurrences of *web*, but no co-occurrences at all. However, both words are strongly associated. Their experiments show that higher order co-occurrences can be exploited to infer "semantic relatedness" (Budanitsky, 1999) between *road* and *street*, and between *web* and *internet*. Throughout this paper, we use the word "similarity" as a measure of semantic relatedness, as opposed to a formal semantic relation (like synonymy or hyponymy).

This chapter presents algorithms for mining higher order associations between words. The strengths of these associations are combined to yield an estimate of word similarity. In the next chapter, we empirically test the hypothesis that similarity knowledge mined using higher order co-occurrences leads to more effective retrieval in comparison to knowledge mined using first order co-occurrences alone.

As we have observed in Chapter 4, LSI can easily be adapted to the problem of learning similarity between features. This may make our goal of proposing a novel similarity mining approach appear superfluous. Later in this chapter we make a comparative study to illustrate the advantages of explicitly capturing higher order associations, as opposed to doing so implicitly as in LSI. In addition, we show how the similarity knowledge mining approach can be extended to incorporate class knowledge in supervised classification tasks. In Chapter 8, we show that a CRN using similarity knowledge based on higher order associations augmented with class knowledge can outperform state-of-the-art text classifiers like Support Vector Machines (SVM) and kNN based on LSI.

6.2 Higher Order Associations

The idea of higher order associations is illustrated through an example in Figure 6.1. Terms *A* and *B* co-occur in Document 1 in Figure 6.1(a), hence they are said to have a first order association between them. In Figure 6.1(b), terms *A* and *C* co-occur in one document, and terms *C* and *B* in another. In our terminology, *A* and *B* share a second order association between them, through *C*. Extending this idea to Figure 6.1(c), we say that *A* and *B* share a third order association between them through terms *C* and *D*. The first order paths that contribute to this third order association are (*A,C*), (*C,D*) and (*D,B*). The similarity between two terms *A* and *B* is a function of the different orders of association between them. This can be depicted as a graph as shown in Figure 6.1(d), where any two nodes sharing a first order co-occurrence relation between them are connected by an arc. Each higher order association between any two given nodes A and B is represented as a path connecting *A* and *B*. (*A,C,B*) is a second order path and (*A,C,D,B*) is a third order path. Until now, we have restricted our attention to the presence or absence of a first order path. A more general formulation of the similarity relations would also need to consider the strength of these first order associations. This is shown as the weighted graph in Figure 6.1(e). The weight of an arc connecting two nodes is proportional to the number of documents in the collection where they co-occur. The strength of the higher order associations is, in turn influenced by the first order association weights. In our implementations, we use the scheme in Figure 6.1(d).

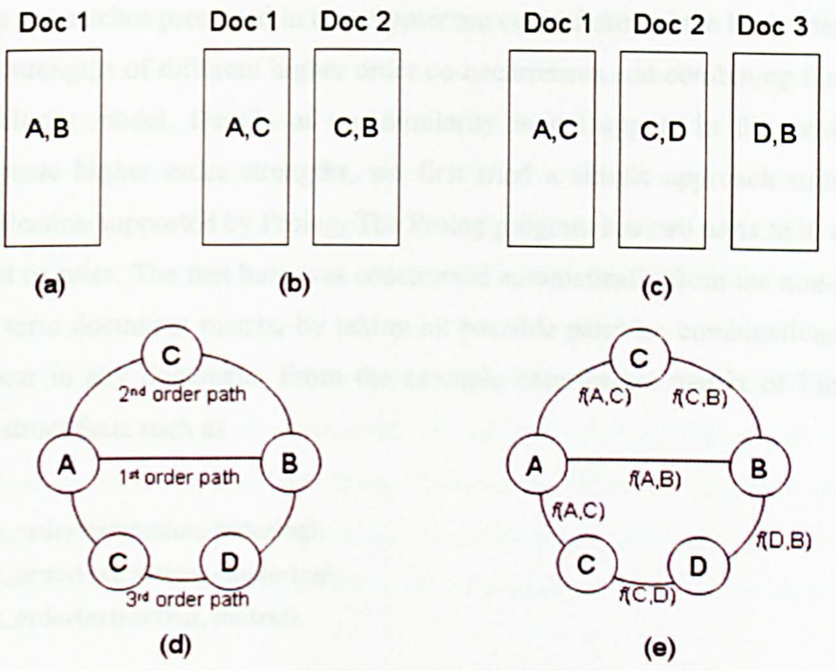
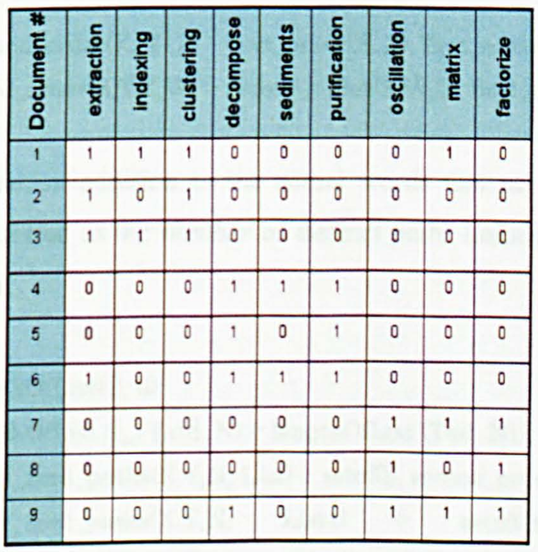


Figure 6.1 Graphical Representation of Higher Order Co-occurrences



Document #	extraction	indexing	clustering	decompose	sediments	purification	oscillation	matrix	factorize
1	1	1	1	0	0	0	0	1	0
2	1	0	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0
4	0	0	0	1	1	1	0	0	0
5	0	0	0	1	0	1	0	0	0
6	1	0	0	1	1	1	0	0	0
7	0	0	0	0	0	0	1	1	1
8	0	0	0	0	0	0	1	0	1
9	0	0	0	1	0	0	1	1	1

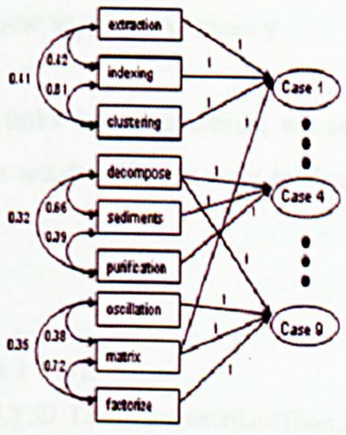


Figure 6.2 An Example Domain

The approaches presented in this chapter are centred around the basic idea of estimating the strengths of different higher order co-occurrences and combining them into a word similarity model. Details of our similarity model appear in the next section. To estimate higher order strengths, we first tried a simple approach using goal driven unification supported by Prolog. The Prolog program has two parts to it: a fact base and a set of rules. The fact base was constructed automatically from the non-zero entries of the term document matrix, by taking all possible pairwise combinations of terms that appear in any document. From the example case-feature matrix of Figure 6.2(a) we construct facts such as

```
first_order(extraction, indexing).
first_order(extraction, clustering).
first_order(extraction, matrix).
```

Defining rules for higher order association is straightforward using Prolog. Second and third order associations are defined in the following statements:

```
second_order(X, Y ,Z) :- first_order(X, Z), first_order(Z, Y), X \== Y.
third_order(X,Y,Z,W) :- second_order(X,W,Z), first_order(W,Y), X \== Y, Z\== Y.
```

Often, in addition to the actual words that act as links between words, we are also interested in the *number* of distinct paths linking up words. This is easy in Prolog, as well:

```
lengthOfList([], 0).
lengthOfList ([_|Tail], N) :- lengthOfList (Tail, N1), N is 1 + N1.
no_of_2ord_paths(X,Y,N, List) :- setof(Z, second_order(X,Y,Z), List), lengthOfList(List,N).
no_of_3ord_paths(X,Y,N,    List1)    :-    setof((K,L),    third_order(X,Y,K,L),    List1),
lengthOfList(List1,N).
```

One main limitation of Prolog in this task is the combinatorial explosion in the number of first order associations that had to be recorded in the fact-base. In realistic tasks over several hundreds of documents, our version of Prolog (SWI-Prolog) often ran out of memory. To address this limitation, we explored the applicability of matrix operations to directly compute the strengths of higher order associations. To start with, we implemented an approach reported by Mill and Kontostathis (2004), where the authors start by computing a first order co-occurrence matrix. For $|W|$ words in the feature set, this is a $|W| \times |W|$ matrix which has a value 1 in the *i,j*th element if word *i* co-occurs with word *j* in at least one document. For all pairs of words that do not co-occur in any document, the corresponding element in the matrix is 0. The diagonal values are set to zero since we are not interested in trivial co-occurrence of a word with itself. The higher-order co-occurrence matrices are calculated using the following steps:

Step 1: The term document matrix A is multiplied with its transpose A^T to obtain the $|W| \times |W|$ matrix T_0.
Step 2: All non-zero values of T_0 are set to 1, and the diagonal values are set to zero to yield a binary first order co-occurrence matrix T.
Step 3: The second order co-occurrence matrix T_2 can be calculated by squaring T. The third order matrix T_3 is given as T^3. Other higher order co-occurrence matrices can be calculated similarly.

Before a matrix is reduced to binary, the value of its *i,j*th element is the number of co-occurrence paths between words *i* and *j*. The strength of a first order co-occurrence path is the number of documents in which two words co-occur. The strength of a second order co-occurrence path between words *a* and *b* is the number of distinct words *c* such that *a* co-occurs with *c* and *b* co-occurs with *c*. In other words, the second order strength of an association is the number of distinct second order paths between the corresponding nodes in the graph model of Figure 6.1(d). Similarly, the strength of a third order association between words *a* and *b* is the number of distinct third order paths (via distinct nodes *c* and *d*) connecting *a* and *b*.

Implementing the above algorithm revealed a critical shortcoming. Let us consider a third order association between terms *a* and *b* via terms *c* and *d*. Thus pairs *a* and *c*, *c* and *d*, and *d* and *b* co-occur with each other. In finding distinct pairs of terms *c* and *d*, we need to ensure that they are not the same as either *a* or *b*. By setting the diagonal elements to 0 in Step 2 above, the algorithm ensures that *a* and *c* are different, and so are *d* and *b*. But in addition we also need to ensure that *d* is not the same as *a*, and *c* is not the same as *b*, and this is not taken care of. Thus the strengths of third order associations were over-estimated by the algorithm. We propose a correction to the algorithm to address this limitation. The brute force approach of explicitly counting terms that satisfy the above-mentioned constraint instead of blindly cubing the binary matrix *T*, turned out to be computationally expensive. We present below a technique that rewrites this procedure as an equivalent matrix manipulation, which can be implemented efficiently in matrix processing environments like Matlab.

Let *T* be the matrix of first order connections with diagonal elements set to zero. For third-order co-occurrences, we seek to enumerate paths of type *i-j-k-l* for all *i* and *l*.

Now

$$\left(T^3\right)_{il} = \sum_{j,k} T_{ij} T_{jk} T_{kl}$$

is the total number of such paths, including paths of type *i-j-i-l* and *i-l-k-l*, which we wish to exclude. Let n_i be the number of paths of type *i-j-i*. This is equal to the total number of paths originating from *i*. We evaluate n_i by summing the rows (or columns) of *T*:

$$n_i = \sum_j T_{ij}$$

Now, the number of paths of type *i-j-i-l* is $n_i T_{il}$ and for type *i-l-k-l* the count is $n_l T_{il}$. If $T_{il} \neq 0$, then we have counted the path *i-j-i-j* twice, so the total number of invalid paths is $(n_i+n_l-1)T_{il}$. Equivalently, if we construct a discount matrix D whose elements $D_{il} = (n_i+n_l-1)$, then the number of invalid paths between words i and j is given by the i,j th element of the pointwise product $D*T$. We use the following procedure:

(1) Calculate T^3.

(2) Enumerate and discount the invalid paths as above. $T^3 - D*T$ is the revised third order matrix.

6.2.1 An Example

We illustrate the above ideas on a toy casebase comprising 4 terms and 4 documents as shown in Figure 6.3. The corrected third order matrix T_3' says that there are two third-order paths between terms *t2* and *t3*, one third order path between *t1* and *t2*, another between terms *t1* and *t3*, and none between *t1* and *t4*. A closer inspection of matrix T reveals that this is indeed true. Figure 6.4 shows a graphical representation of matrix T, where an arc exists between any two nodes iff the corresponding entry in the matrix is 1, denoting that there is at least one document in the collection that has both of these terms. The two third order paths between *t2* and *t3* are *t2-t1-t4-t3* and *t2-t4-t1-t3*. The only third order path between *t1* and *t2* is *t1-t3-t4-t2*, and between *t1* and *t3* is *t1-t2-t4-t3*. There are only two possible candidates for a third order path between *t1* and *t4*: *t1-t2-t3-t4* and *t1-t3-t2-t4*. Either would require a first order association between *t2* and *t3*, which in our example does not exist, since there are no documents that contain both *t2* and *t3*. Hence any third order association between *t1* and *t4* is ruled out.

Now, let us take a closer look at how the discount matrix helps in identifying invalid paths. Matrix *T3* says that that there are 5 third order links between *t1* and *t2*. We enumerate them as follows: *t1-t4-t1-t2*, *t1-t3-t1-t2*, *t1-t2-t1-t2*, *t1-t3-t4-t2* and *t1-t2-t1-t2*. Interestingly, excepting *t1-t3-t4-t2* all other links are invalid associations of type *i-j-i-l* or *i-l-k-l*, and need to be discounted. Applying the formulations presented earlier, the number of *i-j-i-l* links in this case is $n_iT_{il} = n_1T_{12} = 3$. The number of *i-l-k-l* links is $n_lT_{il} = n_2T_{12} = 2$. Adding them up, we have 5 invalid links, but we note that we have counted the link *t1-t2-t1-t2* twice since it can be treated as an *i-j-i-l* link as well as an *i-l-k-l* link. Thus the actual number of invalid links is 4, and this is discounted from the exaggerated estimate of 5 in *T3* to yield the correct value 1 in matrix *T3'*.

$$A = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \longrightarrow T_0 = AA^T = \begin{pmatrix} 3 & 1 & 1 & 2 \\ 1 & 2 & 0 & 2 \\ 2 & 0 & 2 & 1 \\ 2 & 2 & 1 & 3 \end{pmatrix} \xrightarrow{\text{Convert non-zero values to 1 and diag. values to 0}} T = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \longrightarrow T_2 = T^2 = \begin{pmatrix} 3 & 1 & 1 & 2 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 1 & 3 \end{pmatrix}$$

Term Doc Matrix   Term Term Matrix   1st Order Matrix   2nd Order Matrix

$$T_3 = T^3 = \begin{pmatrix} 4 & 5 & 5 & 5 \\ 5 & 2 & 2 & 5 \\ 5 & 2 & 2 & 5 \\ 5 & 5 & 5 & 4 \end{pmatrix} \longrightarrow \text{Column Sums of T} \begin{matrix} n_1 = 3 \\ n_2 = 2 \\ n_3 = 2 \\ n_4 = 3 \end{matrix} \longrightarrow D(i,j) = n_i + n_j - 1 \begin{matrix} D(1,2) = 3+2-1 = 4 \\ D(1,3) = 3+2-1 = 4 \\ D(1,4) = 3+3-1 = 5 \\ D(2,3) = 2+2-1 = 3 \\ D(2,4) = 2+3-1 = 4 \\ D(3,4) = 2+3-1 = 4 \end{matrix} \longrightarrow D = \begin{pmatrix} 0 & 4 & 4 & 5 \\ 4 & 0 & 3 & 4 \\ 4 & 3 & 0 & 4 \\ 5 & 4 & 4 & 0 \end{pmatrix} \longrightarrow T_3' = T_3 - D*T = \begin{pmatrix} 4 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 0 & 1 & 1 & 4 \end{pmatrix}$$

3rd Order Matrix   Discount Matrix   Revised 3rd Order Matrix (* is pointwise product)

Figure 6.3 An Example

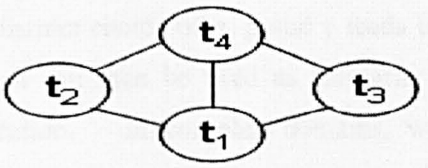


Figure 6.4 Term-Term Association Graph

6.3 Modelling Word Similarities

Once higher order co-occurrences are mined, we need to translate them into a measure of similarity between words. Intuition suggests that very high order co-occurrences do not really indicate similarity. In a study of higher order associations in the context of LSI (Kotostathis & Pottenger 2006), the authors present experimental evidence to confirm that associations beyond an order of 3 have a very weak influence on similarity modeled by LSI. In our word similarity model, we ignore the effects of orders higher than 3. In the last section, we have defined the strength of a higher order association between two terms as the number of co-occurrence paths between those terms. Let *first_order(a,b)*, *second_order(a,b)* and *third_order(a,b)* denote the strengths of first, second and third order associations between terms a and b respectively. The similarity between terms a and b can be expressed as a weighted linear combination of the strengths of the first three orders of co-occurrence as follows:

$$\text{similarity}(a,b) = \alpha \text{ first\_order}(a,b) + \beta \text{ second\_order}(a,b) + \gamma \text{ third\_order}(a,b) \qquad (6.1)$$

Note that the higher the order of association, the larger the number of co-occurrence paths (since $T^n_{i,j} > T^m_{i,j}$, if $n>m$ and if for all $T_{i,j} \neq 0$, $T_{i,j} \geq 1$, which is true in our case), and hence the greater the strength of association. Thus, to make α, β and γ comparable to each other, we need to normalize *first_order(a,b)*, *second_order(a,b)* and *third_order(a,b)* to values in [0,1]. In our implementation, we achieve this by dividing each of these values by the maximum value between any pair of words corresponding to that order. Each distinct choice of α, β and γ leads to a different set of similarities between terms, which can then be used as similarity arcs in the CRN to perform retrieval or classification. In complex domains, we would expect higher order associations to play a critical role and hence such domains should show preference for higher values of β and γ compared to simpler ones.

6.4 Learning Model Parameters Automatically

Next, we dwell on the problem of determining the value of the weights α, β and γ automatically. We present an approach that uses a Genetic Algorithm (Russell & Norvig 2003) to determine these parameters in supervised classification tasks. Since the test set is not available, we instead set our objective to optimizing classification accuracy over five-fold cross validation on the training set. The approach is broadly illustrated in Figure 6.5.

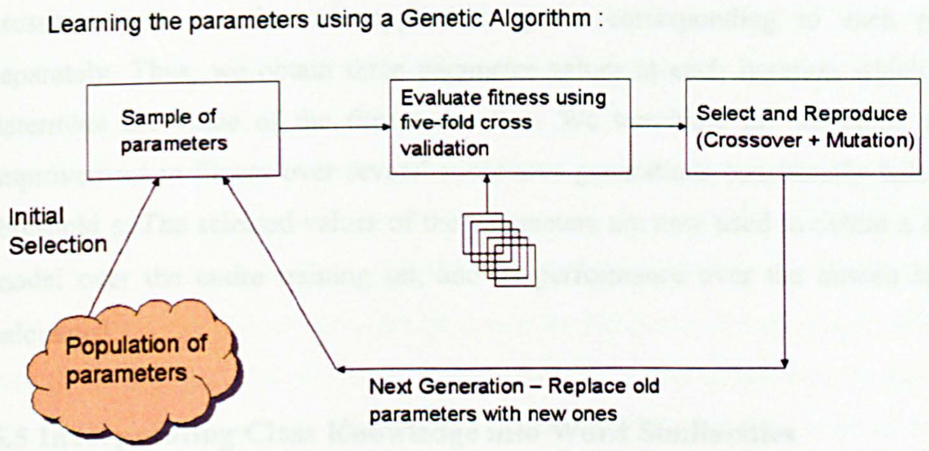

Learning the parameters using a Genetic Algorithm :
Sample of parameters
Evaluate fitness using five-fold cross validation
Select and Reproduce (Crossover + Mutation)
Initial Selection
Population of parameters
Next Generation – Replace old parameters with new ones

Figure 6.5 Parameter Learning using a Genetic Algorithm

To start with, we encode the parameters α, β and γ, which form the solution domain, into a genetic representation. We use a 9 bit string encoding to represent each parameter. To encode a value of 0.8, we multiply this by 100 and convert the resulting value (80) to its binary equivalent 001010000. This representation allows us to represent real valued numbers from 0 to 5.11. We set α to a value of 1, and find values of β and γ to yield the best combination. The algorithm starts off with an initial population of values for the parameters. A sample is randomly drawn from this population and the fitness of the parameters is evaluated. This is done by splitting the training set into 5 equal disjoint subsets. A cross validation is procedure is then run 5 times, each time selecting a different one of these as the validation (or test) set and combining the remaining 4 subsets for the training set. For each of these cross validation procedures, the accuracy of classification on the validation set is calculated, using the similarity model derived from the chosen parameters. The average of the five accuracy values is the quantity we intend to maximize, and is referred to as the "fitness function". The fittest combinations of parameters are chosen to obtain the next generation. A single point crossover over each of the fittest genes is used to obtain a new set of parameters. We also perform mutation (random flipping of bits) to improve diversity in the next generation; the mutation probability is set to 1/16. Note that crossover and mutation are applied to genes corresponding to each parameter separately. Thus, we obtain three parameter values at each iteration which together determine the value of the fitness function. We terminate the iterations when the improvement in fitness over several successive generations consistently falls below a threshold ε. The selected values of the parameters are now used to obtain a similarity model over the entire training set, and its performance over the unseen test set is calculated.

6.5 Incorporating Class Knowledge into Word Similarities

In a supervised classification context, we have class knowledge of training documents in addition to the co-occurrence knowledge. This is ignored in our similarity formulation in (1). However, class knowledge can play an important role in boosting similarity of features if they occur frequently in cases belonging to the same class, and in demoting similarity if they occur in cases belonging to disjoint classes. In this section, our goal is to incorporate this class knowledge as part of pre-processing. The approach is based on the idea of 'sprinkling' described in Chapter 5, where LSI was extended to supervised classification tasks.

Each case in the training set is padded with additional artificial features that are representative of class knowledge. For example in the Hardware domain, all cases belonging to Apple Mac are augmented with artificial terms A, B , C and D, and all documents belonging to PC are padded with terms E, F, G and H. The padded terms, are referred to as *sprinkled* terms. It is interesting to note how these additional terms influence the co-occurrence paths between any given pair of original features. When co-occurrences are mined on the new case-feature representation having these additional terms, features representative of the same class are drawn closer to each other, and features from disjoint classes are drawn farther apart. This happens because the sprinkled terms provide second-order co-occurrence paths between features of the same class. Thus using the similarity formulation in (6.1) over the combination of existing feature set and artificial terms results in increased similarity between features of the same class. The revised similarity values are used to construct a CRN; it is important to note that the sprinkled terms only play a role in computing similarities, but do not appear as IE nodes in the CRN. Thus, an incoming test document whose class is unknown can be processed in the usual manner. Since similarities between the original features as captured by the similarity arcs is already biased by class knowledge, retrieval over the revised architecture would be influenced by the class affiliations of the query features. Most design issues pertinent to our discussion of sprinkling in Section 5.2 also apply to our current architecture. For one, we need to decide the number of additional terms to be added for each class. While sprinkled terms help in emphasizing class knowledge, using too many of them may distort finer word association patterns in the original data. This relates to the structure vs. class knowledge dilemma outlined in Section 4.1. The approaches and heuristics outlined in Section 5.2 to guide the selection of number of sprinkled terms are relevant here. In our experiments reported in the next chapter, we used 8 additional terms per class, as this was empirically found to yield good results.

6.6 Examples of Higher Order Associations

Figure 6.6 shows examples of second and third order associations mined from the RELPOL domain. The weighted graph representation of Figure 6.1(e) is used for these illustrations. The number alongside each path is the product of first order co-occurrences constituting that path. It is important to note that while these numbers have been used to order the paths in the graph visualization, they have no bearing on the strengths of higher order paths as computed using the approaches in Section 6.2.

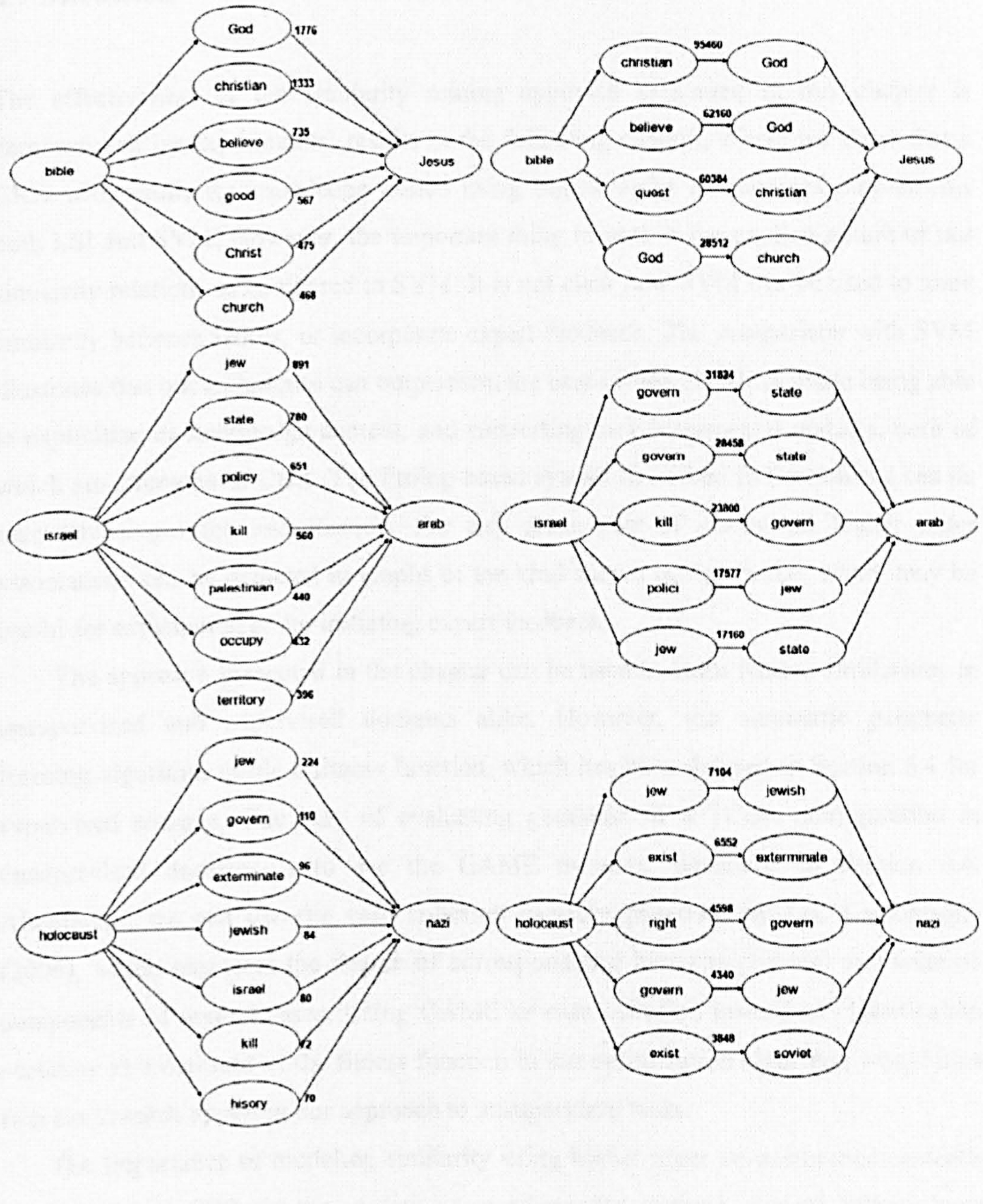


Figure 6.6 Examples of associations mined in RELPOL domain

6.7 Discussion

The effectiveness of the similarity mining approach presented in this chapter is demonstrated by experimental results in the following chapter, where we show that a CRN with similarity knowledge mined using higher order associations outperforms both LSI and SVM. However, the important thing to note is the explicit nature of our similarity relations as compared to SVM. It is not clear how SVM can be used to mine similarity between words, or incorporate expert feedback. The comparison with SVM illustrates that our techniques can outperform the best-in-line classifier while being able to explicitize its knowledge content, and supporting lazy incremental updates, both of which are strengths of CBR. The Prolog-based system described in Section 6.2 has its own advantages for visualization. For any given pair of words, all higher order associations can be depicted in graphs of the kind shown in Figure 6.6, which may be useful for explanation or for initiating expert feedback.

The approach presented in the chapter can be used to learn feature similarities in unsupervised and supervised domains alike. However, the automatic parameter learning algorithm needs a fitness function, which has been defined in Section 6.4 for supervised settings. One way of evaluating goodness of a TCBR configuration in unsupervised domains is to use the GAME measure, described in Section 3.4. Alternately, we can use the case cohesion measure proposed by Luc Lamontagne (2006), which measures the degree of correspondence between problem and solution components of textual cases. Using GAME or case cohesion instead of classification accuracy as a measure of the fitness function in our optimization algorithm would be a first cut towards applying our approach to unsupervised tasks.

The importance of modeling similarity using higher order co-occurrences extends beyond textual CBR. In the context of recommender systems, several authors have reported problems due to sparseness of user-item matrices (Xue et al., 2005); Semeraro et al (2005) for example, report that 87% of the entries in their user-item matrix are zero. Knowledge representations used in collaborative recommenders (like concept lattices (du Boucher-Ryan & Bridge, 2006)) do not consider associations beyond the first order. Higher order associations can help reduce the sparseness and allow for better recommendation. In this context, analysis of higher-order associations in user item matrices will help discover novel product recommendation rules that would normally be implicit in the user ratings. Our approach can also be applied to link analysis in social networks (Mori et al., 2007), for clustering similar words, and resolving ambiguity of words spanning several clusters.

Several works in the past have pointed to the importance of higher order co-occurrence in modeling word similarity. However we have not come across any work that explicitly attempts to obtain a parameterized model of similarity based on these co-occurrences, and learn optimal values of these parameters based on a fitness criterion. As noted in Section 4.2.3, the work by Kontostathis and Pottenger (2006) provides empirical evidence to show that LSI implicitly exploits higher order co-occurrence paths between words to arrive at its revised representations. This provides a fresh explanation for improvements obtained using LSI in text retrieval applications. Edmonds (1997) examines the role of higher order co-occurrence in addressing the problem of lexical choice, which is important to both machine translation and natural language generation. Broadly speaking, the goal is to determine which of the possible synonyms is most appropriate for a given communication (or pragmatic) goal. The authors show that using second order co-occurrence has a favourable influence on the performance of their lexical choice program. Recent work by Lemaire and Denhiere (2006) makes an in-depth study of the relationship between similarity and co-occurrence in a huge corpus of children's texts. They show that while semantic similarity is largely associated with first order co-occurrence, the latter overestimates the former. Higher order co-occurrences as well as lone occurrences (occurrence of word *a* but not *b* and vice versa) were used to account for LSI-inferred term similarities. Unlike our work, the authors do not propose an algorithm to arrive at word similarities; their approach is more analytic than synthetic. Two other recent approaches potentially useful for mining word similarities are distributional word clustering (Baker & McCallum, 1998) which has been used recently in TCBR (Wiratunga et al., 2006), and Propositional Semantic Indexing (Wiratunga et al., 2005a) which mines word relationships using Association Rule Mining (ARM) with the goal of feature generalization. However, probability estimates used in the first approach and the support and confidence estimates used in the second currently fail to accommodate associations beyond the first order. It appears that both approaches can potentially benefit from higher-order knowledge.

6.8 Chapter Summary

We have presented an approach for exploiting higher-order associations between words to acquire similarity knowledge for CRNs. We highlighted the importance of higher order co-occurrences in determining word similarity, presented both supervised and unsupervised algorithms for mining such associations and proposed a word similarity model, whose parameters are learnt using an evolutionary approach. Another contribution of the research is to incorporate class knowledge into the process of mining higher order associations. Though the work has been presented in the context of CRNs, it can be easily extrapolated to learn similarity knowledge over other retrieval formalisms. In the next chapter, we will present empirical evaluation of our approaches, and make a comparative assessment of its performance versus state-of-the-art approaches like SVM and LSI.

Chapter 7

Evaluation

Who cares how it works, just as long as it gives the right answer Jeff Scholnik

In this chapter, we report and analyse results of experimental evaluation of the approaches presented in Chapters 5 and 6 for mining relevance and similarity knowledge in CRNs. The main objective of the experiments is to establish whether the acquired relevance and similarity knowledge leads to improvement in the retrieval effectiveness in CRNs. We also illustrate the goodness of knowledge acquired, in ways that involve the qualitative judgement of the reader, instead of statistical rigour.

For most of our evaluation, we use text classification datasets, and use classification accuracy as a measure of retrieval effectiveness. Using classification datasets is a natural choice when evaluating sprinkled LSI, which is engineered specifically for supervised tasks. However, this is not so for evaluating the algorithm for mining similarity knowledge based on higher order associations, which can be applied to unsupervised tasks as well. However, in light of the difficulty in obtaining human relevance judgements over unsupervised datasets, we have opted to use classification datasets throughout. A note about use of classification accuracy as an evaluation metric is in order. While researchers in text classification have used more involved metrics based on F-measure or ROC, it was noted in (Gabrilovich and Marcovitch 2004) that accuracy is an appropriate and adequate measure in situations where each case (document) belongs to not more than one class, and the class distribution is uniform. This makes accuracy a suitable measure of effectiveness in our experiments.

While carrying out empirical studies, it is useful to characterize the datasets used, so that the results of experimental studies can be meaningfully interpreted with respect to those characterizations. In chapter 3, we introduced six text classification datasets, and presented techniques to visualize these datasets and obtain quantitative estimates of their complexity. We use these datasets in our experiments here, and study the correlations of our accuracy results with the complexity measures, with the objective of obtaining better insights into successes or failures on specific datasets.

The chapter is organised as follows. Section 7.1 presents an evaluation of the effectiveness of LSI-mined relevance knowledge. Section 7.2 evaluates sprinkling, and also shows how word similarities are affected by the process of incorporating class specific knowledge. Adaptive Sprinkling is evaluated in Section 7.3. AS exploits inter-class relationships, hence additional datasets that involve hierarchical and ordinal associations between classes are used. Section 7.4 discussed performances obtained using the similarity knowledge mined using higher order associations. We summarize our findings in Section 7.5.

7.1 LSI performance

In this section, we evaluate the effectiveness of relevance knowledge mined using LSI. We compare these values against a baseline Vector Space approach, called BASE, whose relevance values are simply binary values showing feature presence or absence in a case. We use a weighted *k*NN with *k* =3.

Firstly, we note that LSI performance is critically dependent on the number of dimensions chosen for creating the revised representations. Figure 7.1 shows the performance of LSI, in terms of classification accuracy, over each of the six datasets at different LSI dimensions. The LSI dimensions used in our experiments were 5, 10, 15, 20, 40, 60, 80, 100 and 120. All performance figures are obtained after averaging over 15 test train splits of the datasets described in Section 3.1. The broad pattern that is observed over all datasets is very clear: LSI performance is best when a small number of dimensions ranging from 5 to 20 is used. There is a pronounced peaking behaviour in both four-class problems REC and SCIENCE. This is possibly because these four-class problems have more diversity in their texts in comparison to the relatively simpler spam filtering problems USREMAIL and LINGSPAM. Our complexity analysis and visualization in Chapter 3 shows HARDWARE to be the most complex of the binary problems. Interestingly, the number of LSI dimensions that yield best performance in HARDWARE is 20, which is conspicuously larger than the best dimensionality in the simpler binary problems like RELPOL or the spam filtering datasets.

Table 7.1 Comparing LSI performance against naive VSM

	Routing				**Filtering**	
	REC	**SCIENCE**	**HARDWARE**	**RELPOL**	**USREMAIL**	**LINGSPAM**
BASE	62.79	54.89	59.51	70.51	59.23	85.09
LSI	**79.32**	**72.55**	**66.30**	**91.17**	**94.67**	**97.37**

Table 7.1 shows the peak accuracy results of LSI for each of the six datasets, observed at the dimensionality that yielded best performances. We observe that there is a statistically significant improvement over the baseline in all datasets. The margin of improvement is quite large in all binary datasets excepting HARDWARE. The relatively unimpressive gain in HARDWARE can be explained by correlating the accuracy results with $GAME_{class}$ scores reported in Section . The $GAME_{class}$ scores and LSI accuracies are compared in Table 7.2. We observe a correlation coefficient of

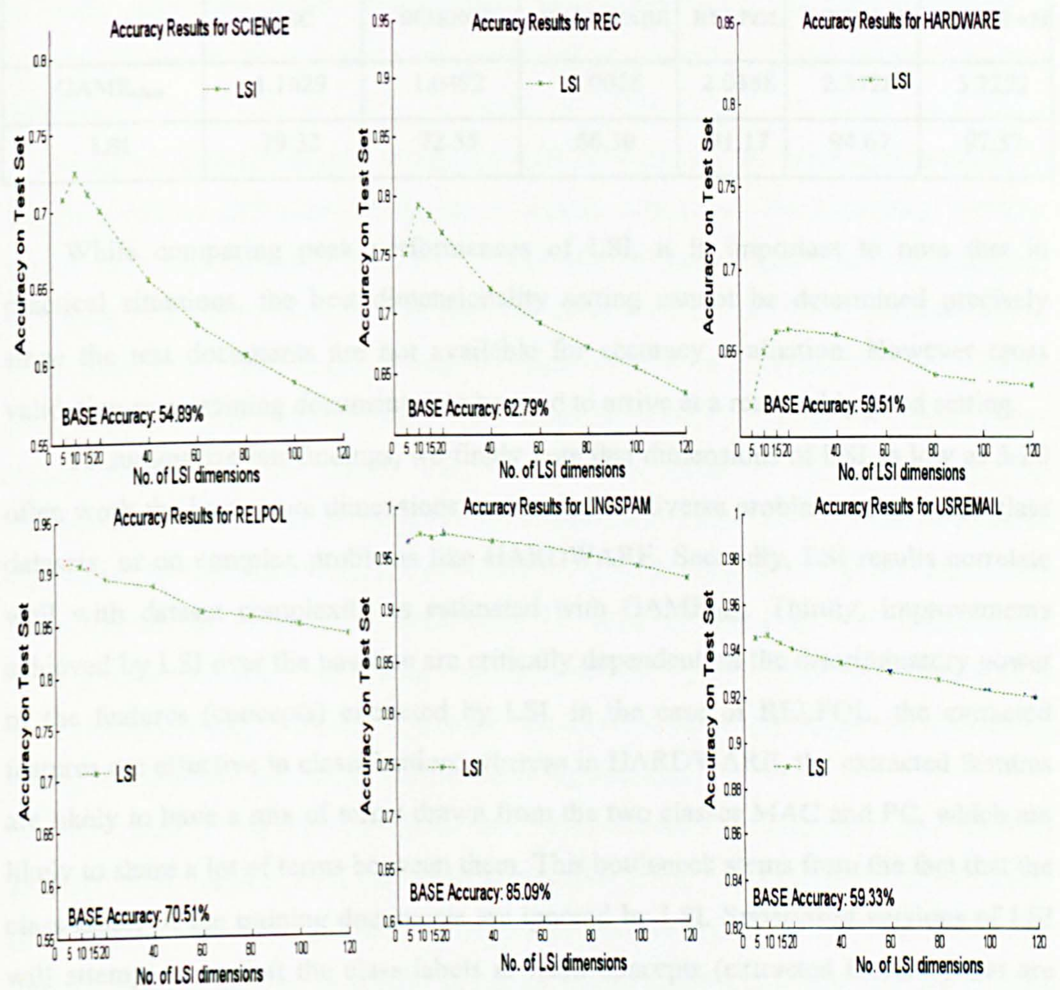


Figure 7.1 LSI performance at various dimensions

0.9176, over the four binary class problems. The high correlation suggests that HARDWARE's inherent complexity, coupled with the fact that LSI fails to take into account class knowledge of the training documents, account for the poor performance. Comparing LSI performances over the four-class datasets show that SCIENCE registers a better performance than REC, and this corresponds well with the complexity scores registered by GAME$_{class}$.

Table 7.2 Comparing $\text{GAME}_{\text{class}}$ with LSI accuracies

	REC	SCIENCE	HARDWARE	RELPOL	USREMAIL	LINGSPAM
$\text{GAME}_{\text{class}}$	1.1629	1.0492	1.0028	2.0358	2.3728	3.2222
LSI	79.32	72.55	66.30	91.17	94.67	97.37

While comparing peak performances of LSI, it is important to note that in practical situations, the best dimensionality setting cannot be determined precisely since the test documents are not available for accuracy evaluation. However cross validation over training documents can be used to arrive at a reasonably good setting.

To summarize our findings, we firstly note that dimensions of LSI as low as 5-20 often work the best; more dimensions are needed on diverse problems as in multi-class datasets, or on complex problems like HARDWARE. Secondly, LSI results correlate well with dataset complexity as estimated with $\text{GAME}_{\text{class}}$. Thirdly, improvements achieved by LSI over the baseline are critically dependent on the discriminatory power of the features (concepts) extracted by LSI. In the case of RELPOL, the extracted features are effective in classification; whereas in HARDWARE, the extracted features are likely to have a mix of terms drawn from the two classes MAC and PC, which are likely to share a lot of terms between them. This bottleneck stems from the fact that the class labels of the training documents are ignored by LSI. Supervised versions of LSI will attempt to exploit the class labels to learn concepts (extracted features) that are better at discriminating between classes.

7.2 Sprinkling

We evaluated the effectiveness of sprinkled LSI representations over the six classification datasets described in Section 3.1. Ideally, the optimal number of sprinkled terms to be used should be determined based on cross-validation carried out over training dataset for each individual trial. However, to simplify evaluation, we used

16 sprinkled terms throughout for our experiments, as this was empirically determined to yield reasonably good performances. Thus, each class was represented by 8 sprinkled terms in the binary classification problems, and by 4 terms in the four-class problems.

Figure 7.2 shows graphs comparing LSI performance, in terms of classification accuracy, before and after sprinkling, at various choices of dimensionality. As with experiments reported in the previous section, the accuracy Figures are obtained after averaging over the 15 trials in each dataset. Firstly, we note that all six datasets benefit from sprinkling. The margin of improvement obtained is also large in all datasets, excepting USREMAIL and LINGSPAM, which are simple domains with already high classification accuracies. Thirdly, the performance of sprinkled LSI peaks at very low dimensions in the range of 5-10. In this respect, the difference with LSI is notable, especially for the three complex datasets, SCIENCE, HARDWARE and REC. The incorporation of the sprinkled terms helps in "aligning" the extracted dimensions to class specific features. Thus the top few dimensions in sprinkled LSI would capture the most discriminating features; in unsupervised LSI however, a larger dimensionality would be needed to capture these discriminating features.

In Table 7.3, we report the peak performance results for sprinkled LSI over each dataset. We compare these performances against BASE (our earlier VSM baseline), peak LSI performances as reported in the last section and Support Vector Machines (SVMs). SVM has been chosen for our comparisons since it has been reported to yield state-of-the-art performances in text classification, in several independent studies. For our SVM experiments we used SVM-Light (Joachims 1998). We used a linear kernel since it was observed to be well suited to textual problems (Joachims 1998). SVM being inherently a binary classification tool, we have not reported SVM results corresponding to the two multi-class problems. However we plan to use multi-class extensions to SVM in future over these datasets. We used paired t-tests ($p = 0.05$) to evaluate the significantly better of each 6 pairs originating from the four classification problems. The paired data for the t-tests comes from the 15 pairs of observations from the 15 train test splits. Results in bold correspond to best accuracy ; in situations where there is no significant statistical difference between the best results, all top figures are shown in bold. Firstly, we note that sprinkled LSI outperforms LSI on all six domains. With the exception of the simple LINGSPAM domain where classification accuracies were already quite high, the differences with and without sprinkling are statistically significant throughout. This conclusively suggests that class knowledge plays a critical role in arriving at better case representations, leading to higher classification effectiveness. Secondly, sprinkled LSI significantly outperforms SVM in three of the four binary classification problems. In USREmail, a relatively simple domain with high classification accuracies, Sprinkled LSI accuracy is higher, though the difference is not statistically significant.

Table 7.4 compares peak accuracies of sprinkled LSI against $\text{GAME}_{\text{class}}$ scores. Sprinkled LSI performance correlates well with the $\text{GAME}_{\text{class}}$ complexity measures. For the binary classification problems, the correlation coefficient is 0.9365, which is higher than what was observed with unsupervised LSI (0.9176). Performance improvement in the complex HARDWARE domain is conspicuous, thus confirming our earlier hypothesis that class knowledge is critical here.

In Table 3.1 of Chapter 3, we presented the accuracy figures of three other classifiers, namely PSI, ECRN and LogitBoost. We do not replicate those results here, but it is clear that sprinkled LSI outperforms these classifiers over most datasets, and compares favourably in the rest. We also note that sprinkled LSI outperforms the very recently proposed TCBR technique PSI on all six datasets; the margins of difference in the multi-class problems REC and SCIENCE are especially conspicuous.

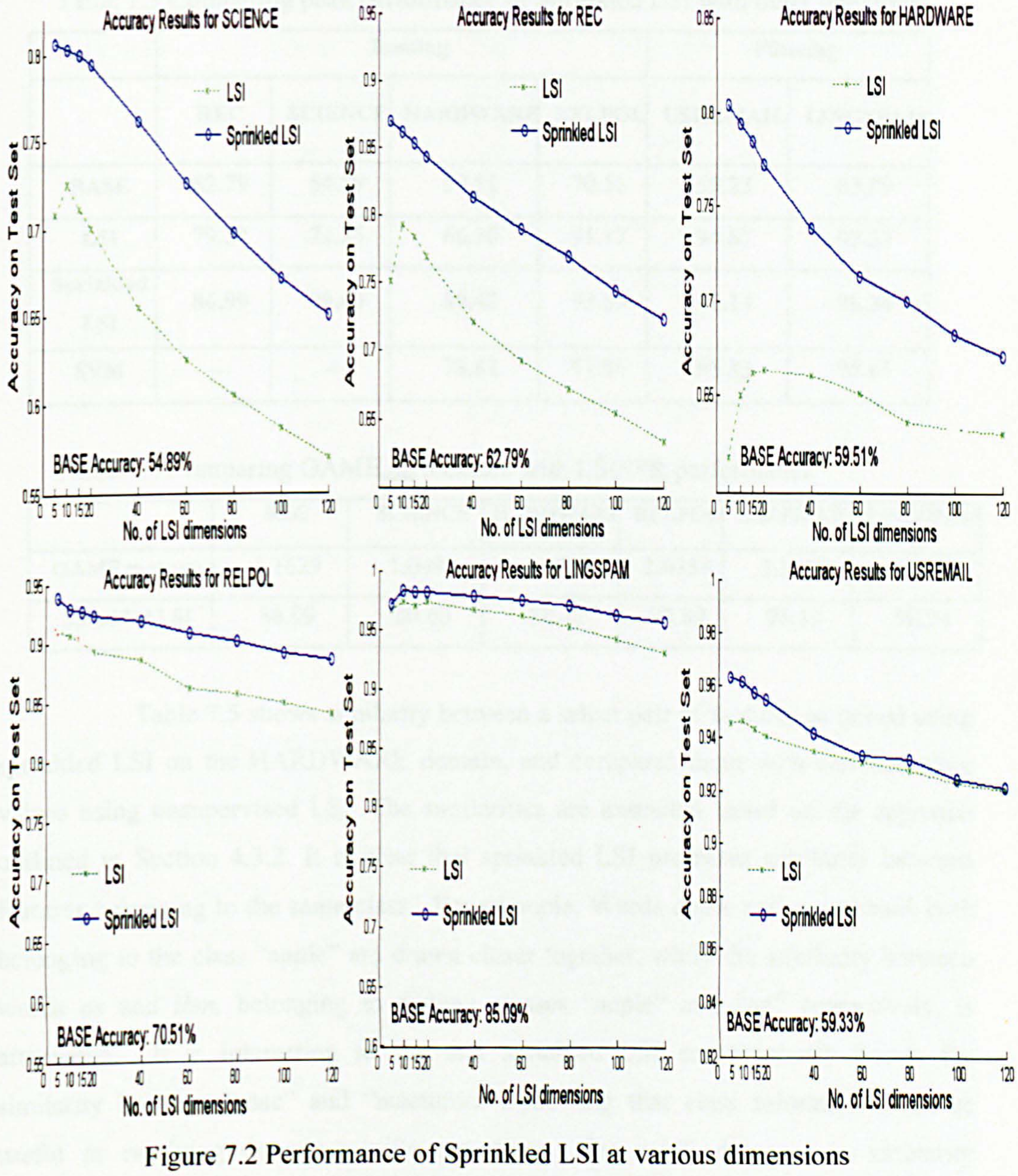


Figure 7.2 Performance of Sprinkled LSI at various dimensions

Table 7.3 Comparing peak performance of Sprinkled LSI with other classifiers

	Routing				Filtering	
	REC	**SCIENCE**	**HARDWARE**	**RELPOL**	**USREMAIL**	**LINGSPAM**
BASE	62.79	54.89	59.51	70.51	59.23	85.09
LSI	79.32	72.55	66.30	91.17	94.67	**97.37**
Sprinkled LSI	**86.99**	**80.60**	**80.42**	**93.89**	**96.13**	**98.34**
SVM	---	---	78.82	91.86	**95.83**	95.63

Table 7.4 Comparing GAME$_{class}$ measure with LSISPR performance

	REC	SCIENCE	HARDWARE	RELPOL	USREMAIL	LINGSPAM
GAME measure	1.1629	1.0492	1.0028	2.0358	2.3728	3.2222
Sprinkled LSI	86.99	80.60	80.42	93.89	96.13	98.34

Table 7.5 shows similarity between a select pair of features as mined using sprinkled LSI on the HARDWARE domain, and compares them with corresponding values using unsupervised LSI. The similarities are extracted based on the approach outlined in Section 4.3.2. It is clear that sprinkled LSI promotes similarity between features belonging to the same class. For example, Words *apple* and *powerbook* both belonging to the class "apple" are drawn closer together, while the similarity between words *os* and *ibm,* belonging to distinct classes "apple" and "pc" respectively, is attenuated. It is interesting to see that sprinkled LSI conspicuously boosts the similarity between "mac" and "macintosh", showing that class information can be useful in resolving domain-specific synonymy. The small decrease in similarity between *apple* and *mac* appears unintuitive, but is possibly a side effect of the LSI-based constrained optimization which favours marginally weakening existing strong bonds, with the intent of reinforcing otherwise weak associations.

Table 7.5 Word Similarities before and after Sprinkling

Word 1	**Word 2**	**LSI**	**LSISPR**	**Difference**
Mac	*powerbook*	3.26	4.61	1.35
Mac	*macintosh*	1.88	3.25	1.37
Appl	*macintosh*	2.00	2.81	0.81
Appl	*powerbook*	3.77	4.12	0.35
Appl	*mac*	19.94	19.02	-0.72
powerbook	*ibm*	0.17	0.09	-0.08
macintosh	*ibm*	0.09	0.23	0.14
Pc	*macintosh*	2.44	1.99	-0.45
powerbook	*macintosh*	0.71	1.20	0.49
Os	*microsoft*	2.27	1.80	-0.46
Os	*macintosh*	2.13	2.34	0.21
Os	*powerpc*	0.15	0.58	0.44
Os	*ibm*	2.82	2.31	-0.51

It is illustrative to compare the association of different terms to classes, before and after sprinkling. Considering the two case-feature matrices corresponding to lower rank approximations generated by LSI and sprinkled LSI, we can obtain a simple measure of prototypicality ("representativeness") of a term to class by adding up relevance values that estimate association of that term to cases belonging to that class. Assuming equal number of cases in each class, the resulting class memberships can be meaningfully compared against each other. Table 7.6 shows the prototypicality of words to each of the four classes Electronics, Cryptography, Space and Medicine in the SCIENCE domain. The figures in bold show the class to which the word is most likely to be assigned by a human, based on knowledge of the domain. We can see clearly that in all cases, Sprinkled LSI changes class memberships to make them better representative of the concepts underlying each class. The words *radio* and *antenna* are relevant to two classes Electronics and Space, and Sprinkled LSI correctly boosts the corresponding memberships. Thus, sprinkling exploits class knowledge effectively to make the LSI mined representations approximate the expert knowledge of the domain more closely.

Table 7.6 Term affiliations to classes in SCIENCE domain before and after sprinkling

	LSI				Sprinkled LSI			
	Electronics	**Cryptography**	**Space**	**Medicine**	**Electronics**	**Cryptography**	**Space**	**Medicine**
encrypt	7.02	**95.29**	-0.73	-4.09	0.08	**110.59**	-0.97	-2.25
secur	3.28	**65.55**	1.23	-1.29	0.32	**68.76**	-0.66	-0.70
chip	**23.30**	70.41	3.56	4.59	**29.81**	80.17	-0.45	4.15
orbit	6.08	0.73	**35.76**	2.93	-0.28	0.28	**52.95**	0.17
space	7.75	11.42	**59.54**	11.38	5.55	10.34	**78.62**	6.22
algorithm	-1.46	**44.81**	0.34	2.91	0.66	**49.06**	0.49	0.79
launch	4.65	0.71	**32.81**	-0.10	-0.35	1.94	**44.40**	-1.12
crypto	5.81	**37.05**	2.94	-2.07	0.33	**41.71**	0.29	-0.38
privaci	5.00	**36.23**	3.03	-0.18	2.19	**40.41**	-0.78	0.64
medic	6.65	-0.03	3.05	**23.61**	-0.18	-0.27	-0.59	**36.25**
circuit	**21.01**	2.01	1.66	0.40	**36.80**	-0.52	-1.39	-2.02
doctor	4.69	-0.49	0.85	**26.03**	0.26	-0.89	-0.52	**34.83**
earth	5.81	0.57	**26.51**	6.01	5.53	-1.56	**40.31**	1.15
moon	4.49	0.91	**33.95**	3.87	2.33	1.28	**40.32**	0.76
nasa	0.03	3.33	**34.43**	5.70	0.82	2.71	**40.13**	4.01
rocket	3.65	0.57	**24.43**	-0.51	-0.06	0.37	**30.49**	-0.17
voltag	**19.55**	1.29	0.93	1.07	**29.70**	-1.30	-1.12	-0.49
public	2.29	**51.61**	12.68	18.46	0.59	**53.55**	19.16	16.03
diseas	1.44	1.50	4.71	**20.08**	-0.21	0.86	0.91	**28.28**
shuttl	-0.05	0.67	**12.90**	2.40	-0.38	-0.45	**21.52**	0.02
rsa	0.71	**22.37**	-0.62	1.17	-0.31	**22.92**	0.36	-0.37
cryptograph	-0.12	**21.62**	0.70	1.04	0.76	**21.71**	0.37	-0.07
health	6.00	1.11	3.84	**18.95**	-0.06	0.62	0.85	**25.71**
medicin	0.68	0.48	2.29	20.48	1.90	-1.07	1.02	26.24
flight	2.00	0.85	**16.42**	5.51	0.44	0.12	**23.38**	4.91
treatment	0.68	-1.15	0.30	**17.54**	-0.79	-1.44	-0.93	**20.80**
lunar	2.16	0.17	**11.67**	1.42	0.03	-0.24	**19.22**	-0.16
electron	**13.29**	22.15	3.69	0.35	**25.38**	17.83	1.00	0.82
patient	-0.42	0.19	1.10	**10.19**	-0.57	-0.17	0.07	**14.72**
spacecraft	0.41	-0.28	**6.84**	1.14	0.22	-0.47	**13.08**	0.01
diet	1.33	0.17	3.98	**15.31**	0.59	0.48	1.28	**18.72**
drug	1.18	11.53	-0.69	**10.09**	-1.14	12.11	-0.47	**11.87**
volt	**9.29**	-0.31	2.28	0.35	**13.86**	-1.13	0.48	-0.79
tv	**6.39**	1.11	1.95	2.37	**13.12**	0.19	1.38	0.48
ohm	**6.72**	1.21	0.34	0.16	**11.57**	0.17	-0.21	-0.35
payload	2.57	-0.16	**11.11**	-1.01	0.24	0.11	**12.02**	-0.70
aerospac	2.03	0.46	**10.01**	-0.44	0.01	0.34	**11.91**	-0.57

	LSI				Sprinkled LSI			
	Electronics	Cryptography	Space	Medicine	Electronics	Cryptography	Space	Medicine
satellit	1.41	1.49	**10.87**	1.12	2.99	1.04	**15.75**	-0.02
propuls	0.09	-0.70	**10.42**	2.47	0.98	-0.11	**13.01**	0.58
atmospher	1.37	-0.29	**7.70**	1.78	1.20	-0.27	**12.67**	0.47
apollo	0.58	0.97	**7.90**	0.16	-0.41	0.30	**12.27**	0.28
symptom	1.72	-1.32	0.79	**10.81**	1.09	-0.51	-0.32	**13.18**
motorola	**7.18**	2.74	1.37	2.16	**15.00**	1.55	1.75	1.27
homeopathi	-0.39	-0.56	0.31	**9.03**	0.20	-0.61	-1.01	**9.75**
surgeri	1.07	-0.27	0.73	**5.21**	-0.51	-0.05	-0.16	**8.80**
altitud	-0.18	-0.03	**6.81**	-0.05	-0.32	-0.19	**8.69**	-0.26
microcircuit	**-0.37**	7.29	-0.94	-0.55	**0.18**	7.33	-0.83	-0.04
planet	0.70	0.10	**3.53**	1.60	0.11	0.04	**8.78**	1.07
password	1.92	**5.01**	0.39	-0.17	0.15	**6.65**	0.85	-0.45
physician	0.29	0.30	1.13	**2.79**	-0.13	0.08	0.88	**6.46**
unix	6.35	**6.57**	2.10	3.47	1.28	**10.00**	0.40	3.34
astronomi	0.23	0.35	**2.83**	3.15	-0.04	0.09	**6.21**	4.05
radio	**12.75**	4.86	**10.08**	4.31	**19.55**	4.55	**12.03**	3.79
associ	1.78	6.52	6.91	12.87	1.44	4.37	11.23	14.35
clinic	1.34	-0.65	0.21	**5.77**	1.61	-0.41	-0.43	**6.48**
planetari	-0.16	2.39	**3.31**	1.19	-0.54	2.14	**6.11**	0.02
antenna	**1.84**	0.05	**4.48**	0.80	**4.13**	-0.10	**5.42**	0.13
chang	9.85	23.64	15.34	18.66	7.94	24.59	11.88	19.44
telescop	2.13	0.49	**7.04**	1.84	1.85	0.54	**7.47**	2.00
cancer	0.66	1.43	0.47	**6.63**	-0.25	1.67	0.87	**7.10**
fever	-0.13	0.11	0.96	**2.65**	-0.16	0.01	2.57	**4.06**

We now briefly turn our attention from evaluating effectiveness of mined knowledge to examining efficiency implications of sprinkling. The only overhead associated with sprinkling is that the size of the case-feature matrix grows because of the augmented sprinkled columns. Table 7.7 shows that this overhead is miniscule in practical scenarios. For our experiment, we compared time taken for computing the SVD of a case-feature matrix having 1000 cases and 1000 features, before and after sprinkling. We use the Matlab implementation of SVD over a PC configured with Pentium 4 (single-core) processor and 512 MB of RAM, for our experiments.

Table 7.7 Time performance overheads with sprinkling

No. of Sprinkled Terms	0	4	8	16	20	50	100
Time taken by SVD (ms)	134	134.92	133.13	133.76	134.37	135.35	140.18

7.3 Adaptive Sprinkling

Adaptive Sprinkling (AS) is different from sprinkling in that it takes into consideration the knowledge of relationships between classes. We evaluated AS on three types of classification problems. The first involves hierarchical classes, which have an *is-a* taxonomy defined over them. The second type originates from sentiment analysis problems, has an ordinal relationship defined between classes. For example, a textual review accompanied by a rating of 1 (on a 10 point scale) is expected to be more similar to one rated at 2 than another at 10. If numeric ratings are treated as class labels, similarity between classes is a function of this ordering. Finally, we consider orthogonal problems where classes bear no explicit relationship to each other. This is the most frequent category of problems; the datasets used to evaluate sprinkling have flat disjoint classes, and they belong to this category. We used the following datasets in our experiments:

1. **Hierarchical dataset:** This dataset was formed from the 20 Newsgroups collection (Lang 1995) which has seven sub-trees: *comp, rec, talk, alt, misc, soc,* and *sci.* We selected the *comp* and *rec* sub-trees which contain 5 and 4 classes (corresponding to leaf-nodes) respectively. The hierarchy is shown in Figure 7.3. We used 500 documents (cases) from each of these nine classes.
2. **Ordinal dataset:** Classification between ordinal classes is an interesting problem in sentiment analysis literature (Pang & Lee 2005). However, due to the relative youth of the field, no suitable benchmark dataset was readily available. We

therefore compiled a new dataset from reviews on the ``actors and actresses" sub-topic of the *Rateitall.com* opinion website. Each review contained an integer rating (1 to 5 inclusive) assigned by the author. These ratings were used as the class labels. We removed all reviews having less than 10 words, and created 5 equally distributed classes, each with 500 reviews.

3. **Orthogonal dataset:** We used the *acq, crude* and *earn* classes of the Reuters-21578 collection (Reuters 1997) to form this dataset. 500 documents were selected from each class, such that each document belongs to at most one class.

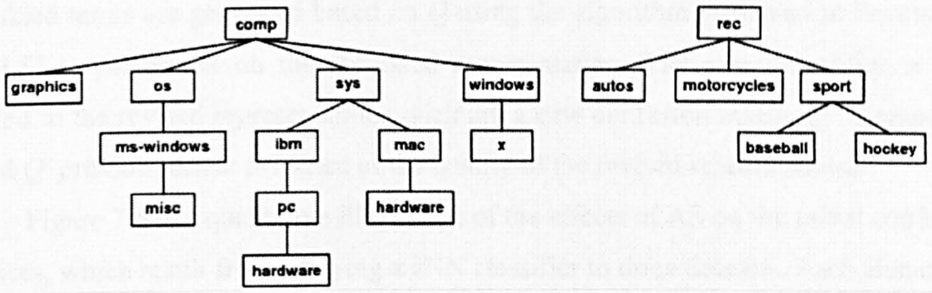


Figure 7.3 Organization of 20 Newsgroups sub-corpus used for evaluating AS over hierarchical classes

All three datasets underwent similar pre-processing. After stop word removal and stemming, binary valued term-document matrices were constructed. For each of the datasets, Information Gain (IG) was used to select the top 1000 discriminating words. For experiments using SVM, we used the SVM^{multiclass} implementation (Joachims 1998), with a linear kernel as before. Since we have single labelled documents in each dataset, and the all classes are distributed equally, accuracy suffices as a measure of effectiveness. For all datasets we performed classification using 10 equally sized train-test pairs, and used the paired t-test to assess significance.

7.3.1 Confusion matrices before and after sprinkling

As described in Section 5.3 high off-diagonal values in a confusion matrix indicate classes that the classifier finds hard to separate. This forms the intuitive basis for using the confusion matrix to generate the sprinkling codes. In our experiments, a 5-fold cross-validation on the raw training data yields five confusion matrices. These five matrices are then used to construct an average confusion matrix Q, each of whose elements are obtained by averaging the corresponding elements in the five matrices. Sprinkled terms are generated based on Q using the algorithm presented in Section 5.3 and LSI is performed on the sprinkled representation. The same classifier is then applied to the revised representations, yielding a new confusion matrix Q'. Comparing Q and Q' provides direct evidence of the quality of the revised representation.

Figure 7.4 is a qualitative illustration of the effects of AS on the initial confusion matrices, which result from applying a *k*NN classifier to three datasets. Each element of the matrix is mapped onto a cell colour. A light colour signifies a high value in that cell, a dark colour signify a low value. For a perfect classification, all cells except those on the diagonal should be dark, as this indicates total agreement between the expert and the classifier.

In all three datasets, we observe that AS results in a reduction in inter-class confusion. The first column in the matrix of Figure 7.4A and the second one of Figure 7.4C, reveal pairs of classes that kNN finds hard to classify. Interestingly, AS succeeded in reducing inter-class confusion, as is revealed by the near-diagonal patterns in matrices of Figures 7.4B and 7.4D.

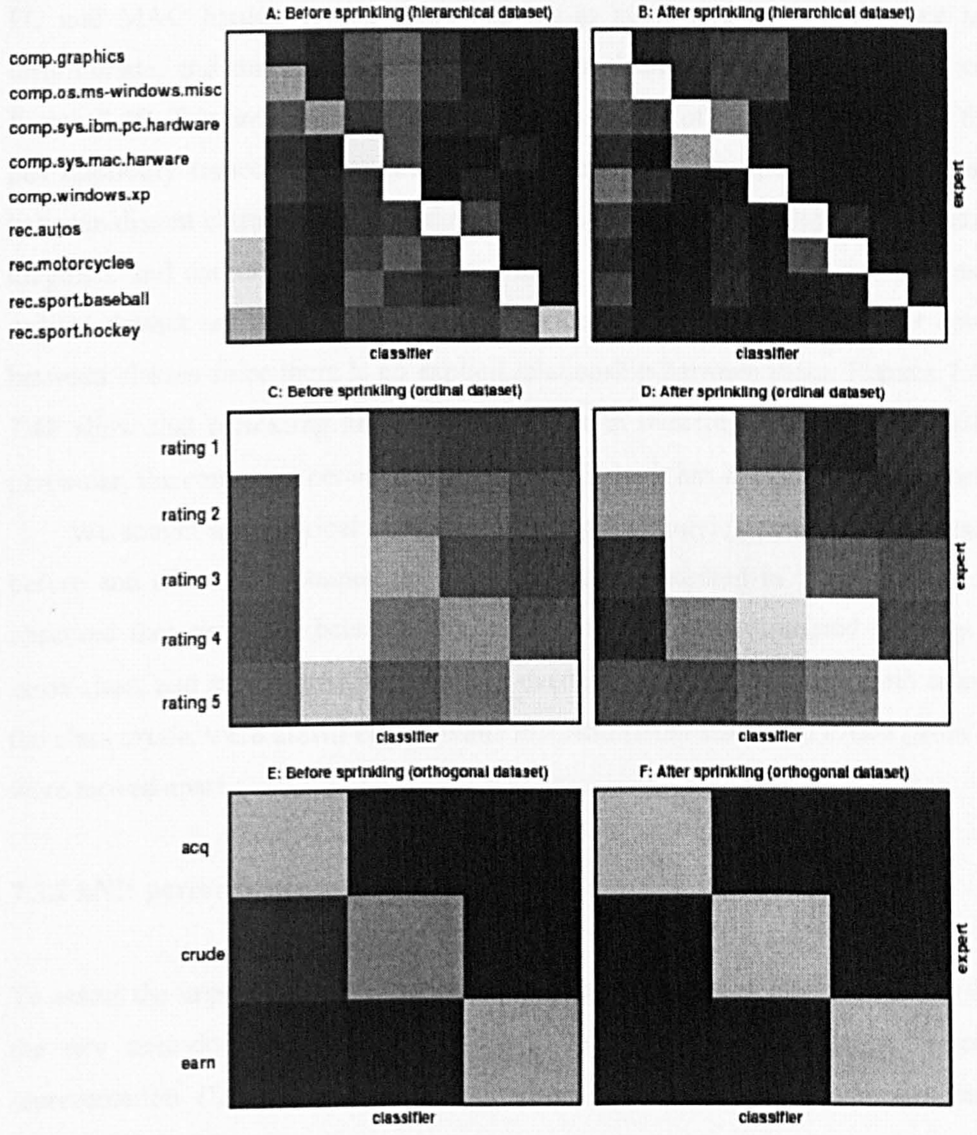


Figure 7.4 Confusion matrices before and after sprinkling

A closer look at the confusion matrices obtained after sprinkling reveals patterns that are consistent with the relationship between classes. In the hierarchical dataset, the confusion is mainly between classes within the same sub-tree. There are two broad confusion zones, one between the five classes of the *comp* subtree, the other between four classes of rec. Furthermore very closely related classes like those corresponding to

PC and MAC hardware, and those relating to autos and motorcycles are hard to discriminate, and this is reflected in the lighter shades in the corresponding cells of Figure 7.4B. For ordinal classes, the confusion matrix of Figure 7.4D shows that AS has implicitly mined the similarity between rating classes and attenuated confusion between distant classes. This is evident from the broad pattern of light shades along the diagonal, and darker shades elsewhere. This is expected, since adjacent classes of an ordinal dataset are the most similar. The orthogonal dataset has the least confusion between classes since there is no explicit relationship between them. Figures 7.4E and 7.4F show that sprinkling has a positive effect in reducing inter-class confusion. In particular, the confusion between classes *acq* and *crude* has been markedly reduced.

We sought an empirical explanation for this by studying similarity between terms before and after AS, obtained using the approach described in Section 4.3.2. It was observed that similarity between words were boosted if they related strongly to the same class, and attenuated otherwise. For example, *opec* and *refinery*, both relevant to the class *crude*, were drawn closer, while *dividend* (from *earn*) and *crude* (from *crude*) were moved apart.

7.3.2 kNN performance before and after AS

To assess the impact of sprinkling we constructed three representations of each dataset: the raw term-document matrix (baseline), the LSI-generated reduced dimensional representation (LSI), and the approximation of the original matrix generated by sprinkled LSI (LSI+AS).

Effects of sprinkling on kNN: We used two variants of kNN, the first based on the Euclidean distance measure (kNNE) and the second on cosine similarity (kNNC). Both use a weighted majority vote from the 3 nearest neighbours. Table 7.8 reports kNN performances, before and after sprinkling, at the LSI dimension empirically found best. These are compared against baseline SVM performance. For each dataset, the performances significantly better ($p < 0.05$) than the rest, are shown in bold. Firstly, we observe that AS leads to sizable improvements in performance of both kNNE and kNNC over the respective baselines. kNNE and kNNC performances with LSI+AS are significantly better than LSI on all datasets. Secondly, LSI+AS enhances kNN performance to be competitive with, and occasionally outperform, baseline SVM.

At different LSI dimensions: Figure 7.5 shows kNNC and kNNE performances over various LSI dimensions. We note that LSI+AS consistently outperforms LSI at all dimensions, on both measures.

The poor performance of all classifiers on the ordinal dataset can be attributed to classes that are not neatly separable. This is partly caused by subjective differences between reviewers, who use different ratings to express similar judgements. The positive impact of AS on confusion matrices in Figure 7.4D suggests that a regression-based technique can fare better than a classifier that attempts to predict a precise rating. Furthermore, the IG measure used for feature selection assumes classes to be disjoint and needs to be reformulated to accommodate inter-class similarity (Mukras et al., 2007).

7.3.3 SVM performance before and after AS

Table 7.9 shows the impact of sprinkling on SVM performance. It may be noted that the confusion matrix used to generate sprinkled terms reflected weaknesses specific to SVM, hence AS should ideally emphasise differences between classes that SVM on its own found hard to classify. The results are in line with our expectation, as LSI+AS significantly ($p < 0.05$) outperforms the baseline on all three datasets. There is some evidence to suggest that LSI alone improves SVM performance, but the difference is not statistically significant except for the orthogonal dataset.

Table 7.8 kNN performance before and after AS

		Hierarchical	Ordinal	Orthogonal
kNNC	Baseline	48.02	25.84	93.47
	LSI	49.53	29.08	94.80
	LSI + AS	60.40	**31.00**	**95.20**
kNNE	Baseline	20.80	25.40	78.60
	LSI	35.73	29.00	91.87
	LSI + AS	59.38	**30.16**	93.80
SVM	Baseline	**65.47**	**30.12**	94.27

The likeness in performance between LSI and LSI+AS on the orthogonal dataset is indicative of the fact that class knowledge plays a less critical role here, in comparison to the ordinal and hierarchical cases.

Table 7.9 SVM performance before and after Sprinkling

		Hierarchical	Ordinal	Orthogonal
SVM	Baseline	65.47	30.12	94.27
	LSI	65.71	31.12	**95.27**
	LSI + AS	**66.33**	**32.08**	**95.27**

7.3.4 AS versus sprinkling

We carried out experiments to compare the effectiveness of AS against sprinkling on the multi-class orthogonal datasets REC and SCIENCE. Table 7.10 shows the accuracy figures obtained after averaging over 15 trials. Over both datasets, we observed a small improvement in the accuracy. This is explained by the ability of AS to improve separability between classes that are more likely to be confused. The results confirm

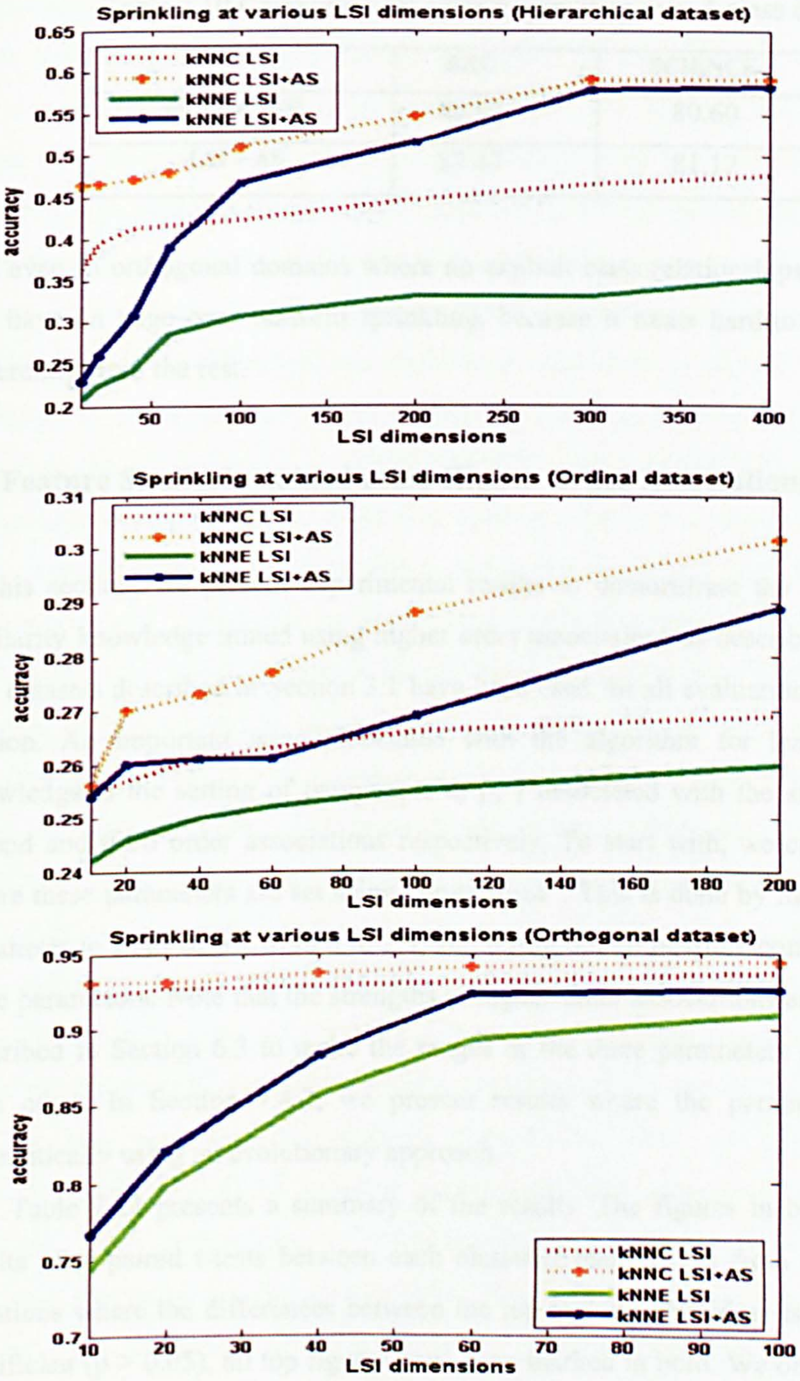


Figure 7.5 kNN performance before and after AS at various dimensions

Table 7.10 Comparing AS against sprinkling over 4-class datasets

	REC	SCIENCE
Sprinkled LSI	86.99	80.60
LSI + AS	87.47	81.12

that even in orthogonal domains where no explicit class relationships are known, AS can have an edge over uniform sprinkling, because it treats hard-to-separate classes differently from the rest.

7.4 Feature Similarity mined using Higher Order Associations

In this section, we present experimental results to demonstrate the effectiveness of similarity knowledge mined using higher order associations as described in Chapter 6. The datasets described in Section 3.1 have been used for all evaluation reported in this section. An important issue associated with the algorithm for learning similarity knowledge is the setting of parameters α, β, γ associated with the strengths of first, second and third order associations respectively. To start with, we consider the case where these parameters are set using "brute force". This is done by incrementing each parameter in steps of 0.1 from 0 to 2.5, and trying out all possible combinations of the three parameters. Note that the strengths of higher order associations are normalized as described in Section 6.3 to make the ranges of the three parameters compatible with each other. In Section 7.4.2, we present results where the parameters are learnt automatically using an evolutionary approach.

Table 7.12 presents a summary of the results. The figures in bold are the best results after paired t-tests between each classifier over results from the 15 trials. In situations where the differences between the top ranking classifiers is not statistically significant ($p > 0.05$), all top figures have been marked in bold. We observe that using second and third order co-occurrences at parameter settings that yield best performance results in better classification accuracies compared to using first-order co-occurrences alone ($\beta,\gamma = 0$). While the differences are statistically significant on all four datasets, the magnitude of improvement is more conspicuous in HARDWARE and RELPOL, which are harder domains, compared to USREMAIL and LINGSPAM, which already recorded high accuracies with simpler approaches. We compared our approaches to a CRN based on similarity relations mined using LSI. It may be noted that the use of higher order co-occurrences leads to better accuracies compared to LSI-based similarities and the differences are statistically significant on all four domains. This is all the more noteworthy in the light of our paired tests that reveal that LSI does better than first order co-occurrences on both HARDWARE and RELPOL, while results are statistically equivalent on the other two datasets. These two observations show LSI does better than using first order associations alone, but is outperformed comprehensively when higher orders are used. While Kontosthathis & Pottenger (2006) show that LSI implicitly models higher order co-occurrences, it is simultaneously constrained by the need to maximize variance across the concept dimensions, and by the need to produce the best k-rank approximation to the original case-feature matrix, in the least-squares sense. The experimental results confirm our intuition that these constraints could prove to be unnecessarily restrictive in the classification domain, and can be relaxed to obtained better performance. Another relative advantage of our approach vis a vis LSI-mined similarity is that we can *explicitly* capture higher order associations and embed into the similarity knowledge. This is useful for facilitating better explanation and visualization of the mined knowledge, as shown in Section 6.6.

We also note that our approach outperforms SVM on all datasets except HARDWARE where SVM performs significantly better. One possible reason for the relatively poor performance in HARDWARE could be a significant overlap in vocabularies used to describe problems in Mac and PC. The problem is compounded by the fact that we ignore class knowledge of training documents while constructing similarity relations between terms. In contrast this is a critical input to SVM. Motivated by this observation, we explored the idea of incorporating sprinkling into the similarity mining algorithm, which is described in Section 5.3.

Table 7.11 reports α, β and γ values at which best performances are observed. Easier domains like USREMAIL and LINGSPAM appear to prefer lower values of β and γ compared to the two harder binary problems HARDWARE and RELPOL, and the two multi-class datasets REC and SCIENCE (refer $\text{GAME}_{\text{class}}$ scores in Table 3.1). We will re-examine this observation in the light of more experimental results in Section 7.4.2.

Table 7.11 Empirically determined best values of α,β and γ

	REC	SCIENCE	HARDWARE	RELPOL	USREMAIL	LINGSPAM
$(\alpha,\beta,\gamma)_{\text{optima}}$	(1, 0.42, 0.49)	(1, 0.95, 1.07)	(1, 0.37, 1.15)	(1, 0.61, 1.04)	(1, 0.21, 0.15)	(1, 0.27, 0.31)

Table 7.12 Comparing Classifier Accuracies

	REC	SCIENCE	HARDWARE	RELPOL	USREMAIL	LINGSPAM
BASE(VSM)	62.79	54.89	59.51	70.54	59.23	85.09
LSI-mined Similarities	82.16	75.37	72.40	93.39	95.83	98.32
SVM	--	--	**78.83**	92.28	**95.83**	96.36
First Order	82.71	77.04	71.71	93.09	95.77	98.26
Higher Order	**85.35**	**82.55**	74.51	**95.30**	**96.40**	**98.59**

7.4.1 Sprinkled Higher Order

In Section 5.3, sprinkling was used to create artificial second order associations between features representative of the same class. In this sub-section, we evaluate the hypothesis that this indeed leads to better classification effectiveness by biasing the feature similarities to reflect class knowledge. We note that the number of sprinkled terms is an important parameter for this algorithm and needs to be set based on training set cross validation for optimal performance. For our evaluations, we simplify this choice by using 8 sprinkled terms per class, as this was empirically found to yield good results.

The results are summarized in Tables 7.13 and 7.14. Sprinkling led to conspicuous improvement in performance over the HARDWARE dataset from 74.51% to 80.44%. This unambiguously points to the importance of class knowledge in this dataset. Table 7.13 suggests that sprinkled higher orders outperforms SVM on all datasets; in the USREMAIL dataset, the improvement is not statistically significant. This is possibly because the domain is simple and had already high recorded accuracies. For the RELPOL domain however, adding class knowledge led to a slight drop in the performance from 95.30% to 93.93% (Table 7.14), which was still significantly better than both LSI and SVM. The drop in RELPOL performance indicates that in this domain, class knowledge is not as important as in HARDWARE. In our current implementation, we have used uniform number of sprinkled terms over all domains. Performance could be improved by optimising the number of sprinkled terms for each individual domain. For example, HARDWARE would be more heavily sprinkled than RELPOL.

Table 7.13 Comparing Sprinkled Higher Orders against SVM

	HARDWARE	**RELPOL**	**USREMAIL**	**LINGSPAM**
Sprinkled Higher Order	**.8044**	**.9393**	**.9630**	**.9838**
SVM	.7883	.9228	**.9583**	.9636

Table 7.14 Comparing Higher Orders with and without Sprinkling

	REC	**SCIENCE**	**HARDWARE**	**RELPOL**	**USREMAIL**	**LINGSPAM**
Sprinkled Higher Order	**.8574**	**.8339**	**.8044**	.9393	**.9630**	**.9838**
Higher Order	**.8530**	.8254	.7451	**.9530**	**.9640**	**.9859**

7.4.2 Using GA to learn parameters

Performing exhaustive search on the parameter space allows us to empirically ascertain the contributions of each co-occurrence order. However, in practice, we would need a mechanism to determine the parameters automatically based on a given text collection. Section 5.4 presents an approach based on Genetic Algorithms to achieve this. The parameters are learnt on the training set, with the objective of maximizing classification accuracy on the unseen test set. Since the test set is not available, we instead set our objective to optimizing classification accuracy over 5-fold cross validation on the training set.

Table 7.15 Comparing effectiveness of empirically determined and GA-learnt parameters

	HARDWARE	RELPOL	USREMAIL	LINGSPAM
Sprinkled HO (parameter learning)	.7938	.9304	**.9593**	.9814
Sprinkled HO	**.8044**	**.9393**	**.9630**	**.9838**

Table 7.16 Parameter values learnt by GA

	HARDWARE	RELPOL	USREMAIL	LINGSPAM
$(\alpha,\beta,\gamma)_{optimal}$	(1,1.88 ,1.56)	(1,1.01 ,1.15)	(1,0.97,0.85)	(1,0.73, 0.96)

We carried out experiments over the binary classification datasets, as preliminary evaluation of the feasibility of this idea. Table 7.15 presents the classification accuracies when the parameters were learnt using the GA-based approach. We used the architecture of Figure 5.5 where sprinkled terms were used as carriers of class knowledge. The accuracy Figures with the learnt parameters are very similar to the Figures obtained by the "brute force" approach presented earlier where the best values are chosen after exhaustively searching the parameter space in fixed increments. While there is still a statistically significant difference in three of four datasets, the very close average values suggest that the GA-based approach holds promise in significantly lowering manual overheads in parameter setting, while still continuing to deliver good performance. We need further research into better tuning of our approach for facilitating faster and more effective search in the parameter space. Table 7.16 shows the values of α,β and γ that were learnt by our algorithm for each of the four datasets. Comparing these values with the corresponding ones in Table 1, we observe a significant increase in the values of β. This can be attributed to the fact that sprinkled terms provide second order co-occurrence paths between terms of the same class. Increasing β thus helps in boosting similarity between terms of the same class, and decreasing similarity between terms of disjoint classes. This explains the greatly improved performance in the HARDWARE domain with sprinkling.

7.5 Chapter Summary

We have presented experimental studies to evaluate supervised extensions of LSI for acquiring relevance knowledge and also evaluated the approach for mining similarity knowledge based on higher-order associations between features. Using LSI has been shown to yield significant improvements over the baseline Vector Space representation, however the absence of class knowledge is a major handicap, especially in complex domains like HARDWARE. Sprinkled LSI incorporates class knowledge in LSI. The resulting relevance and similarity knowledge lead to considerable improvements over LSI performance. The acquired similarity and relevance knowledge can be incorporated into a CRN, or for that matter any other instance-based retrieval formalism. We also showed that sprinkled LSI helps in promoting similarities between features belonging to the same class. From a practical standpoint, it is interesting to note that sprinkling involves minimal computational overheads over LSI on its own. The effectiveness of representations learnt by Adaptive Sprinkling has been demonstrated over three different types of classification problems, and over two classifiers kNN and SVM. Higher Order Associations have been shown to be effective in mining feature similarity, however the absence of class knowledge in learning these similarities is a bottleneck. Experimental results show that using sprinkling as a pre-processing step can help us in overcoming this limitation.

Chapter 8

Fast Case Retrieval Network

The best way to accelerate a PC is at 9.8m/sec/sec. Marcus Dolengo

So far, we have dwelt on how statistical techniques can be used to acquire knowledge for TCBR, with the objective of facilitating effective retrieval. In this chapter we will address issues related to efficiency, i.e. time and space performance of retrieval. This is important in practical usage scenarios, where the feature set size and the number of cases can be extremely large, posing challenges to retrieval strategies and memory requirements.

While CRNs scale up well with increasing casebase size, their retrieval efficiency is critically determined by the size of the feature set and nature of similarity relations defined on these features. In text retrieval applications, it is not unusual to have thousands of terms, each treated as a feature. The aim of this chapter is to propose an approach to improve the retrieval efficiency of CRNs. The basic idea involves introducing a pre-computation phase that eliminates redundant similarity computations at run time. This new retrieval mechanism is referred to as Fast CRN (FCRN). Our experiments reveal that the proposed architecture can result in significant improvement over CRNs in retrieval time without compromising retrieval effectiveness. The architecture also reduces memory requirements associated with representing large casebases.

Section 8.1 presents a concise introduction to the steps involved in CRN retrieval mechanism. We introduce FCRNs in Section 8.2, which is followed by an analysis of computational complexity and memory requirements in Section 8.3. We present experimental results in Section 8.4. Section 8.5 discusses additional issues, such as maintenance overheads that need to be considered while deploying real world applications using FCRNs. We also present an extension to FCRN to cater to varying precision/recall needs. In Section 8.6, the main contributions of this chapter have been summarized. In Appendix A2, we show that the ranking of the retrieved results produced by FCRNs is same as that of CRNs, under both the Euclidean and cosine similarity measures.

8.1 Retrieval in Case Retrieval Networks

The CRN was introduced informally in Section 2.3.2. To facilitate further analysis, we formalize the CRN retrieval mechanism in this section. A CRN is defined over a finite set of s IE nodes E, and a finite set of m case nodes C. Following the conventions used by Lenz (1999), we define a similarity function σ:

$$\sigma: \ E \times E \rightarrow \Re$$

and a relevance function

$$\rho: \ E \times C \rightarrow \Re$$

We also have a set of propagation functions Π_n: $\Re^n \rightarrow \Re$ defined for each node in $E \cup C$. The role of the propagation function is to aggregate the effects of incoming activations at any given node. While Lenz (1999) leaves open the choice of the propagation function, for simplicity we assume that a summation is used for this purpose.

The CRN uses the following steps to retrieve nearest cases:

Step 1: Given a query, initial IE node activations α_0 are determined.

Step 2: *Similarity Propagation*: The activation is propagated to all similar IE nodes.

$$\alpha_1(e) = \sum_{i=1}^{s} \sigma(e_i, e).\alpha_0(e_i) \qquad (8.1)$$

Step 3: *Relevance Propagation*: The resulting IE node activations are propagated to all case nodes

$$\alpha_2(c) = \sum_{i=1}^{s} \rho(e_i, c).\alpha_1(e_i) \tag{8.2}$$

The cases are then ranked in descending order of $\alpha_2(c)$ and the top k cases retrieved.

A CRN facilitates efficient retrieval compared with a linear search through a casebase. As noted in Section 2.3.2, intuitively the speedup is because computation for establishing similarity between any distinct pair of IEs happens only once. Moreover, only cases with non-zero similarity to the query are taken into account in the retrieval process. A detailed complexity analysis is available in (Lenz 1999).

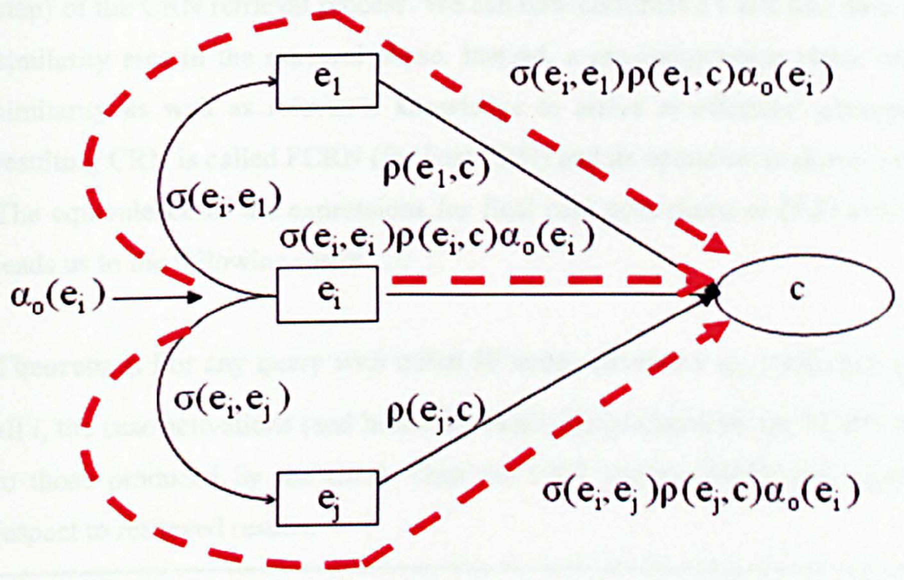
We observe that in the face of a large number of IEs, Step 2 accounts for most of the retrieval time. The idea of FCRN stems from the need to identify and eliminate redundant computations during this similarity propagation step.

8.2 Fast Case Retrieval Network (FCRN)

We now present the basic idea behind FCRN. We substitute the expansion of the term $\alpha_1(e)$ from (8.1) into the expression for final case activation in (8.2). This yields:

$$\alpha_2(c) = \sum_{j=1}^{s} \rho(e_j, c). \sum_{i=1}^{s} \sigma(e_i, e_j).\alpha_0(e_i) \tag{8.3}$$

Let us consider the influence of a single IE node e_i on a single case node c. For this, we need to consider all distinct paths through which an activation can reach case node c, starting at node e_i. Figure 8.1 illustrates three different paths through bold dashed arrows from e_i to c, along with activations propagating through each path.



e_1
$\sigma(e_i,e_1)\rho(e_1,c)\alpha_0(e_i)$
$\rho(e_1,c)$
$\sigma(e_i,e_1)$
$\sigma(e_i,e_i)\rho(e_i,c)\alpha_0(e_i)$
$\alpha_0(e_i)$
e_i
c
$\sigma(e_i,e_j)$
$\rho(e_j,c)$
e_j
$\sigma(e_i,e_j)\rho(e_j,c)\alpha_0(e_i)$

Figure 8.1 Different paths through which an activation can reach case *c* from an IE e_i

We observe that the influence of node e_i on node *c* can be computed as the aggregation of effects due to all nodes e_j that e_i is similar to, and is given by:

$$inf\,(e_i,c) = \sum_{j=1}^{s} \rho(e_j,c)\sigma(e_i,e_j)\alpha_0(e_i). \tag{8.4}$$

The last term can be extracted out of the summation as follows:

$$inf\,(e_i,c) = \left\{\sum_{j=1}^{s} \rho(e_j,c)\sigma(e_i,e_j).\right\}\alpha_0(e_i) \tag{8.5}$$

We refer to the term within parenthesis as the "effective relevance" of the term e_i to case *c* and denote it by Λ ($e_{i,}$ *c*). It can be verified that (8.3) can be alternatively rewritten as:

$$\alpha_2(c) = \sum_{i=1}^{s} \Lambda(e_i,c).\alpha_0(e_i) \tag{8.6}$$

The significance of this redefinition stems from the observation that given an effective relevance function Λ: E × C → ℜ, we can do away with Step 2 (similarity propagation step) of the CRN retrieval process. We can now construct a CRN that does not use any similarity arcs in the retrieval phase. Instead, a pre-computation phase makes use of similarity as well as relevance knowledge to arrive at effective relevances Λ. The resulting CRN is called FCRN (for Fast CRN) and its operation is shown in Figure 8.2. The equivalence of the expressions for final case activations in (8.2) and (8.6) above leads us to the following result.

Theorem 1. For any query with initial IE node activations α_0, such that $\alpha_0(e_i) \in \Re$ for all *i*, the case activations (and hence the rankings) produced by the FCRN are identical to those produced by the CRN. Thus the CRN and the FCRN are equivalent with respect to retrieved results.

Precomputation Phase

The similarity and relevance values are used to pre-compute the effective relevance values

$$\Lambda(e_i, c) = \left\{ \sum_{j=1}^{s} \rho(e_j, c)\sigma(e_i, e_j). \right\}$$

Retrieval Phase

Step 1: Given a query, initial IE node activations α_0 are determined.

Step 2: The resulting IE node activations are propagated directly to all case nodes

$$\alpha_2(c) = \sum_{i=1}^{s} \Lambda(e_i, c).\alpha_0(e_i)$$

The cases are then ranked according to their activations, and the top *k* retrieved

Figure 8.2 Precomputation and Retrieval in FCRN

Figure 8.3 shows an example CRN depicting a trivial setup with 4 IEs and 3 cases, and the corresponding equivalent FCRN. It is observed that while the relevance values in the original CRN were sparse, the effective relevance values in the FCRN are relatively dense. This is because in the FCRN an IE is connected to all cases that contain similar

IEs. In the example shown, the effective relevance between case C_1 and Information Entity IE_1 is computed as follows:

$$\begin{aligned} A(IE_1,C_1) &= \rho(IE_1,C_1)\sigma(IE_1,IE_1) + \rho(IE_2,C_1)\sigma(IE_1,IE_2) + \rho(IE_3,C_1)\sigma(IE_1,IE_3) + \\ &\quad \rho(IE_4,C_1)\sigma(IE_1,IE_4) \\ &= (1\times1) + (0\times0) + (0\times0.5) + (1\times0.7) = 1.7 \end{aligned}$$

Other elements of the effective relevance table can be similarly computed. It is interesting to note that the effective relevance of the *i*th IE with the *j*th case is given by the dot product of the *i*th row of the similarity table (σ) with the *j*th row of the relevance table (ρ).

In practice, similarity measures based on Euclidean distance or cosine similarity are often used to evaluate similarity between cases. Appendix A2 shows how the FCRN can be extended to handle these distance measures. It also presents a proof that ranking of cases produced by FCRNs is same as that produced by a CRN, under both of these distance measures.

8.3 Time and Space Complexity of FCRN

In this section, we formalize our intuitions on efficiency improvements obtained by FCRNs.

8.3.1 Time Complexity Analysis

Let us compare the retrieval time complexity of FCRNs with CRNs. Figure 8.4 illustrates the pseudo-codes for retrieval using the CRN and FCRN.

Relevance function ρ

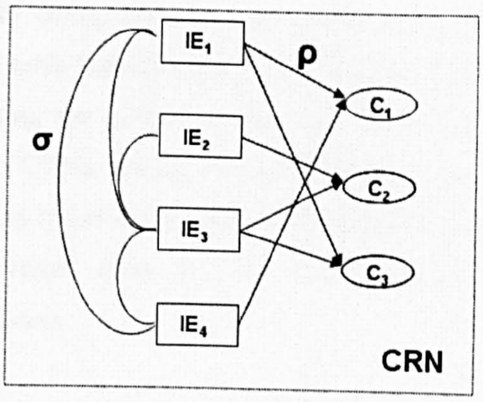
	IE_1	IE_2	IE_3	IE_4
C_1	1	0	0	1
C_2	0	1	1	0
C_3	1	0	1	0

Similarity function σ

	IE_1	IE_2	IE_3	IE_4
IE_1	1	0.0	0.5	0.7
IE_2	0.0	1	0.5	0.0
IE_3	0.5	0.5	1	0.3
IE_4	0.7	0	0.3	1

Effective Relevance function Λ

	IE_1	IE_2	IE_3	IE_4
C_1	1.7	0	0.8	1.7
C_2	0.5	1.5	1.5	0.3
C_3	1.5	0.5	1.5	1.0



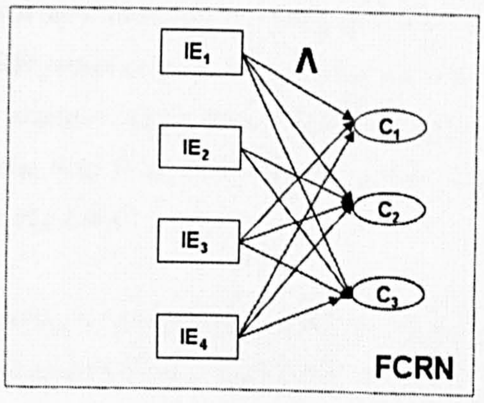


Figure 8.3 A CRN over 3 cases and 4 IEs, and an operationally equivalent FCRN

The retrieval complexity is a function of loops /* A */ and /* B */ in the pseudo-codes:

$$complexity(CRNRetrieval) \propto O(A \times B \times C)$$

and

$$complexity(FCRNRetrieval) \propto O(B \times C)$$

The following two reasons contribute to the speedup in FCRN retrieval:

1. Step A in the CRNRetrieval pseudo-code involves spreading activation to IE nodes similar to the query IEs based on similarity values. This step is eliminated in FCRN retrieval since the similarity knowledge is transferred to

the effective relevance values during the pre-computation step. Thus, FCRN retrieval amounts to a simple table lookup for all cases "effectively" relevant to the query IEs and aggregating the scores received by each case from the individual query IEs. Using FCRNs, we can obtain efficiency very similar to inverted files typically used in Information Retrieval applications (Rijsbergen 1979). However unlike inverted files, FCRNs also integrate similarity knowledge in the retrieval process.

2. Step B in FCRNRetrieval involves a loop over IE nodes activated by the query. In contrast, Step B of the CRN retrieval loops over all IEs similar to IE nodes activated by the query. In a situation where most IEs are connected to many others by non-zero similarities, Step B in FCRN would involve much fewer iterations compared to step B of a CRN.

While the above two factors lead to saving in retrieval time in FCRN, it is important to note that the step C of FCRN retrieval could be more expensive than the step C of the CRN. This is because effective relevance values in FCRN are less sparse compared to relevance values in CRN; thus, we would expect an IE to have non-zero relevances to a larger number of cases in FCRN, compared to a CRN. If the CRN relevance values are optimally dense in that each IE has a non-zero relevance to every case (as is expected to be true with relevances acquired by LSI), the step C of a CRN is equivalent in time complexity to the step C of FCRN, since CRN relevances are just as non-sparse as the FCRN effective relevances. In such a case, FCRN is guaranteed to be faster than the CRN. However, in the presence of highly sparse relevance values and dense similarity relations, the step C of FCRN would be slower than the corresponding step of the CRN, thus partially offsetting the savings obtained by eliminating Step A and reducing time complexity of Step B. In the empirical evaluations reported later in this chapter, we focus on dense relevance relations as is typical when the knowledge is acquired using statistical techniques presented earlier in this thesis; however, it is important to note that the performance gains obtained with FCRNs would be less conspicuous when the CRN relevance values are sparse.

```
CRNRetrieval
    FOR each activated query IE (attribute A, value $V_q$ in query)        /* A */
        Determine all related IEs using similarity function $\sigma$
        FOR each IE that is found relevant                                 /* B */
            FOR each case associated with that IE                          /* C */
                Increment score of case
            END FOR
        END FOR
    END FOR
    Rank and display related cases

FCRNRetrieval
    FOR each activated query IE (attribute A, value $V_q$ in query)        /* B */
            FOR each case associated with that IE                          /* C */
                Increment score of case
            END FOR
    END FOR
    Rank and display related cases
```

Figure 8.4 Pseudo-codes for retrieval using CRN and FCRN

8.3.2 Memory Requirements

Typically CRNs consume more memory when compared to a flat casebase, which has a linear listing of cases along with their constituent attribute values. This difference can be largely attributed to the following two factors: CRNs explicitly record $|E|$ number of values corresponding to IEs, and $|E|^2$ values are required to model similarities between IEs. In addition we have $|Casebase| \times |E|$ relevance values between the IEs and the cases.

A flat casebase that models the case memory as a linked list of all cases will need to store $|Casebase|$ number of cases and $|Casebase| \times |E|$ number of relevance values.

$$\begin{aligned}\text{memory (flat casebase)} &\propto |\text{Casebase}| \times |\text{E}| + |\text{Casebase}| \\ &\propto |Casebase| \times (|E| + 1)\end{aligned}$$

The memory requirement of a CRN is approximately given by:

$$\begin{aligned}\text{memory (CRN)} &\propto |\text{E}| + |\text{CaseBase}| + |\text{E}|^2 + |\text{Casebase}| \times |\text{E}| \\ &\propto |E| + |E|^2 + |CaseBase| \times (|E|+1) \\ &\propto |\text{E}| + |\text{E}|^2 + \text{memory(flat casebase)}\end{aligned}$$

In FCRN we do not need to explicitly record the similarities between IEs, since this knowledge is contained within effective relevance values. The memory requirement of FCRN is given by:

$$\begin{aligned}\text{memory (FCRN)} &\propto |\text{E}| + |\text{CaseBase}| \times (|\text{E}|+1) \\ &\propto |\text{E}| + \text{memory(flat casebase)}\end{aligned}$$

In textual applications, the number of IEs could be extremely large, and the saving of $|E|^2$ could mean substantial gains in terms of memory requirements.

It is worth noting that while the in-memory requirement for FCRN retrieval is considerably less than in CRN, we would still need to store the $|E|^2$ similarity values for off-line maintenance. In a situation where a particular IE is deleted, we would need to re-evaluate the effective relevance values to reflect this change. This is possible only when the similarity information is available.

8.4 Experimental Results

In this section, we present empirical results to illustrate FCRN efficiency in practical applications. The objective of our first set of experiments is to observe how CRNs and FCRN scale up with increasing number of IEs, and with varying nature of similarity interconnections between these IEs. Towards this end, it is sufficient to simulate a large number of IEs and cases with randomly generated similarity and relevance values. The synthetic nature of the datasets is not a major concern, since we are not really concerned with the actual cases retrieved. Sparseness of similarity values can be controlled by forcing a fraction of these values to 0. In any real world application, the actual non-zero similarity and relevance values used would be different from the randomly generated values used in our evaluation, but the time complexity of the retrieval process is independent of the actual values used, since neither the CRN nor FCRN exploit the distributions of values to alter the retrieval process. So our experiments are expected to provide fair estimates of efficiency over realistic datasets. An experimental strategy similar to ours was also used in (Lenz 1999).

Table 8.1 shows the impact of the increase in number of IE nodes on the retrieval time. For this experiment, the query was randomly generated and IE nodes activated accordingly. The casebase has 1000 cases. The similarity matrix is optimally dense in that each IE node is connected to each other by a non-zero similarity value. Thus this result may be viewed as a worst-case comparison of the CRN performance against FCRN. It may be noted that the CRN retrieval time increases almost linearly as the number of IE nodes increases from 1000 to 6000. As the number of IEs goes beyond 6000, CRN performance degrades steeply. In contrast, the FCRN shows stable behaviour with increasing number of IEs. This is attributed to the savings in similarity computation, and corresponds closely to our theoretical analysis in Section 3.2.

The objective of our next experiment is to empirically evaluate the impact of the nature of similarity interconnections on the relative performance of the CRN and the FCRN. We recall that a bulk of the savings in retrieval time with FCRNs can be accounted for by the fact that FCRN does away with the similarity propagation step. The time consumed in similarity propagation is critically dependent on the density of the similarity matrix, which is defined as the proportion of non-zero similarity values in the similarity matrix. We conducted an experiment to study the FCRN performance against CRN, as a function of the similarity matrix density. Our experimental setup is similar to that in the first experiment. We simulate 8000 IEs and 1000 cases with

Table 8.1 Retrieval time as a function of the number of IE nodes

No. of IE Nodes	CRN Retrieval Time (secs.)	FCRN Retrieval Time (secs.)
1000	0.04	$<10^{-3}$
2000	0.12	$<10^{-3}$
3000	0.22	$<10^{-3}$
4000	0.35	$<10^{-3}$
5000	0.49	$<10^{-3}$
6000	0.66	$<10^{-3}$
7000	1.42	0.01
8000	3.40	0.01
9000	3.86	0.01
10000	4.98	0.02

randomly generated similarity and relevance values. We now relax the density of the similarity matrix, by deliberately setting a value of 0 to a fraction of the similarity values, and compare FCRN performance against the CRN, for different settings of similarity matrix density. The results are shown in Table 8.2. As the density increases from 0 (when no IE node is similar to any other node) to 1 (when all IE nodes are related to all others), the CRN retrieval time increases considerably from a sub-millisecond to about 3.38 seconds. Since FCRN does away with the step of similarity propagation across IEs, its performance is not critically impeded by growth in similarity matrix density. The very small increment in the FCRN retrieval time when the density increases from 0.8 to 1.0 is not surprising, given the fact that the effective relevance values are influenced by the density of the similarity matrix. Hence an increase in number of similarity interconnections can have an adverse effect on the sparseness of the effective relevance values, leading to a consequent slowdown in retrieval. It may be noted that retrieval times recorded in all Tables in this section are rounded to two significant decimal places.

In addition to empirical evaluation on synthetic data, we also carried out experiments on a real world classification task over a textual dataset comprising 2189 personal emails organized into 76 folders (classes). Each class corresponds to one of the folders (like "sports", "hobbies" or "meetings") into which the emails are organized. The total number of features in this dataset is 32,699. Since many of these features have very poor discriminatory power, the feature set size was pruned to 6000 using chi-square based feature selection (Yang & Pederson 1997). A CRN was constructed to classify incoming emails into one of the 76 classes. Instead of modeling the emails as textual cases as is usually done, we treated the classes as cases. Thus the CRN had 6000 IE nodes and 76 case nodes.

Table 8.2 Retrieval time as a function of the density of similarity matrix

Density of the Similarity Matrix	**CRN Retrieval Time (secs.)**	**FCRN Retrieval Time (secs.)**
0	$<10^{-3}$	$<10^{-3}$
0.2	0.92	$<10^{-3}$
0.4	1.71	$<10^{-3}$
0.6	2.43	$<10^{-3}$
0.8	2.81	$<10^{-3}$
1.0	3.38	0.01

Traditional techniques for modelling relevance do not directly apply in our case, since relevance values in our architecture relate IEs to classes, instead of relating IEs to cases. In our classifier, we use the chi-square metric (Yang & Pederson 1997) as a measure of the relevance of an IE to a particular class. The chi-square metric measures the lack of independence between an IE and a class. Thus the relevance value is 0 when an IE is independent of the class, and high when it is strongly dependent.

The similarity between IEs is computed using Latent Semantic Indexing (LSI), using the method described in Section 4.3.2. We have seen earlier that LSI has an adverse effect on the sparseness of the similarity matrix. As the number of IEs increase, this can lead to considerable slowdown in retrieval or classification.

In Table 8.3, we report experimental results comparing the time performances of the FCRN against a CRN in this domain. As the number of IEs increase from 1000 to 6000, the CRN slows down considerably. The slowdown is especially conspicuous when the number of IEs exceeds 4000. In contrast, the FCRN scales up well.

Table 8.3 Time performance as a function of the number of IEs in the email dataset

No. of IE Nodes	**CRN Retrieval Time (secs.)**	**FCRN Retrieval Time (secs.)**
1000	0.02	$<10^{-3}$
2000	0.22	$<10^{-3}$
3000	0.34	$<10^{-3}$
4000	1.01	$<10^{-3}$
5000	1.87	0.01
6000	2.82	0.01

8.5 Discussion

In this section we consider some additional issues that need to be taken into account when building CBR systems using FCRNs. We also present an extension of the FCRN idea to allow for flexible control of precision and recall during retrieval.

8.5.1 Computation Node

One obvious limitation of the CRN mechanism is its inability to handle query values (in the textual case, terms) that are not present in the predefined set of IEs used to build the CRN. To address this issue, Lenz (1999) presents the concept of a computation node which is created at run time. A computation node represents an IE corresponding to the new query value. The similarity of the computation node to existing IE nodes is computed at run-time using a similarity function that needs to be defined over the attribute space. Once the new similarity arcs are constructed, the retrieval can proceed in the usual manner. With FCRNs, a similar computation node creation step is involved. However, it only plays a role in activating the IE nodes via the newly constructed similarity arcs. If one or more of these IE nodes were already activated, the new activations are added to the existing values. Once the IE node activations (α_0 values) are evaluated, the case nodes are activated directly using the effective relevance values.

8.5.2 Maintenance Overheads with FCRNs

The downside of FCRNs is that incremental and batch maintenance of the casebase involves extra pre-computations. The effective relevance values need to be recomputed each time new cases or IEs are inserted or existing cases/IEs deleted or edited. However, the recomputations can be limited to only those effective relevance values that could potentially be affected. We consider two specific update scenarios below:

1. *Insertion of new cases or deletion of existing cases*: Deletion of an existing case is straightforward and only involves setting all effective relevance values connecting IEs to that case, to zero. This does not influence the effective relevances of the other cases. However, when a new case is added, the effective relevances of IEs present in the case to the case needs to be pre-computed, based on the similarity and relevance knowledge. Existing effective relevance values of IEs to the remaining cases are not affected, since effective relevance of an IE to a case is independent of the relevance of the IE to any other case in the casebase.

2. *Insertion of new IEs or deletion of existing IEs:* When an existing IE is deleted, effective relevances of all IEs having non-zero similarity to the deleted IE, need to be updated. This can prove to be computationally expensive, especially in the face of large numbers of IEs and cases. We present an efficient update strategy (we have not empirically evaluated this claim) that is based on two key ideas. Firstly, we make incremental changes to existing effective relevance values, rather than recomputing these values from scratch. Secondly, we eliminate redundant computations by restricting incremental changes to only those effective relevance values that can get affected. When an IE node e_d is deleted, the effective relevance of a node $\Lambda(e_i,c)$ is decremented by an amount $\Delta\Lambda(e_i,c)$ to yield the revised relevance value $\Lambda^*(e_i,c)$ which is given by:

$$\Lambda^*(e_i,c)=\begin{cases}0 \ when\ i=d\\ \Lambda(e_i,c)-\Delta\Lambda(e_i,c)\ where\ \Delta\Lambda(e_i,c)=\sigma(e_i,e_d)\rho(e_d,c)\ otherwise\end{cases}$$

These operations can be speeded up by maintaining an *update table*, which is constructed from the similarity and relevance tables and plays the role of an inverted index. A lookup on the table shows the incremental change that must be made on each of the affected effective relevance values and saves the overhead of computing the values from scratch

It may be noted that no updates are needed in situations where $\Delta\Lambda(e_i,c)$ evaluates to zero. This happens when either $\sigma(e_i,e_d)$ is 0 or when $\rho(e_d,c)$ is 0. The update table eliminates such redundant computations by restricting incremental changes to only those effective relevance values that get affected.

As in the case of IE deletion, when a new IE is added, the effective relevances of all IEs bearing non-zero similarity to the new IE need to be re-evaluated. When a new IE node e_n is added, the revised relevance values are given by:

$$\Lambda^{*}(e_i,c) = \begin{cases} \sum_{j=1}^{s} \rho(e_j,c)\sigma(e_i,e_j).\ \textit{when } i = n \\ \Lambda(e_i,c) + \Delta\Lambda(e_i,c)\ \textit{where } \Delta\Lambda(e_i,c) = \sigma(e_i,e_n)\rho(e_n,c)\ \textit{otherwise} \end{cases}$$

Again, we can restrict incremental updates to only those effective relevance values that get affected by the IE insertion.

We note that it may be restrictive to suppose that the update operations can always be localized to those similarity and relevance values that are immediately affected by the nodes inserted or deleted. The approaches outlined above for speeding up updates work well when the similarity and relevance knowledge are externally obtained (as from background knowledge like WordNet) or are derived from local properties of the collection (the relevance of an IE to a case is not dependent on other IEs or cases). However they may result incorrect updates when similarity or relevance knowledge is introspectively derived from global properties of the collection. Let us consider a situation where the relevances are obtained by combining local measures like term frequency and global measures like inverse document frequency. A single case deletion will necessitate the recomputation of inverse document frequencies pertaining to all relevance values. As with relevance values, similarity knowledge may need revision each time an update is made. In realistic situations, such bulk updates will be computationally expensive. A practical approach would be to perform incremental local updates as outlined above whenever a node is inserted or deleted, and relegate bulk recomputations to a later time, when enough updates would have happened to make significant impact on the global measures. It is important to note that this recomputation overhead when using introspective techniques to acquire similarity and relevance knowledge is not specific to the FCRN, but is a concern shared by CRN and the flat casebase representation as well.

8.5.3 Multiple-pass retrieval using FCRNs

Textual domains often come with a wide variety of retrieval requirements. In retrieval applications, we may prefer to have a high-precision search in certain situations, high-recall search in others. One way to realize these diverse requirements is to control the number of IE nodes that are activated during the retrieval process. Thus a very high precision search might look for retrieving cases where at least one of the query IEs is explicitly present. In such situations we can have a CRN retrieval that bypasses the step of identifying similar IEs. We call this "zero-pass" activation. The CRN described in Section 3.1 uses one step of activating IEs similar to the query IEs. We refer to this as "one-pass" activation. A "two-pass" activation will involve an additional step of identifying IEs similar to the IEs identified as similar in one-pass activation.

Thus a two pass activation involves the following three steps of spreading activation:

Step 1: Activate IEs similar to the query IEs using similarity arcs.

Step 2: Activate IEs similar to the IEs activated in Step 1 using similarity arcs.

Step 3: Use relevance arcs to spread activation to cases from all IEs activated in Step 2.

In the following, we show that a two-pass CRN retrieval can be efficiently modeled using FCRN. The case activation of a two-pass CRN would be given by:

$$\alpha_2(c) = \sum_{k=1}^{s} \rho(e_k,c).\sum_{j=1}^{s} \sigma(e_j,e_k)\left\{\sum_{i=1}^{s} \sigma(e_i,e_j).\alpha_0(e_i)\right\} \tag{8.7}$$

We consider the influence of a single IE node e_i on a single case node c. The first step is to aggregate effects due to all intermediate nodes e_j through which similarity may propagate from e_i to e_k. In the next step, the activation of node c is obtained as an aggregation of relevance propagation from e_k to c for all instantiations of k. The final expression is given by

$$\text{inf}(e_i,c) = \sum_{k=1}^{s} \rho(e_k,c)\sum_{j=1}^{s} \sigma(e_j,e_k)\sigma(e_i,e_j)\alpha_0(e_i) \tag{8.8}$$

The last term can be extracted out of the summation to yield

$$\text{inf}(e_i,c) = \left\{\sum_{k=1}^{s} \rho(e_k,c)\sum_{j=1}^{s} \sigma(e_j,e_k)\sigma(e_i,e_j)\right\}\alpha_0(e_i) \tag{8.9}$$

The term within parenthesis is the effective relevance of the term e_i to case c:

$$\Lambda(e_i,c) = \left\{\sum_{k=1}^{s} \rho(e_k,c)\sum_{j=1}^{s} \sigma(e_j,e_k)\sigma(e_i,e_j)\right\} \tag{8.10}$$

It can be verified that (7) can be alternatively rewritten as

$$\alpha_2(c) = \sum_{i=1}^{s} \Lambda(e_i,c).\alpha_0(e_i) \tag{8.11}$$

The equivalence of the expressions for final case activations in (8.7) and (8.11) above, leads us to the following result.

Theorem 2. For any query with initial IE node activations α_0 , such that $\alpha_0(e_i) \in \Re$ for all i, the case activations (and hence the rankings) produced by the FCRN with effective relevances computed as given by (8.10) above, are identical to those produced by a two pass CRN.

We can make similar extensions for cases where the number of passes is more than 2, each higher pass attempting to achieve a higher recall while possibly sacrificing precision.

It may be noted that CRN retrieval complexity is critically influenced by the number of passes, since this determines the number of spreading activation cycles through the similarity arcs. In contrast, the worst-case complexity of retrieval in a FCRN is independent of the number of passes, since pre-computed effective relevance values obviate the need for multiple rounds of similarity propagation. This property of FRCNs facilitates multiple pass retrievals at similar orders of retrieval time as in a single pass activation. In practice we can have effective relevance values pre-computed for different multiple pass networks, and the precision and recall can be tuned by switching between these options. The time complexity of FCRN is stable across these options. It is important to note that in certain applications it would be important to discount the influence of cycles (an IE node reinforcing its own activation over multiple passes) in the activation process. One solution is to use marker passing (Wolverton 1995).

8.6 Chapter Summary

We have presented a Fast Case Retrieval Network formalism that remodels the retrieval mechanism in CRNs to eliminate redundant computations. This has significant implications in reducing retrieval time and memory requirements when operating over casebases indexed over large numbers of IEs and cases. A theoretical analysis of computational complexity and memory requirements comparing FCRNs against CRNs is presented. Experimental results over large casebases demonstrate significant speedup in retrieval with FCRN in the presence of dense similarity and relevance values, as is typical with statistically acquired knowledge. As part of future work, we plan to conduct detailed studies of the impact of density of relevance values on the relative performances of FCRN and CRN.

Chapter 9

Conclusion

Thus grew the tale of Wonderland;
Thus slowly, one by one,
Its quaint events were hammered out-
And now our tale is done Lewis Carroll

This thesis has investigated the problem of acquiring knowledge for TCBR applications using statistical approaches, and proposed novel approaches to acquire knowledge with the goal of improving effectiveness and efficiency of retrieval. In this chapter we conclude the thesis by taking stock of our main contributions and identifying promising areas for future work.

9.1 Contributions

We had enumerated the objectives of our research in Section 1.2. In this section, we examine our contributions in the light of those objectives.

1. *Propose supervised extensions of LSI to mine relevance knowledge in classification domains.* While LSI has been shown to be useful for knowledge acquisition in TCBR, it is limited by its inability to take into confidence class knowledge of training documents in supervised classification domains. We have presented an analysis of this problem that shows the need to strike a tradeoff between the often conflicting goals of preserving the structure of the

original case-feature representation, and improving class separability. We have presented sprinkling, an approach that attempts to strike this balance while incorporating class knowledge into LSI. The approach is based on the simple idea of appending class labels to training cases, thereby augmenting the set of features. The appended features are referred to as sprinkled terms. When LSI is carried out over the augmented representation, terms representative of the same class are pulled closer to each other. The features extracted with sprinkled LSI are better at discriminating between classes compared to those mined by LSI on its own. To summarize our contributions, we note several interesting aspects of sprinkling. Firstly, our experimental studies verify that sprinkling succeeds in enhancing the performance of instance based learners like kNN to make them comparable with, or outperform state-of-the-art techniques like SVM. This result is of potential interest not only to TCBR, but to the wider Machine Leaning community as well, because of its practical implications for applications where lazy incremental updates are desirable. Also, while SVM-like kernel methods suffer from the "black-box" syndrome, kNN is well recognized to be suitable for explanation and visualization, making expert-initiated refinement possible. Secondly, sprinkling is a simple approach that yields significant improvements while incurring nominal overheads in terms of computation time. Thus it can be easily integrated into existing practical LSI systems. Thirdly, though presented in the context of LSI, sprinkling can be used to generate revised representations usable by any approach founded on the vector space model. Sprinkled LSI can also be used to mine similarity knowledge that uses combination of co-occurrences and class affiliations to mine feature similarities.

2. *Propose approaches that extend the scope of LSI to handle situations where class inter-relationships are critical, e.g. hierarchical and ordinal domains.* One limitation of sprinkling is that it treats all classes and classifiers equally,

and fails to take into account relationships between classes, as exist in hierarchical and ordinal classification tasks. Even in the absence of an explicit relationship, some classes are more easily separable than others, and the complexity of decision boundaries should ideally influence the number of sprinkled terms. Our next contribution in the form of Adaptive Sprinkling addresses these concerns, by exploiting information implicitly captured in confusion matrices generated by classifiers. An advantage of this approach is that it does not need the knowledge engineer to specify the precise relationship between classes in advance. Experiments on hierarchical and ordinal datasets conclusively demonstrate the effectiveness of this approach. To our knowledge, ours is the first work combining the strengths of LSI, like higher order co-occurrence modeling and the ability to recover from word choice variability, with the knowledge of class relationships as inferred from confusion matrices. The result is a revised vector space representation that adapts itself to domain needs. The approach used in our work may be useful beyond the context of TCBR which presupposes the notion of instance based retrieval. We have also shown that AS-generated SVM representations result in significant improvements in SVM performance as well. The ability to exploit confusion matrices and generate representations tailored to classifier needs is a contribution of potential interest to the Machine Learning community.

3. *Propose supervised and unsupervised approaches to exploit higher order associations to mine feature similarity.* We have presented an approach for exploiting higher-order associations between words to acquire similarity knowledge for CRNs. We demonstrated the importance of higher order co-occurrences in determining word similarity, presented both supervised and unsupervised algorithms for mining such associations and proposed a word similarity model, whose parameters are learnt using an evolutionary approach. We have demonstrated the effectiveness of the learnt similarity knowledge and

shown that using second and third order-co-occurrences yields better results than using first-order co-occurrence alone. Another contribution of our research is to incorporate class knowledge into the process of mining higher order associations. We have demonstrated the effectiveness of this extension as our approach outperforms state-of-the-art classifiers like SVM and LSI/*k*NN on classification tasks of varying complexity. Though the work has been presented in the context of CRNs, it can be easily extended to learn similarity knowledge over other retrieval formalisms.

4. *Propose a fast retrieval formalism that can use the acquired relevance and similarity knowledge to facilitate effective retrieval while minimizing retrieval time by cutting down on redundant computations.* We have presented a Fast Case Retrieval Network formalism that remodels the retrieval mechanism in CRNs to eliminate redundant computations. This has significant implications in reducing retrieval time and memory requirements when operating over casebases indexed over large numbers of IEs and cases. A theoretical analysis of computational complexity and memory requirements comparing FCRNs against CRNs is presented. Experimental results over large casebases demonstrate significant speedup in retrieval with FCRN. It may be noted that FCRN can be applied to improve retrieval efficiency in large scale non-textual CBR applications, as well.

5. *Propose novel approaches to visualize and estimate complexity of textual casebases, so that they can be meaningfully compared.* We presented a simple approach to visualize textual casebases. The stacked image display can help knowledge engineers to get a bird's eye view of the domain, thus facilitating knowledge acquisition. The visualization has three main advantages over other approaches. Firstly, it shows case and feature clusters in relation to each other, thus allowing case clusters to be explained in terms of feature clusters, and vice

versa. Secondly, since stacking does not rely on any abstraction, it preserves the structure of cases and displays case and feature vectors as they are. This helps casebase maintenance since noisy cases, redundant features or "bridge" features are revealed. Finally, stacking is fast and simple to implement, has no convergence problems, and is parameter-free for all practical purposes. We have also introduced a complexity measure founded on the idea of stacking. We showed that in classification tasks, an adapted version of this measure corresponds closely to accuracies reported by standard classifiers.

9.2 Desirable Extensions

This research reported in this work has attempted to throw new light on the area of acquiring knowledge for TCBR, with the goal of facilitating effective and efficient retrieval. However, given the breadth of scope both in terms of techniques and potential applications, it is but natural that this is far from finished work. In this section, we take a closer look at the limitations of the work reported here, and identify ways of addressing these limitations to fill in gaps or extend its scope.

Lack of linguistic or background knowledge. Our work relies on knowledge introspectively acquired from a collection of training cases. However, recently the use of background knowledge like linguistic knowledge in the form of WordNet or web collections like Wikipedia (Gabrilovich & Markovitch 2007) have received a lot of attention from researchers in text mining. While experiments reported in this thesis use bag of words as the starting point for mining relevances or similarity, there is no inherent limitation in using semantically richer units like phrases or attribute values extracted by Information Extraction as information entities instead (Orecchioni et al., 2007) Relevance mining approaches based on sprinkling and AS, and similarity mining approaches based on higher order associations, do not make any assumptions on the nature of Information Entities. As for the integration of background knowledge, it will be interesting to use sprinkled terms to model knowledge outside class knowledge. In web mining, meta-tags often carry important information about web-pages, that can be used to bias clustering of web-pages, in a way not very different from how class labels were used by sprinkling. In such an application, sprinkling can mean injecting meaningful terms, instead of artificial ones. Another idea is to sprinkle additional cases, instead of features as has been done in our work, that use forced co-occurrences to capture background knowledge of similarities between features. The possibility of augmenting the labeled cases with unlabelled training data where the sprinkled terms show no class affiliations may open up interesting avenues for applying semi-supervised approaches to acquiring relevance knowledge. This is particularly interesting from a practical standpoint, since unlabelled cases are often more readily available than labelled ones.

Beyond LSI. In this thesis, we have focused on how sprinkling can be used to incorporate class knowledge into LSI to improve classification effectiveness. However, the general idea of sprinkling has the potential to make significant improvements to other concept learners as well. Probabilistic models like PLSI can benefit by using revised probability estimates obtained from sprinkled representations. Association rule mining approaches, distributional clustering and FCA can all be potentially extended to operate over sprinkled representations to bias their inferred knowledge by drawing together features representative of the same class. We have shown that SVM benefits from LSI generated representations obtained using sprinkling. Demonstrating the applicability of the basic idea of sprinkling to improve the classification effectiveness of popular concept induction approaches has the potential of extending the impact of our work beyond TCBR to the broader field of Machine Learning.

Visualization for Maintenance. We have used visualization based on the idea of stacking to obtain qualitative insights into the nature of textual casebases. However, given the advantages of the stacking approach relative to other visualization approaches as summarized in Section 9.1, the work can have interesting implications for casebase maintenance. Visualization can help us identify redundant and noisy cases and features, and thus facilitate casebase editing leading to improvement in retrieval effectiveness; this has been demonstrated in structured CBR by Massie (2006). However, to enhance usability, the visualization tool needs to undergo several changes. Firstly, it should be interactive with abilities to zoom into specific regions of the stacked image, facilitate flexible changes to the casebase and allow for restacking after incorporating the changes made. Secondly, instead of just displaying a list of words associated with each topic chunk, it should display the associations between features in such topic chunks. One idea is to integrate the facility for displaying association graphs that show higher order links between words (as shown in Section 6.6) into the stacked image, and invoke it whenever the user requests a zoom-in on any region of the image. Since the stacking process ensures that neighbouring features in the image are similar in their co-occurrence patterns, the association graph is expected to provide more insight into the nature of their associations. A final point is that we need consolidated user studies to evaluate the usefulness of our visualization approach with respect to other approaches in real world TCBR tasks. We have not emphasized this aspect enough in this thesis, primarily because visualization-driven maintenance was peripheral to the central theme of our work.

Evaluation over unsupervised collections. While sprinkling is devised to operate specifically over supervised collections, higher order association mining approaches make no assumption of class knowledge. However, all our evaluations have been carried out over classification datasets. This was mainly because it is difficult to obtain unsupervised collections on which human relevance judgements are available on topics (queries). In the IR community, TREC and MUC have provided platforms for creating such evaluation datasets and allowing them to be used by the community to benchmark their performance results. It will be important for TCBR researchers to create similar evaluation datasets on select domains, so that the goodness of the automatically acquired knowledge can be verified against expert judgements. This will also facilitate fresh research on how human experts and automated learners can collaborate and complement each other in the process of knowledge discovery and elicitation. The non-availability of unsupervised collections also meant that the original GAME measure could not be experimentally evaluated, though results on its supervised counterpart $\text{GAME}_{\text{class}}$ were reported. In the case of complexity evaluation, however, the lack of benchmark datasets can perhaps be circumvented by verifying alignment of textual review reports (treated as problems) to review ratings (treated as solutions) in product review domains; while it may be argued that this is no different from classification domains that we have evaluated, we feel that the wider range of review ratings can effectively map onto, and simulate a larger solution vocabulary.

Dynamic Knowledge. Textual casebases, not unlike structured casebases, change over time. This means that the associated knowledge containers, namely the relevance and similarity knowledge, need to be updated to reflect these changes. Several approaches have been investigated by the LSI research community to speed up the process of updating LSI representations when changes are made to document collections. Three such approaches were briefly presented in Chapter 3. We note that these approaches directly apply to sprinkled LSI and AS-based LSI as well. However, more research needs to go into speeding up updates to similarity knowledge acquired using higher order association mining. Our current prescription is a lazy strategy that makes quick incremental but approximate changes whenever a change happens, and relegates the job of making accurate changes at a later "bulk update" stage. A similar idea was also presented in the context of making efficient updates to the stacked image for visualization. We have presented an algorithm to make fast updates to FCRN whenever changes are made to the casebase, by eliminating some of the redundant computations involved in computing effective relevance values from scratch.

In addition to the broad areas identified above, there are avenues for extending and fine-tuning the approaches presented in this thesis. We have made mentions of such possibilities in context when presenting these approaches. Examples include better heuristics for selecting the Maximum Sprinkling Length (MSL), faster evolutionary strategies for learning parameters in the similarity mining approach based on higher order associations, and efficient ways of selecting best starting cases for the stacking algorithm.

9.3 Closing Notes

This thesis is positioned at the confluence of two significant problems of topical interest. The first concerns the knowledge engineering bottleneck that has plagued real-world AI systems over the last few decades. The second is the problem of making sense of huge volumes of unstructured data to address diverse information needs of users. Implicit in the statement of this second problem is the need to attain a reasonable tradeoff between the twofold criteria of ensuring retrieval effectiveness and efficiency. As this thesis is being written, both problems mentioned above are holy grails of computing science, though a significant volume of research deals with downsized version of these problems, in that they narrow down their scope to realizing realistic targets in meeting application-specific requirements. CBR, not unlike other AI approaches, needs knowledge acquisition to populate its knowledge containers. IR needs to handle large volumes of unstructured texts. TCBR, however, needs to tackle both problems. This is because TCBR strives to a middle ground between CBR and IR, in that it aims at improving retrieval effectiveness of IR by using sophisticated domain-specific knowledge, and it extends CBR by relaxing the need to have structured representations for cases. In this thesis we have attempted a comprehensive study of approaches that address the aforementioned two problems of contemporary interest in the context of TCBR. However, because of the general nature of the problems, the ideas, techniques and formalisms presented are expected to be of interest to a much wider research community from the fields of CBR, IR, Text Mining, Machine Learning and Information Visualization.

We have shown that statistical learning approaches can be effective in reducing manual knowledge engineering overheads associated with acquiring knowledge containers for TCBR. Novel extensions of LSI were presented to handle supervised classification domains, and we have shown the idea can be extended to model complex class relationships in hierarchical and ordinal domains, while taking into account the fact that certain classes are easier to separate than others. We have presented a novel algorithm to mine similarity between features based on their higher order associations. One significant contribution of our work is in demonstrating that, when equipped with knowledge that is statistically mined using the proposed approaches, instance based approaches can outperform state-of-the-art machine learning approaches like SVM. This is of notable interest to the TCBR community, especially in the light of several advantages associated with instance based learners like support for lazy incremental updates and explicitness of knowledge allowing for good explanation, visualization and ability to accommodate expert-initiated feedback. Though experimental evaluations were carried out over supervised datasets, unsupervised TCBR systems can benefit from the rich representations as well. To address efficiency issues, we presented a novel retrieval formalism. It is interesting to note that while CRNs are good at handling a large number of cases, FCRN is specifically designed to improve CRN time performance by being able to handle high dimensional non-sparse representations efficiently. The curse of dimensionality (Russell & Norvig 2003) has been a challenge for scale up of real world systems, and FCRNs provide a practical approach to speed up retrieval while making best use of the rich relevance and similarity knowledge, which are critical to the effectiveness of TCBR systems. Our contributions on the visualization and complexity front have been under-exploited in this thesis, in that we have restricted our attention to using these techniques for explaining our empirical results. The bigger application context, however, is maintenance of TCBR knowledge containers. There are not many works till date in this area, and we hope our contributions will lead to more active interest in the research community at effectively bridging the knowledge gap between the expert and the system.

Bibliography

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications 7(1)*, (pp. 39–59).

Austin, J. (1962). *How to do Things with Words* . Oxford University Press.

Baker, L., & McCallum, A. (1998). Distributional clustering of words for text classification. *Proceedings of SIGIR-98* (pp. 96-103). ACM Press.

Balaraman, V., & Chakraborti, S. (2004). Satisfying Varying Retrieval Requirements in Case-Based Intelligent Directory Assistance. *Proc. of the FLAIRS Conference* .

Bergmann, R. (2002). Experience Retrieval. In *Experience Management : Foundations, Development Methodology, and Internet-Based Applications* (pp. 417-426).

Berry, M., Dumais, S., & O'Brien, G. (1995). Using linear algebra for intelligent information retrieval. *SIAM Rev., 37*, (pp. 573-595).

Blumberg, R., & Atre, S. (2003). The Problem with Unstructured Data. *DM review magazine Feb 2003.*

Breimen, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees.*

Brodley, C., & Utgoff., P. (1995). Multivariate decision trees. *Machine Learning, 19* , 45-77.

Brown, M., Förtsch, C., & Wissmann, D. (1998). Combining IR and CBR for "Middle Ground" Text Retrieval Problems. *Papers from the AAAI-98 Workshop on Textual Case-Based Reasoning*, (p. 3).

Brüninghaus, S., & Ashley, K. (2001). The Role of Information Extraction for Textual CBR. *Proc. of 4th ICCBR* (pp. 74-89). Springer.

Budanitsky, A. (1999). Lexical semantic relatedness and its application in natural language processing. *Technical Report CSRG390.* University of Toronto.

Chakraborti, S., Ambati, S., Balaraman, V., & Khemani, D. (2003). Integrating Knowledge Sources and Acquiring Vocabulary for Textual CBR. *Proc. of the 8th UK CBR Workshop* , (pp. 74-84).

Chakraborti, S., Beresi, U., Wiratunga, N., Massie, S., Lothian, R., & Watt, S. (2007). A Simple Approach towards Visualizing and Evaluating Complexity of Textual Case Bases. *To appear in the Proc. of the fourth workshop on TCBR (ICCBR 07).*

Chakraborti, S., Lothian, R., Wiratunga, N., & Watt, S. (2006). Sprinkling: Supervised Latent Semantic Indexing. *Procs. of 28th European Conf. on Information Retrieval* , (pp. 510-515).

Chakraborti, S., Lothian, R., Wiratunga, N., Orecchioni, A., & Watt, S. (2006). Fast Case Retrieval Nets for Textual Data. *Proc. of the 8th European Conference on Case-Based Reasoning (ECCBR-06)* (pp. 400-414). Springer.

Chakraborti, S., Mukras, R., Lothian, R., Wiratunga, N., Watt, S., & Harper, D. (2007). Supervised Latent Semantic Indexing using Adaptive Sprinkling. *Proc. of the Twentieth IJCAI Conference* , (pp. 1582-7).

Chakraborti, S., Watt, S., & Wiratunga, N. (2004). Introspective Knowledge Acquisition in Case Retrieval Networks for Textual CBR. *Proc. of the 9th UK CBR Workshop*, (pp. 51-61).

Chakraborti, S., Wiratunga, N., Lothian, R., & Watt, S. (2007). Acquiring Word Similarities with Higher Order Association Mining. . *To appear in the Proc. of the 7th International CBR Conference (ICCBR'07).* Springer.

Chakraborti, S., Wiratunga, N., Lothian, R., & Watt, S. (2005). Fast Case Retrieval Nets for Textual CBR. *Procs. of the 10th UK CBR Workshop* .

Chavez, E., Navarro, G., Baeza-Yates, R., & Marroquin, J. (2001). Searching in metric spaces. *ACM Computing Surveys Vol. 33 Issue 3* , 273-321.

Chin, K. (1998). *Support Vector Machines applied to Speech Pattern Classification.* University of Cambridge.

Cimiano, P., Staab, S., & Tane, J. (2003). Automatic Acquisition of Taxonomies from Text: FCA meets NLP. *In Proceedings of the ECML/PKDD Workshop on Adaptive Text Extraction and Mining*, (pp. 10-17).

Cohen, W., & Singer, Y. (1999). Context-sensitive learning methods for text categorization. *ACM Trans. Inf. Syst. 17(2)*, (pp. 141-173).

Deerwester, S., Dumais, S., Furnas, G., Landuer, T., & Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science, vol. 41, no.6*, 391-407.

Delany, S., & Bridge, D. (2006). Feature-Based and Feature-Free Textual CBR A Comparison in Spam Filtering. *Procs. of Irish Conference on AI and Cognitive Science*, (pp. 244-253).

Delany, S., & Cunningham, P. (2004). An Analysis of Case-base Editing in a Spam Filtering System . *Proc. of the 7th ECCBR* (pp. 128-141). Springer.

Delany, S., Cunningham, P., Tsymbal, A., & Coyle, L. (2004). A Case-based Technique for Tracking Concept Drift in Spam Filtering. *Applications and Innovations in Intelligent Systems XII, Procs. of AI 2004*, (pp. 3-16). Springer.

Díaz-Agudo, B., & González-Calero, P. (2001). Formal concept analysis as a support technique for CBR. *Knowledge Based Syst. 14(3-4)*, 163-171.

Drucker, H., Wu, D., & Vapnik, V. (1999). Support Vector Machines for Spam Categorization . *IEEE Tans. on Neural Networks, 10*, (pp. 1048-1054).

du Boucher-Ryan, P., & Bridge, D. (2006). Collaborative Recommending using Formal Concept Analysis. *Knowledge-Based Systems, vol.19(5)*, (pp. 309-315).

Dumais, S. T. (1993). LSI meets TREC: A status report. The First Text REtrieval Conference (TREC1). *D. Harman (Ed.), National Institute of Standards and Technology Special Publication 500-207*, (pp. 137-152).

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization . *Proc. of 7th International Conf. on Information and Knowledge Management.*

Eades, P. (1984). A heuristic for graph drawing. *Congressus Numerantium 42*, (pp. 149–160).

Edmonds, P. (1997). Choosing the word most typical in context using a lexical co-occurrence network . *Meeting of the Association for Computational Linguistics*, (pp. 507-509).

Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook.* Cambridge University Press.

Frank, E., Holmes, G., Kirkby, R., & Hall, M. (2002). Racing committees for large datasets. *Discovery Science, 2002.*

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *Ann. Stat.*, (pp. 337-407).

Furnas, G. W., K., L. T., Gomez, L. M., & Dumais, S. T. (1987). The Vocabulary problem in Human- System Communication. *Communications of the ACM, 30,11*, (pp. 964-971).

G.-R., X., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., et al. (2005). Scalable Collaborative Filtering Using Cluster-based Smoothing. *Procs. of the 28th ACM SIGIR Conference* , (pp. 114-121).

Gabrilovich, E., & Markovitch, S. (2007). Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. *Proceedings of 20th IJCAI conference.*

Gabrilovich, E., & Markovitch, S. (2005). Feature Generation for Text Categorization using World Knowledge. *Procs. of IJCAI 2005*, (pp. 1048-1053).

Gee, K. (2003). Using latent Semantic Indexing to Filter Spam . *Proc. of the 2003 ACM Symposium on Applied Computing*, (pp. 460-464).

Gervás, P., Díaz-Agudo, B., Peinado, F., & Hervás, R. (2005). Story plot generation based on CBR. *Knowl.-Based Syst. 18(4-5)* , 235-242.

Grishman, R. (1997). Information Extraction: Techniques and Challenges. *Procs. of SCIE'97*, (pp. 10-27).

Gupta, K., & Aha, D. (2004). Towards acquiring case indexing taxonomies from text. *Barr, V and Zdravko, M (eds) Proceedings of the Seventeenth Annual Conference of* *the International Florida Artificial Intelligence Research Society* (pp. 172-177). AAAI Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). The Elements of Statistical Learning. Springer-Verlag .

Hayes, P. J., Andersen, P. M., Nirenburg, I. B., & Schmandt, L. M. (1990). TCS : A shell for content-based text categorization. *In Procs of CAIA-90, 6th IEEE Conference on AI Applications*, (pp. 320-326).

Hilario, M., & Kalousis, A. (1999). *Characterizing Learning Models and Algorithms for Classification.* CUI - University of Geneva.

Hoenkamp, E. (2003). Unitary Operators on the Document Space. *JASIST 54(4)*, (pp. 321-334).

Hoffman, T. (1999). Probabilistic Latent Semantic Analysis. *Proc. of Uncertainty in Artificial Intelligence, UAI'99.*

Inselberg, A. (1985). The plane with parallel coordinates. *The Visual Computer 1*, (pp. 69–91).

Jarmasz, M., & Szpakowicz, S. (2003). Roget's thesaurus and semantic similarity. . *Procs of the International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, (pp. 212-219). Borovets, Bulgaria.

Joachims, T. (1998). Text Categorization with Support Vector Machines Learning with Many Relevant Features. *Proc. of ECML* (pp. 137-142). ACM Press.

Kolda, T., & O'Leary, D. (1998). A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Transactions on Information Systems, vol 16, issue 4* , 322-346.

Kolodner, J. (1993). *Case-Based Reasoning.* San Mateo: Morgan Kaufmann.

Kontostathis, A., & Pottenger, W. (2006). A framework for understanding LSI performance. . *Information Processing and Management. Volume 42, number 1,*, (pp. 56-73).

Kunze, M., & Hübner, A. (1998). Textual CBR – Case Studies of Projects Performed. *Papers from the AAAI-98 Workshop on Textual Case-Based Reasoning*, (p. 40).

Kushmerick, N., Johnston, E., & McGuinness, S. (2001). . *Proc. of IJCAI'01 Workshop on Adaptive Text Extraction and Mining.*

Lamontagne, L. (2006). Textual CBR Authoring using Case Cohesion. *Proceedings of the ECCBR'06 Workshops, TCBR'06 - Reasoning with Text,* (pp. 33-43).

Lang, K. (1995). NewsWeeder: learning to filter netnews. *Procs. of ICML'95,* (pp. 331-339).

Lee, D., & Seung, H. (2001). Algorithms for non-negative matrix factorization. . *In Advances in Neural Information Processing Systems, vol. 13,* (pp. 556-562).

Lemaire, B., & Denhière, G. (2006). Effects of High-Order Co-occurrences on Word Semantic Similarity. *Current Psychology Letters, 18, Vol. 1.*

Lenz, M. (1996). Case Retrieval Nets Applied to Large Case-Bases. *Proc. 4th German Workshop on CBR.* Informatik Preprints, Humboldt-Universität zu Berlin.

Lenz, M. (1999). *Case Retrieval Nets as a Model for Building Flexible Information Systems, PhD dissertation.* Humboldt Uni. Berlin. Faculty of Mathematics and Natural Sciences.

Lenz, M. (1998). Knowledge Sources for Textual CBR Applications. *Textual CBR Papers from the 1998 Workshop Technical Report WS-98-12* (pp. 24-29). AAAI Press.

Lenz, M., & Burkhard, H. (1996). *Case Retrieval Nets: Foundations, Properties, Implementation, and Results, Technical Report.* Humboldt-Universität zu Berlin.

Lenz, M., & Burkhard, H.-D. (1996). Case Retrieval Nets Basic Ideas and Extensions. *KI*, (pp. 227-239).

Lenz, M., & Burkhard, H.-D. (1997). CBR for Document Retrieval - The FAllQ Project. *Case-Based Reasoning Research and Development, Springer Verlag, LNAI 1266,* (pp. 84-93).

Lenz, M., Auriol, E., & Manago, M. (1998). Diagnosis and Decision Support (Chapter 3). *Case-Based Reasoning Technology, Lecture Notes in Artificial Intelligence 1400,* (pp. 51-90).

Lenz, M., Hubner, A., & Kunje, M. (1998). Textual CBR (Chapter 5). *Case-Based Reasoning Technology, Lecture Notes in Artificial Intelligence 1400,* (pp. 115-137).

Lopez De Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., et al. (2005). Retrieval, reuse, revision and retention in case-based reasoning. *Knowl. Eng. Rev. 20:3* , 215-240.

Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research, Methods, Instruments and Computers, 28-2* , (pp. 203-208).

Lytinen, S., & Tomuro, N. (2002). The Use of Question Types to Match Questions in FAQFinder, Mining Answers From Texts and Knowledge Bases. *AAAI Technical Report SS-02-06* (pp. 46-53). AAAI Press.

Mann, W., & Thompson, S. (1987). *Rhetorical Structure Theory: A Theory of Text Organization.* Tech Rep. ISI/RS-87-190, USC Information Sciences Institute.

Manning, C., & Schütze, H. (1999). Foundations of Statistical Natural Language Processing. Cambridge, MA: MIT Press.

Manning, C., Raghavan, P., & Schütze, H. (2008 (expected)). Introduction to Information Retrieval,. Cambridge University Press.

Mao, W., & Chu, W. W. (2007). The phrase-based vector space model for automatic retrieval of free-text medical documents. *Data and Knowledge Engineering Vol. 61, Issue 1* , 76-92.

Massie, S. (2006). *Complexity Modelling for Case Knowledge Maintenance in Case Based Reasoning, PhD Thesis.* The Robert Gordon University.

Massie, S., Wiratunga, N., Donati, A., & Vicari, E. (2007). From Anomaly Reports to Cases. *To appear in the Procs. of the 7th International CBR Conference (ICCBR'07).* Springer .

Mill, W., & Kontostathis, A. (2004). *Analysis of the values in the LSI term-term matrix, Technical report.* Ursinus College .

Miller, G. (1995). WORDNET: A Lexical Database for English. *Communications of the ACM 38(11),* (pp. 39-41).

Mitchell, T. (1997). *Machine Learning.* Mc Graw Hill International .

Mori, J., Ishizuka, M., & Matsuo, Y. (2007). Extracting Keyphrases To Represent Relations in Social Networks from Web. *Proc. of the Twentieth IJCAI Conference,* (pp. 2820-2825).

MUC. (1991). *Procs of the Sixth Message Understanding Conference (MUC-6).* Morgan Kaufmann.

Mukras, R., Wiratunga, N., Lothian, R., Chakraborti, S., & Harper, D. (2007). Information Gain Feature Selection for Ordinal Text Classification using Probability Re-distribution. *Proc. of IJCAI Textlink Workshop.*

Mullins, M., & Smyth, B. (. (2001). Visualisation methods in case-based reasoning. *Workshop proceedings of the fourth ICBR.*

Nahm, U., & Mooney, R. (2001). Mining Soft-Matching Rules from Textual Data. *Procs. of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01),*, (pp. 979-984).

Orecchioni, A., Wiratunga, N., Massie, S., Chakraborti, S., & Mukras, R. (2007). Learning Incident Causes. *Proc. of the TCBR Workshop 2007 (ICCBR).*

Pang, B., & Lee, L. (2005). Seeing Stars: Rxploiting Class Relationships for Sentiment Categorization with respect to Rating Scales . *Procs. of ACL 2005,* (pp. 115-124).

Panurgy. (2006, August). *Panurgy Technology News.* Retrieved from http://www.chesapeake.panurgy.com/Newsletter/2006-August.htm

Patterson, D., Dobrynin, V., Galushka, M., & Rooney, N. (2005). Sophia : A novel approach for textual case-based reasoning. *Proc. of the Nineteenth IJCAI Conference,* (pp. 1146-1153).

Pedersen, T., Patwardhan, S., & Michelizzi, J. (2004). WordNet: : Similarity - Measuring the Relatedness of Concepts. *Procs. of AAAI conf.,* (pp. 1024-1025).

Porter, M. (1980). An algorithm for suffix stripping. *Program, 14(3)* , 130-137.

Reuters. (1997). *Reuters-21578 Text Classification corpus.* Retrieved from daviddlewis.com/resources/testcollections/reuters21578/

Richter, M. M. (1998). Chapter: Introduction. *Lenz, M.; Bartsch-Sprl, B.; Burkhard, H.-D.; and Wess, S., eds., Case-Based Reasoning Technology: From Foundations to Applications, Lecture Notes in Artificial Intelligence 1400.* Springer Verlag.

Rijsbergen, C. J. (1979). *Information Retrieval 2nd edition.* Butterworths.

Riloff, E. (1993). Automatically Constructing a Dictionary for Information Extraction Tasks. *Proceedings of the Eleventh National Conference on Artificial Intelligence* (pp. 811-816). Menlo Park, CA. : AAAI Press/MIT Press.

Riloff, E., & Lehnert, W. (1994). Information Extraction as a Basis for High-Precision Text Classification. *ACM Transactions on Information Systems* , 296-333.

Rosenfeld, A., & Kak, A. (1982). *Digital Picture Processing.* New York : Academic Press.

Rumelhart, D., & McClelland, J. (1986). Parallel distributed Processing Explorations in the Microstructure of Cognition. Volume 1. Foundations. Cambridge: MIT Press.

Russell, S., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach.* Pearson US.

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C., & Stamatopoulos, P. (2003). A Memory-based Approach to Anti-Spam Filtering for Mailing Lists. *Information Retrieval, 6* , 49–73.

Schaaf, J. (1995). "Fish and Sink" An Anytime Algorithm to Retrieve Adequate Cases. *Case-Based Reasoning Research and Development (Proc. of International Conference on CBR 1995)* (pp. 371-380). Springer.

Scott, S., & Matwin, S. I.-3. (1999). Feature Engineering for Text Classification. *Proc. of ICML*, (pp. 379-388).

Sebastiani, F. (2002). Machine learning in automated text categorization. . *ACM Computing Surveys 34(1)*, (pp. 1-47).

Semeraro, G., Lops, P., & Degemmis, M. (2005). WordNet-based User Profiles for Neighborhood Formation in Hybrid Recommender Systems. *Procs. of Fifth HIS Conference* , (pp. 291-296).

Smeaton, A., Keogh, G., Gurrin, C., McDonald, K., & Sødring, T. (2002). Analysis of papers from twenty-five years of SIGIR conferences: what have we been doing for the last quarter of a century? *ACM SIGIR Forum ACM SIGIR Forum Volume 36 , Issue 2 (Fall 2002).*

Smyth, B., & McKenna, E. (1999). Footprint Based Retrieval. *Proc. of 3rd ICCBR*, (pp. 343-357).

Stanfill, C., & Waltz, D. (1986). Toward Memory-Based Reasoning. *Communications of the ACM 29, 12,* , 1213-1228.

Sun, J., Chen, Z., Zeng, H., Lu, Y., Shi, C., & Ma, W. (2004). Supervised Latent Semantic Indexing for Document Categorization . *Proc. of ICDM* (pp. 535-538). IEEE Press.

Terra, E., & Clarke, C. (2003). Frequency Estimates for Word Similarity Measures. *Proceedings of HLT-NAACL 2003, Main Papers*, (pp. 165-171).

Utgoff, P. (1989). Incremental induction of decision trees. *Machine Learning, 4* , 161-186.

Vinay, V., Cox, I., Milic-Fralyling, N., & Wood, K. (2006). Measuring the Complexity of a Collection of Documents. *Proc. of 28th ECIR*, (pp. 107 - 118).

Wang, M., Nie, J., & Zeng, X. (2005). A Latent Semantic Classification Model. *Proc. of 14th ACM CIKM*, (pp. 261-262).

Weber, R., Ashley, K., & Bruninghaus, S. (2006). Textual Case-based reasoning. *Knowledge Engineering Review, Vol. 20:3* , 255-260.

Weß, S., Althoff, K., & Derwand, G. (1994). Using k-d trees to Improve the Retrieval Step in Case-Based Reasoning,. *Topics in Case-Based Reasoning, Proc. of European Workshop on CBR-93* (pp. 167-181). Springer.

Wess, S., Althoff, K.-D., & Derwand, G. (1993). Using k -d Trees to Improve the Retrieval Step in Case-Based Reasoning. *Proc. of EWCBR*, (pp. 167-181).

Wiener, E., Pederson, J., & Weigend, A. (1995). A Neural Network Approach to Topic Spotting. *Proc. of SDAIR'95*, (pp. 317-332).

Wille, R. (1982). Restructuring lattice theory: an Approach based on Hierarchies of Concepts. *I. Rival (Ed.), Ordered sets. Reidel, Dordrecht-Boston* , 445-470.

Wilson, D., & Bradshaw, S. (1999). CBR Textuality. *Proc. of the Fourth UK Case-Based Reasoning Workshop*, (pp. 67-80).

Wiratunga, N., Lothian, R., Chakraborti, S., & Koychev, I. (2005). A Propositional Approach to Textual Case Indexing. *Proc. of The PKDD Conference, Springer* , (pp. 380-391).

Wiratunga, N., Lothian, R., Chakraborti, S., & Koychev, I. (2005). Textual feature construction from keywords. *Weber, R and Branting, LK (eds) Proceedings of the Textual Case-Based Reasoning Workshop*, (pp. 110-119). Chicago.

Wiratunga, N., Massie, S., & Lothian, R. (2006). Unsupervised Textual Feature Selection. *Proc. of the 8th ECCBR Conference* (pp. 340-354). Springer.

Wolverton, M. (1995). An Investigation of Marker Passing Algorithms for Analogue Retrieval. *Case-Based Reasoning Research and Development (Proc. of International Conference on CBR 1995), LNAI 1010* (pp. 359-370). Springer.

Wolverton, M., & Hayes-Roth, B. (1994). Retrieving Semantically Distant Analogies with Knowledge-Directed Spreading Activation. *Proc. of AAAI -94* .

Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., et al. (2005). Scalable Collaborative Filtering using Cluster-based Smoothing. *Procs. of 28th ACM SIGIR*, (pp. 114-121).

Yang, Y., & Pederson, J. (1997). A Comparative Study on Feature Selection in Text Categorization. *Proc. of the International Conference on Machine Learning*, (pp. 412-420).

Zelikovitz, S. (2003). Mining for features to improve classification. *Proc of Machine Learning, Technologies and Applications* .

Zelikovitz, S. (2004). Transductive LSI for Short Text Classification Problems. *Procs of FLAIRS Conference 2004.*

Zelikovitz, S., & Hirsh, H. (2001). Using LSI for Text Classification in the Presence of Background Text. *Proc. of CIKM 2001*, (pp. 113-118).

Appendix A1

Realizing Textual Similarity Measures using Case Retrieval Networks

During retrieval, the ranking of cases depends on the distance metric (or similarity measure) realized by the CRN. Two measures commonly used with textual data are the cosine measure and the Euclidean distance. The original work of Lenz (1999) prescribes no method to realize these measures. Here we propose simple extensions to the CRN towards this end, which are used in our implementations in the thesis.

The cosine measure has been popular in Information Retrieval applications (Rijsbergen 1997) where the query is typically much shorter than the texts (cases). When the cases and queries are treated as vectors in a feature space, the cosine similarity depends only on the angle between the vectors and not their lengths. Several textual CBR systems also use the cosine measure in a retrieval scenario where users are expected to type in only a few words, based on which relevance of cases needs to be estimated. An example is FAQFinder (Lenz 1998a). The Euclidean distance metric is more commonly used in CBR, but it needs the query and the case to be compatible. This is true in classification applications and in case competence modeling, where cases within the same casebase are compared with each other.

Let us consider a simplistic situation where each case (as well as the query) is represented by a binary-valued feature vector whose elements correspond to the presence or absence of an IE. We assume no knowledge of similarity between words.

Let the aggregation function at each case node be modelled by a simple addition of incoming relevance activations.

Given a query Q, activations are propagated via similarity arcs. The incoming activations are aggregated at each IE node, and we obtain a revised query Q^*. It is evident that the activation of case node C_i is given by the dot product of the revised query Q^* and the case vectors, where each case vector is composed of the relevance values relating the IEs to that case.

Given this observation, extending the CRN to realize the cosine similarity is straightforward. We note that the cosine similarity between Q^* and C_i, $\cos(Q^*,C_i)$ is related to the dot product (Q^*,C_i) by the following equation:

$$\cos(Q^*,C_i) = Q^*.C_i /(\|Q^*\| \cdot \|C_i\|) \qquad \text{(A1.1)}$$

where $\|Q^*\|$ and $\|C_i\|$ are vector norms. $\|C_i\|$ can be pre-computed for all cases in the casebase, while $\|Q^*\|$ needs to be evaluated during retrieval. We need a post-processing phase to compute the cosine similarity using the dot product and the case and query norms. Figure A.1 shows the schematic of a CRN to realize this.

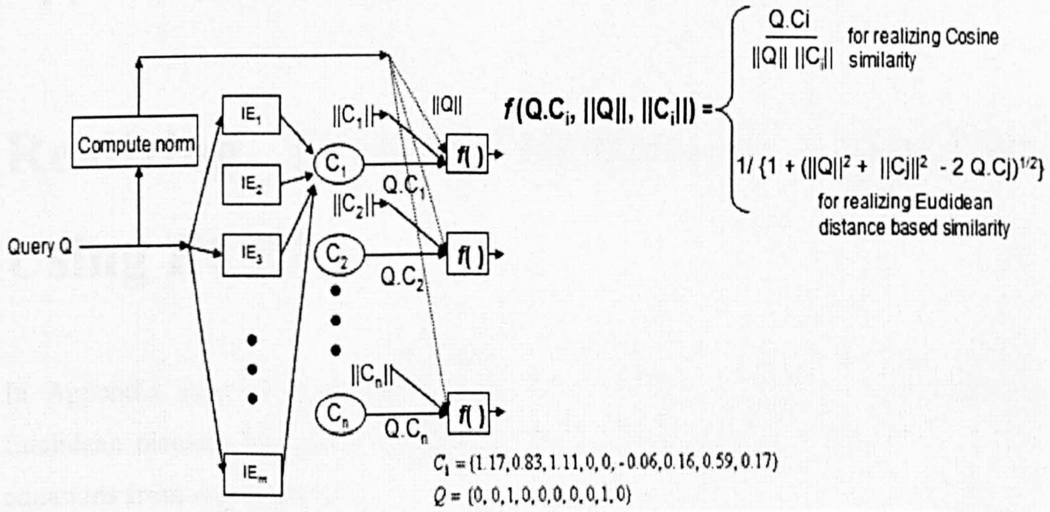
Extending the CRN to rank cases according to the Euclidean distance metric appears to be trickier, since IEs not activated by the query case also take part in the matching process. A workaround is to rewrite the Euclidean distance between the case and the query $ED(Q^*,C_i)$ in terms of the dot product and the case and query norms as:

$$ED(Q^*,C_i) = \{\|Q^*\|^2 + \|C_i\|^2 - 2(Q^*.C_i)\}^{1/2} \qquad \text{(A1.2)}$$

As with evaluating cosine measure, $\|C_i\|$ can be pre-computed for each case in the casebase and $\|Q^*\|$ is evaluated at run time. $Q^*.C_i$ is the dot product evaluated by the CRN. In our experiments, we use the following formulation to obtain a similarity measure based on Euclidean Distance:

$$sim(Q^*,C_i) = 1/\{1 + ED(Q^*,C_i)\} \qquad \text{(A1.3)}$$

Figure A1.1 shows the post-processing needed for realizing Euclidean distance-based similarity. An example distance calculation is also shown based on the relevance values shown in Figure 4.2. We assume no similarity between IEs in the example.


Query Q
Compute norm
IE1
IE2
IE3
IEm
C1
C2
Cn
	C1	
	C2	
	Cn	
	Q	
Q.C1
Q.C2
Q.Cn
f( )
f(Q.Ci, ||Q||, ||Ci||) =
Q.Ci / ||Q|| ||Ci|| for realizing Cosine similarity
1/ {1 + (||Q||² + ||Cj||² - 2 Q.Cj)^1/2} for realizing Euclidean distance based similarity

$C_1 = \{1.17, 0.83, 1.11, 0, 0, -0.06, 0.16, 0.59, 0.17\}$

$Q = \{0, 0, 1, 0, 0, 0, 0, 0, 1, 0\}$

$\| C_1 \| = \sqrt{1.17^2 + 0.83^2 + 1.11^2 + (-0.06)^2 + 0.16^2 + 0.59^2 + 0.17^2} = 1.9225$

$\| Q \| = \sqrt{1^2 + 1^2} = 1.4142$

$Q.C_1 = (1.11 \times 1) + (0.59 \times 1) = 1.7$

$\cos(Q, C_1) = Q.C_1 / (\| Q \| . \| C_1 \|) = 0.6253$

$ED(Q, C_1) = \{\| Q \|^2 + \| C_1 \|^2 - 2(Q.C_1)\}^{1/2} = 1.5152$

Figure A1.1 Extending the CRN to realize Euclidean and cosine measures

Appendix A2

Realizing Textual Similarity Measures Using FCRNs

In Appendix A1, we have shown how the CRN can be extended to realize the Euclidean distance and cosine similarity measures. We reproduce the following two equations from Appendix A1:

$$\cos(Q^*, C_i) = Q^* . C_i / (\| Q^* \| \cdot \| C_i \|) \qquad \text{(A2.1)}$$

$$ED(Q^*, C_i) = \{\| Q^* \|^2 + \| C_i \|^2 - 2(Q^* . C_i)\}^{1/2} \qquad \text{(A2.2)}$$

While the same idea can be extended to FCRNs as well, a few differences are worth noting. Firstly, since the similarity propagation phase is absent in the FCRN, we no longer have access to the revised query Q^*. Secondly, the case vector in the FCRN is composed of the effective relevances of the IEs to the case, so IEs not present in a case may also have non-zero effective relevance. This is because the similarity knowledge between IEs is taken into account while computing effective relevances. We denote the case vector as C^*, which is a revised version of the original case vector C. Thus, while a CRN uses the similarity knowledge to revise the query at run-time, the FCRN uses a pre-computation step that uses the similarity knowledge to revise the cases in the casebase.

Though the cosine and Euclidean measures produced by the FCRN are different from those produced by the CRN, it follows from the equivalence shown in Section 8.2

that the dot-product case activations produced by CRN and FCRN are identical. Thus $Q^*.C_i = Q.C_i^*$. This property may be used to rank cases in FCRN in the same order as in a CRN, with respect to both measures. Using equation (A2.1), the ratio of cosine measures of cases C_1 and C_2 with respect to query Q^* is given by

$$\frac{\cos(Q^*,C_1)}{\cos(Q^*,C_2)} = \frac{Q^*.C_1}{Q^*.C_2} \times \frac{\| C_2 \|}{\| C_1 \|} = \frac{Q.C_1^*}{Q.C_2^*} \times \frac{\| C_2 \|}{\| C_1 \|} \tag{A2.3}$$

We note that this ratio is independent of $\| Q^* \|$. The terms $Q.C_1^*$ and $Q.C_2^*$ are outputs of the FCRN, while $\| C_1 \|$ and $\| C_2 \|$ can be pre-computed. Thus we can evaluate cosine measures for all cases relative to the first case, and produce a ranking that is equivalent to that produced by the CRN.

We can also use a FCRN to generate the same ranking as in CRN, with respect to the Euclidean distance measure. Using equation (A2.2), the difference between squares of Euclidean distances of the query Q^* to cases C_1 and C_2 is given by

$$\begin{aligned} ED^2(Q^*,C_1) - ED^2(Q^*,C_2) &= \| C_1 \|^2 - 2(Q^*.C_1) - \| C_2 \|^2 + 2(Q^*.C_2) \\ &= \| C_1 \|^2 - 2(Q.C_1^*) - \| C_2 \|^2 + 2(Q.C_2^*) \end{aligned} \tag{A2.4}$$

We note that this difference is independent of $\| Q^* \|$, and all terms on the right hand side are available to the FCRN either at its output or from pre-computation. Thus we can evaluate the squares of Euclidean distances for all cases in relation to the first case, and produce a ranking of cases equivalent to that generated by the CRN.

Appendix A3

The Extended Case Retrieval Network (ECRN) : Additional Results

The ECRN was briefly presented in Section 3.2, and shown to yield effectiveness improvements comparable to, and occasionally outperforming the best off-the-shelf classifiers. In this appendix, we take a closer look at some other interesting aspects of ECRN.

A3.1 Training Time Reduction with VSM based Weight Initialization

In Section 3.2, we noted that the ECRN was different from a traditional neural network in that we can ascribe meaning to its nodes and weights in terms of cases and relevance values. This also facilitates instantiation of the ECRN network with LSI-mined weights during training, instead of an arbitrary instantiation with low values as is typical with neural networks. Fig. A3.1 shows that this can lead to conspicuous improvement in training times in one of the problem domains (LINGSPAM). We observe that the convergence is much faster when domain knowledge in the form of binary weights from the vector space model is used for instantiation. With arbitrary instantiation, the training error stagnates after around 70 epochs and is clamped to a value of around 0.4, possibly because of getting stuck at a local minimum. This is not unusual as it has been observed in neural network literature that the speed of convergence can critically depend on the starting weights. Using VSM-based instantiations in ECRN helps to recover from this problem on all six datasets.

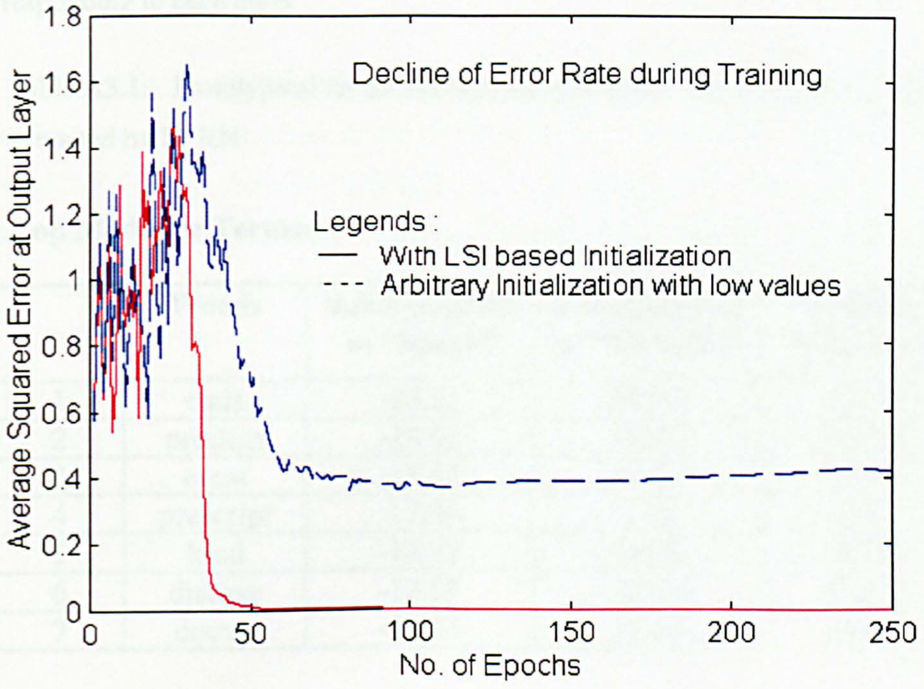


Figure A3.1 Error rate reduction with training

A3.2 Prototypicality of terms to classes

We performed an experiment to determine how prototypical terms are, to the different classes. Towards this end, we treated each term as a document containing just that term and no other, and allowed it to be classified by the ECRN. We thus obtained a score corresponding to each class at the output layer of the ECRN. The term was assigned to the class with highest similarity. The top few prototypical terms from a two class sub-problem constructed from SCIENCE, having terms pertaining to medicine and space domains, are shown in Table A3.1 below. The absolute difference between the scores assigned to two classes is treated as a measure of prototypicality, and used as the ranking criterion. Most of the top terms correspond well to the topics intuitively, suggesting that the ECRN is effective in separating out the discriminating features corresponding to each class.

Table A3.1. Prototypical terms corresponding to Space and Medicine domains, extracted by ECRN

Top Medicine Terms:

	Words	Belongingness to "Space"	Belongingness to "Medicine"	Absolute Difference
1	caus	-25.11	25.08	50.19
2	product	-18.62	18.61	37.24
3	creat	-17.67	17.59	35.19
4	prescript	-17.44	17.42	34.86
5	kind	-14.49	14.48	28.97
6	disease	-12.59	12.58	25.17
7	doctor	-12.35	12.34	24.69

Top Space Terms:

	Words	Belongingness to "Space"	Belongingness to "Medicine"	Absolute Difference
1	space	14.68	-14.67	29.35
2	accessdigexnet	14.04	-14.03	28.08
3	launch	12.46	-12.45	24.90
4	orbit	12.24	-12.22	24.46
5	scispac	12.15	-12.14	24.28
6	rocket	11.72	-11.71	23.43
7	nuclear	11.44	-11.43	22.87

A3.3 Limitations of the ECRN

It is important to note that the relevance values acquired by ECRN improve effectiveness when embedded as part of the network, but not when decoupled from the network. This is a serious limitation, as the relevance weights cannot be used independently by a CRN modelling an instance based classifier. This also has the disadvantage that knowledge of feature similarity cannot be mined from the ECRN. As an example, the similarity knowledge extracted from the revised weights learnt by ECRN yielded only an accuracy of 62% in the LINGSPAM domain when plugged into an ECRN; this compares poorly with 98.32% recorded by LSI-mined similarities. Thus the improved effectiveness achieved by ECRN is grossly outweighed by the fact that the acquired knowledge is not readily usable by instance based learners.

Appendix A4

Published papers

1. Chakraborti, S., Beresi, U., Wiratunga, N., Massie, S., Lothian, R., & Watt, S. (2007). A Simple Approach towards Visualizing and Evaluating Complexity of Textual Case Bases. *To appear in the Proc. of the fourth workshop on TCBR (ICCBR 07).*

2. Chakraborti, S., Lothian, R., Wiratunga, N., & Watt, S. (2006). Sprinkling: Supervised Latent Semantic Indexing. *Procs. of 28th European Conf. on Information Retrieval*, (pp. 510-515).Springer.

3. Chakraborti, S., Lothian, R., Wiratunga, N., Orecchioni, A., & Watt, S. (2006). Fast Case Retrieval Nets for Textual Data. *Proc. of the 8th European Conference on Case-Based Reasoning (ECCBR-06)* (pp. 400-414). Springer.

4. Chakraborti, S., Mukras, R., Lothian, R., Wiratunga, N., Watt, S., & Harper, D. (2007). Supervised Latent Semantic Indexing using Adaptive Sprinkling. *Proc. of the Twentieth IJCAI Conference* , (pp. 1582-7).AAAI Press

5. Chakraborti, S., Watt, S., & Wiratunga, N. (2004). Introspective Knowledge Acquisition in Case Retrieval Networks for Textual CBR. *Proc. of the 9th UK CBR Workshop*, (pp. 51-61).

6. Chakraborti, S., Wiratunga, N., Lothian, R., & Watt, S. (2007). Acquiring Word Similarities with Higher Order Association Mining. . *To appear in the Proc. of the 7th International CBR Conference (ICCBR'07).* Springer.

7. Chakraborti, S., Wiratunga, N., Lothian, R., & Watt, S. (2005). Fast Case Retrieval Nets for Textual CBR. *Procs. of the 10th UK CBR Workshop* .

8. Mukras, R., Wiratunga, N., Lothian, R., Chakraborti, S., & Harper, D. (2007). Information Gain Feature Selection for Ordinal Text Classification using Probability Re-distribution. *Proc. of IJCAI Textlink Workshop.*

9. Orecchioni, A., Wiratunga, N., Massie, S., Chakraborti, S., & Mukras, R. (2007). Learning Incident Causes. *Proc. of the TCBR Workshop 2007 (ICCBR).*

10. Wiratunga, N., Lothian, R., Chakraborti, S., & Koychev, I. (2005). A Propositional Approach to Textual Case Indexing. *Proc. of The PKDD Conference, Springer* , (pp. 380-391).

11. Wiratunga, N., Lothian, R., Chakraborti, S., & Koychev, I. (2005). Textual feature construction from keywords. *Weber, R and Branting, LK (eds) Proceedings of the Textual Case-Based Reasoning Workshop (ICCBR),* (pp. 110-119). Chicago.

Note: An updated version of publication 5 has appeared in the BCS SGAI journal Expert Update. An updated version of publication 7 is also slated to appear in a forthcoming issue of the same journal.