Detection and Management of Redundancy for Information Retrieval

A thesis submitted for the degree of
Doctor of Philosophy

Yaniv Bernstein B.C.S., B.App.Sci. (Comp. Sci.) Hons.
School of Computer Science and Information Technology,
Science, Engineering, and Technology Portfolio,
RMIT University,
Melbourne, Victoria, Australia.

26th July, 2006

Declaration

Yaniv Bernstein
School of Computer Science and Information Technology
RMIT University
26th July, 2006

Acknowledgments

I am grateful to Professor Justin Zobel for his careful and generous supervision throughout this project. To be guided by someone with such immense skill both as a mentor and as a collaborator has been a privilege. I have greatly enjoyed my time working with Justin and look forward to doing so again in the future.

My thanks also to my second supervisor, Saied Tahaghoghi. I have found Saied patient, attentive, and keen to help. My meetings with him were always very useful.

Though he had no formal role in this project, I also thank Andrew Turpin for his insights, occasional contributions, and general good company.

In 2004 I was involved in the RECAP project along with Justin Zobel, Alistair Moffat, W. Bruce Croft and Donald Metzler. Though the work emanating from this project has not been directly integrated into this thesis, it was influential in shaping some of my ideas. I found the experience of working with this group invigorating, stimulating and enjoyable.

The research presented in Chapter 6 was the result of work jointly conducted with fellow student Michael Cameron. Michael's impressive knowledge, insight, inventiveness, and work ethic meant that we were able to produce a significant piece of work in just a few months. I very much enjoyed working with Michael, and learned much from our collaboration.

The research presented in Chapter 7 resulted from a collaboration with another fellow student, Milad Shokouhi. Milad's enthusiastic and incisive contributions meant that work progressed smoothly and rapidly. The result was once again a substantive piece of work and a publication within a short period.

Being housed in the 'SEG Cave' has made my PhD feel a lot less like hard work. I am indebted to all the people who have passed through here over the years for their help and support, and for making the Cave such a great place to be. I would like to particularly thank Michael Cameron, Bodo Billerbeck, Nick Lester, Steven Garcia and Jonathan Yu for their enjoyable and morale-boosting companionship.

My parents have never been any less than entirely supportive in everything I choose to do. I thank them for their wise advice, for always being there, and for their unquestioning faith in me.

Finally, I thank my wife Chloe for her careful proofreading and judicious advice. I am grateful to her for supporting me in many different ways over the last three years. Without the benefit of her warm companionship and level-headedness, this would have been a less successful and less enjoyable endeavour.

Credits

Portions of the material in this thesis have previously appeared in the following publications:

Compact Features for Detection of Near-Duplicates in Distributed Retrieval
Yaniv Bernstein, Milad Shokouhi and Justin Zobel. Proceedings of Thirteenth Symposium on String Processing and Information Retrieval (SPIRE 2006), Glasgow, United Kingdom, October 2006. To appear.

Accurate Discovery of Co-Derivative Documents via Duplicate Text Detection
Yaniv Bernstein and Justin Zobel. Information Systems, To appear.

Fast Discovery of Similar Sequences in Large Genomic Collections
Yaniv Bernstein and Michael Cameron. Proceedings of the European Conference on Information Retrieval (ECIR 2006), London, United Kingdom, April 2006, pp. 432-443

Clustering Near-Identical Sequences for Fast Homology Search
Michael Cameron, Yaniv Bernstein and Hugh E. Williams. Proceedings of the Conference on Research in Computational Molecular Biology (RECOMB 2006), Venice, Italy, April 2006, pp. 175-189

Redundant Documents and Search Effectiveness
Yaniv Bernstein and Justin Zobel. Proceedings of the ACM CIKM Conference on Information and Knowledge Management (CIKM 2005), Bremen, Germany, November 2005, pp. 736-743

A Scalable System for Identifying Co-Derivative Documents
Yaniv Bernstein and Justin Zobel. Proceedings of the String Processing and Information Retrieval conference (SPIRE 2004), Padua, Italy, October 2004, pp. 55–67

This work was supported by the Australian Research Council and the Data Engineering group at RMIT University.

The thesis was written in the `vim` editor on Gentoo GNU/Linux, and typeset using the $\LaTeX\,2_\varepsilon$ document preparation system.

Note

Unless otherwise stated, all fractional results have been rounded to the displayed number of decimal figures.

Contents

List of Figures

List of Tables

Abstract

The growth of the web, authoring software, and electronic publishing has led to the emergence of a new type of document collection that is decentralised, amorphous, dynamic, and anarchic. In such collections, redundancy is a significant issue. Documents can spread and propagate across such collections without any control or moderation. Redundancy can interfere with the information retrieval process, leading to decreased user amenity in accessing information from these collections, and thus must be effectively managed.

The precise definition of redundancy varies with the application. We restrict ourselves to documents that are co-derivative: those that share a common heritage, and hence contain passages of common text. We explore document fingerprinting, a well-known technique for the detection of co-derivative document pairs. Our new lossless fingerprinting algorithm improves the effectiveness of a range of document fingerprinting approaches. We empirically show that our algorithm can be highly effective at discovering co-derivative document pairs in large collections.

We study the occurrence and management of redundancy in a range of application domains. On the web, we find that document fingerprinting is able to identify widespread redundancy, and that this redundancy has a significant detrimental effect on the quality of search results. Based on user studies, we suggest that redundancy is most appropriately managed as a postprocessing step on the ranked list and explain how and why this should be done.

In the genomic area of sequence homology search, we explain why the existing techniques for redundancy discovery are increasingly inefficient, and present a critique of the current approaches to redundancy management. We show how document fingerprinting with a modified version of our algorithm provides significant efficiency improvements, and propose a new approach to redundancy management based on wildcards. We demonstrate that our scheme provides the benefits of existing techniques but does not have their deficiencies.

Redundancy in distributed information retrieval systems — where different parts of the collection are searched by autonomous servers — cannot be effectively managed using traditional fingerprinting techniques. We thus propose a new data structure, the grainy hash vector, for redundancy detection and management in this environment. We show in preliminary tests that the grainy hash vector is able to accurately detect a good proportion of redundant document pairs while maintaining low resource usage.

Chapter 1

Introduction

For much of human history the propagation of knowledge and information has been a costly affair. For a document to be made available to a wide audience, it had to undergo a laborious publication process before being physically distributed through standard channels such as newsagents, booksellers and libraries. The complexity and expense of this process ensured that documents were carefully prepared before publication, and that publication occurred according to well-coordinated procedures. Thus, the libraries that represented the collective record of human knowledge consisted largely of an orderly accumulation of documents of well-known provenance, and of generally high quality.

In recent decades, the emergence of information technology — in particular the growing pervasiveness of digital documents and networking — has all but removed these barriers to publication. This has resulted in a fundamental change to the way in which human knowledge is accumulated. Liberated from the constraints imposed on them by the formal publication and distribution process, documents can now be cheaply and easily made public by individuals. Freed from the need for physical aggregation in libraries, electronic documents can form emergent, *ad hoc* collections with no central control.

In contrast to the rigid, centralised, carefully curated, and relatively static libraries of professionally authored and edited documents that have traditionally codified human knowledge, the new type of collection is decentralised, amorphous and dynamic. It contains documents, often self-published, of varying quality. The archetype of this new type of collection is the world-wide web, a vast and eclectic aggregation of knowledge and nonsense from around the planet. However, such collections exist in many other contexts, such as in enterprises and on personal computers.

The revolution in publication brought about by information technology presents new opportunities. In particular, the ability of computers to manipulate digital information allows us to analyse and search repositories of knowledge in unprecedented ways. However, the new type of collection also presents difficult challenges. The organic and decentralised nature of such collections means that there is no possibility for moderation or quality control, nor for organisation or curation. Whereas an information scientist working on a search system for a library has the luxury of dealing with a well-annotated, structured, and categorised dataset, the information scientist designing a search system for a digital collection must expect their dataset to be loosely annotated and chaotic. Novel approaches to collection management are necessary in order for search to be effective in such a domain.

One of the challenges of the new type of collection is management of redundancy. The technologies that enable the rapid transfer of information around the world also result in a massive amount of data duplication, because data in digital documents can be copied with trivial ease. Combined with the lack of centralised control of digital collections, this means that collections are likely to contain a large number of document-pairs that are duplicates or near-duplicates of each other, and are therefore mutually redundant.

The presence of redundancy in collections can prove a serious hindrance to the efficient discovery and utilisation of information. In standard ad hoc search, repetition of effectively the same document in the ranked result list reduces user amenity. In distributed search, this problem is compounded by collection overlap, which is when the same documents appear in multiple independent collections that participate in the search process. In genomic search, the presence of redundancy has the potential to distort the search process itself.

In this thesis, we address questions concerning the definition, detection and management of redundancy in digital document collections.

How do we define redundancy?

When a concept seems to have an intuitive or 'common-sense' meaning it is tempting to neglect to provide an explicit definition. Much previous work addressing the issue of redundancy management in document collections talks of algorithms for finding 'duplicates' or 'near-duplicates'. In the absence of an independent definition, it is easy to succumb to the temptation of defining the problem in terms of the solution. Near-duplication is often implicitly defined as the property that is detected by the proposed algorithm, or in terms of some measure that is easily calculated by the algorithm.

We are careful to define redundancy in a functionally-driven way: that is, we define redundancy for a particular application in terms that are appropriate to the intended use-case for that application. In this way, we are able to evaluate the effectiveness of our tools for redundancy management with reference to the anticipated use of the tools, rather than with respect to an arbitrary or circular definition.

In Chapter 5 we address the management of redundancy in results from web searches. In this situation we wish to shield the user from reading multiple documents that essentially contain the same content, but at the same time avoid suppressing novel information that may be of interest. Guided by a use-case in which redundant documents are not indexed, we define the relation of content equivalence, which exists between pairs of documents that contain the same content. We conduct a user study to correlate content equivalence with the score produced by a document fingerprinting system. The outcome of this user study prompted us to formulate an alternative definition of redundancy, which we call conditional content equivalence. This relation was more accurately identified by our fingerprinting system, but required a different approach to document management: all documents must be indexed, and redundancy management must take place as a post-processing step on a ranked list. Our experience demonstrated that definitions have an impact on practice, and thus must not be neglected.

In Chapter 6 we improve on an existing technique for management of redundancy in genomic sequence databases. Our new methodology uses wildcards in order to search redundant sequences in parallel. As our aim is improved search speed, we implicitly define redundancy between sequences as the property that allows them to efficiently form a cluster. Outside the context of our implementation, our definition of redundancy would probably be meaningless. However, it is appropriate for our use-case. Because we evaluate success in terms of reduced execution times rather than directly by the number of redundant sequence-pairs identified, we avoid the issue of circularity in our definition.

How do we detect redundancy in text documents?

We hypothesize that the best candidate tools for managing redundancy are those that are aimed at detecting pairs of documents that are *co-derivative*. A pair of documents is co-derived if they share a common heritage: either one document is derived from the other, or both are derived from a third ancestor document. It is reasonable to conjecture that the set of redundant documents in a collection is a subset of the set of those that are co-derivatives.

There are several methodologies for the detection of co-derivation between documents in a collection. We investigate three broad categories of technique: approximate string matching, term-vector based algorithms, and document fingerprinting. While approximate string matching and term-vector techniques can both be effective for the *search problem* in which the co-derivatives of a query document are located, they cannot be efficiently implemented for the more difficult *discovery problem* in which all co-derivative document pairs in the collection must be identified.

Document fingerprinting techniques work by creating a 'fingerprint' for each document. A document fingerprint contains one or more feature descriptors from the document such that pairs of co-derivatives are likely to share some features in their respective fingerprints, whereas unrelated documents are unlikely to do so. Document fingerprinting techniques can be categorised by the type of feature they index. We focus on the most common group of algorithms, which make use of features called *chunks*. Chunks are contiguous text sequences extracted by passing a sliding window over each document. The intuition is simple: a pair of co-derived documents are quite likely to have many chunks in common, whereas this is extremely unlikely in unrelated documents.

Due to the large number of candidate features generated by chunk-based fingerprinting algorithms, practical implementations must generally make use of a *selection heuristic* that selects a subset of the available chunks to form the representative fingerprint for each document. While selection heuristics play an important role in limiting the resource consumption of chunk-based fingerprinting algorithms, they have the effect of introducing lossiness to the fingerprinting system. With current selection heuristics, there is a distinct possibility that even closely co-derived documents may be overlooked by the system.

In order to address this problem, we have developed SPEX, the first lossless selection heuristic for chunk-based fingerprinting systems. Lossless chunk selection is possible because singleton chunks — those only appearing once in the collection, and often composing a sizable majority of the total — have no influence on the outcome of the co-derivative discovery process. Thus, SPEX is able to significantly reduce the total size of the fingerprinting index by using a memory-efficient iterative hashing technique to keep track of non-singleton chunks.

Although SPEX has been successfully used to index collections of up to 18 GB, there are some concerns about resource consumption and scalability. In particular, we were unable to index the 426 GB GOV2 collection on any of the hardware at our disposal. This was not so much due to the raw size of the collection as the level of redundancy within it. Spex operates by identifying and discarding singleton chunks. Given the highly redundant nature of GOV2, such chunks constituted a relatively small proportion of the overall pool. As such, SPEX was relatively ineffective. This sensitivity to the overall level of redundancy in the collection is a fundamental deficiency in the nature of the algorithm, and suggests that SPEX should be restricted to domains where the degree of redundancy is at most moderate.

How should redundancy be managed for web search?

On the web, document redundancy exists on a large scale and in many contexts. Examples include website mirroring, document syndication, automatically generated text, per-site documentation, boilerplate text, multiple formats, aliasing, and spam pages. The consequences of this redundancy include larger indexes and slower search times, but most significantly a degradation in the utility of ranked lists to the user. A ranked list containing many copies of the same document — even if it is highly relevant — is not ideal from the perspective of the user who, in general, wants to view each document only once.

Modern ranking algorithms and most current metrics for the evaluation of search engine quality derive from the probability ranking principle, which states that documents should optimally be ranked in decreasing order of likelihood of relevance to the user. The consequence is that common practice in both ranking and evaluation is to treat the relevance of a document with respect to the query as the only property of interest. Thus, the issue of inter-document redundancy has been something of a blind-spot in information retrieval research.

We study the effect of redundancy on search in two web collections that were created for the TREC initiative.[1] Those are the 18 GB GOV1 collection and the 426 GB GOV2 collection, both crawls of the `.gov` domain. Using DECO— our fingerprinting package incorporating the SPEX algorithm — on these collections we were able to demonstrate that there was a high level of content equivalence amongst documents in both collections. This confirms that redundancy is a serious problem in the management of web collections.

To more directly evaluate the effect of this collection redundancy on search results, we made use of data from the 2004 TREC terabyte track. The ad hoc search task of the track required participants to submit ranked lists for each of 50 topics created for the track. Documents were then judged for relevance using a standard pooling process. Running DECO on the documents from the judgement pool allowed us to estimate the degree of redundancy

[1]The TREC initiative is funded by the United States National Institute of Science and Technology with the aim of creating a standard infrastructure for the evaluation of information retrieval systems. For more information, see Section 2.4.

amongst judged relevant documents that were returned by the best efforts of leading research search systems. Applying these results to the runs themselves allowed us to estimate the degree to which search engine effectiveness was being overestimated for this task and, as a corollary, the degree to which effectiveness could be improved if search engines made use of a fingerprinting or similar tool in order to manage redundancy in result lists.

Our evaluation of the GOV1 and GOV2 collections discovered a high level of redundancy in both collections. In particular, we were able to show that more than 25% of documents in the GOV2 collection were redundant. While the level of redundancy in the ranked search runs was lower than it was in the collection as a whole, it was still quite high: approximately 15% of all relevant documents appearing in the ranked lists were equivalent to another document higher up in the list. Interestingly, the level of redundancy was actually higher amongst documents that were judged to be relevant than those that were not judged relevant. This results in a significant inflation of reported search effectiveness values. We estimated that reported effectiveness was about 20% higher than would be the case if redundancy were taken into account. The failure of commonly-used evaluation metrics to take redundancy into account has stifled progress in redundancy management, as improvements in this area do not lead to improved metric values. Given a metric that takes proper account of the effect of redundancy, the picture looks quite different. Using such a metric in Chapter 5, we estimate that many search systems could improve their effectiveness by more than 15% if redundancy were properly managed.

Using DECO to discover co-derivation relationships between documents in the judged pool for the 2004 terabyte track also gave us a convenient opportunity to evaluate the quality of the expert relevance judgements for the documents in the pool. The results were concerning: more than 12% of potentially relevant documents in content-equivalent clusters had been inconsistently judged.

How should redundancy be managed for genomic search?

Sequence homology search is an important tool for biologists, who use it to infer information regarding the structure and function of new genetic material by comparing it to the existing body of knowledge.

Comprehensive databases of genomic sequences such as GenBank contain sequences submitted by laboratories from all over the world. Despite the fact that exact duplicates are generally removed prior to distribution, such collections continue to exhibit a high degree of redundancy in the form of near-duplicate sequences.

While repetitive result lists are an irritating consequence of redundancy in genomic sequence databases, redundancy also has more serious consequences in this context. Many genomic tools such as the popular BLAST tool make use of collection statistics during the search process. A high degree of redundancy skews these statistics and potentially reduces search effectiveness. This is especially true for profile-based tools such as PSI-BLAST.

Furthermore, the need for sensitive approximate-matching alignment measures during the genomic search process means that such tools are generally several orders of magnitude slower than a term-based text search engine on collections of similar size. Thus, the inflation in database size brought about by internal redundancy leads directly to slower searches, often increasing the time taken to return a result by several seconds.

In the past, redundancy in genomic sequence databases has typically been managed by the creation of representative-sequence databases (RSDBs), in which redundant sequences are clustered and only one sequence from each cluster — the representative — is retained. The resulting database is guaranteed to contain no sequence-pairs with a level of identity exceeding the clustering threshold. When compared to the source database, RSDBs are smaller, produce less repetitive results, and lead to less statistical skew and profile corruption.

Current techniques for RSDB construction use fast pairwise comparison techniques to calculate similarity between all (or in some cases most) pairs of sequences. Despite their speed, however, the number of comparisons grows quadratically in the size of the collection; thus, as collections grow, these construction techniques are becoming less feasible. We have investigated the possibility of using document fingerprinting, which does not suffer from quadratic complexity in computational cost, to discover pairs of sequences exceeding the identity threshold for clustering.

Document fingerprinting techniques have not been previously applied to genomic sequences, but only trivial changes are required in order for a working system to be constructed. However, the vastly different characteristics of genomic sequence strings when compared to natural-language documents present some challenges to the efficiency of fingerprinting. We introduce the slotted SPEX selection heuristic, a derivative of the SPEX algorithm that is particularly suitable for this domain. Slotted SPEX builds on the existing SPEX algorithm by adding a level of controlled lossiness. The result is a lower load on the hashtables, faster execution time, and a smaller index, while still guaranteeing that all contiguous sequences above a specified threshold length will be detected. Our experiments show that slotted SPEX is remarkably accurate at isolating pairs of sequences with a high degree of identity. Even with parameterisations that make the heuristic significantly more selective, we are able to achieve near-perfect accuracy at discovering sequence pairs with an identity of 90% or above. Using slotted SPEX as the basis for clustering allowed us to build a RSDB of the GenBank nonredundant protein database in one-sixth the time of the fastest existing method.

While RSDBs are undeniably useful for certain applications, they are somewhat unsatisfactory as a general method for the management of redundancy in genomic sequence collections. In many cases, searching against an RSDB will result in markedly lower effectiveness than would search against the full database. Because significant numbers of sequences have been simply removed from the RSDB, the best results for a given query may be absent from the database. We introduce a new integrated approach for managing redundancy in genomic databanks, which consists of novel techniques for clustering and sequence representation, and an adaptation of the BLAST search process. Our approach is based on representing clusters of similar sequences using a special *union-sequence*, in which points of difference between sequences in the cluster are indicated using a wildcard. We are able to simultaneously represent all sequences in the cluster using this single union-sequence. When using our new approach we are able to reduce the size of the GenBank nonredundant protein database by over 25% and reduce search time by over 20% without any significant change to search effectiveness.

How should redundancy be managed in a distributed context?

In a distributed search system, users search multiple autonomous systems at once by interacting with a *broker* system. The broker takes the user's query, chooses the systems to which the query will be sent, and merges the results from the various sources in order to present the user with a single ranked list of results.

Alongside the problems of intra-collection redundancy, distributed search systems face the additional issue of inter-collection redundancy or collection overlap, in which the same or highly similar documents appear in more than one of the collections being searched. Despite the seriousness of the redundancy problem in this context, it has not been previously addressed beyond the trivial measure of taking hashes to eliminate exact duplicates.

The problem of collection overlap cannot be efficiently solved using the standard chunk-based fingerprinting techniques discussed above, because the relatively large fingerprints required for them to be effective would consume an undesirable amount of bandwidth in a distributed setting. Furthermore, as redundancy management must be performed by the broker at result-merger time, the computational cost of evaluating pairs of large fingerprints would be prohibitive.

We have proposed a new data structure for the problem of redundancy management in a distributed environment. The grainy hash vector (GHV) is a compact data structure that encodes a large number of features within a single 32- or 64-bit word using a narrow-hash representation. We used chunks as features, but the data structure can be used equally well with other feature types. The small vectors do not place a large burden on network bandwidth, and the single-word nature of the structures mean that bit-parallelism can be used to process them extremely rapidly. We demonstrate that GHVs can be used as reasonably accurate predictors of conditional equivalence for the distributed search context.

1.1 Thesis structure

We present a broad background of relevant work and concepts in Chapter 2. We describe the co-derivation relation and the search and discovery problems for co-derivatives. The chapter also includes a discussion of search engines: the principles underlying their operation, the models and mechanisms used in producing a final ranked list, and the many ways in which the effectiveness of search engines can be evaluated. We also discuss the fields of distributed search, approximate string matching, and genomic search methodologies.

In Chapter 3 we investigate the problem of discovering co-derivative relationships between documents in a collection. After first considering some alternatives, we present a comprehensive taxonomy of document fingerprinting techniques. We discuss the basic intuition underlying the fingerprinting approach, and show how a range of seemingly disparate techniques share the same fundamental mechanics. We describe the many choices that can be made in building a document fingerprinting system, and the consequences of these choices. We finish the chapter with a discussion of the efficiency aspect of the fingerprinting process.

In Chapter 4 we present SPEX, our novel algorithm that for the first time allows lossless selection of chunks (features) in the fingerprinting process. After describing the operation of the algorithm, we introduce DECO, our software package for document fingerprinting that incorporates SPEX as well as number of other novel implementational optimisations. After defining performance metrics for the discovery problem, we perform a series of experiments that demonstrate the effectiveness of the SPEX approach.

In Chapter 5 we examine the prevalence of redundancy in web collections, and the impact of such redundancy on the user search experience. We define content equivalence and conditional content equivalence, and report on the outcomes of a user study correlating these relations to scores produced by our fingerprinting system. Using these results, we demonstrate that redundancy is not only a significant presence in underlying web collections, but that it also has a real and significant impact on the quality of search results.

We apply document fingerprinting to the management of redundancy in genomic sequence collections in Chapter 6. We discuss the challenges faced in implementing fingerprinting for genomic data before describing slotted SPEX, a modified version of SPEX that is tailored for this domain. We demonstrate that fingerprinting is accurate at discovering pairs of sequences with a high level of identity, and use a tool based on fingerprinting to create representative-sequence databases (the most popular existing strategy for redundancy management) up to six times faster than the fastest existing methods. Finally, we present our novel method for redundancy management in genomic collections, including new techniques for clustering, representation and search. We demonstrate that using our methodology on a comprehensive protein sequence database such as GenBank results in a smaller database representation and searches that are more than 20% faster that when using a standard database representation.

In Chapter 7 we introduce the grainy hash vector (GHV), a new data structure for managing redundancy in cooperative distributed information retrieval systems. We demonstrate that despite their small size, GHVs are able to effectively identify a significant proportion of conditionally equivalent document-pairs within ranked lists. We verify empirically that by taking advantage of bit-parallelism, the comparison of pairs of GHVs consumes few processor cycles, and thus that their use for merge-time processing of ranked lists is a reasonable proposition.

We conclude the work, and discuss possibilities for future research, in Chapter 8.

Chapter 2

Background

Our study of the management of redundancy proceeds from a base of existing work, and traverses several domains within the area of information retrieval. In this chapter we provide the background and contextual information necessary to interpret the contributions presented in later chapters.

We found that task definition for this work was more complex that initially thought; despite the intuitive simplicity of the notion of inter-document redundancy, we discovered some subtle issues that have not always been successfully negotiated in earlier work. We discuss these issues, and our approach to defining the task, in Section 2.1.

Hash functions play an important role in a number of the techniques and algorithms discussed in this thesis. We briefly discuss some pertinent hash functions and their features in Section 2.2.

A core focus of our study is the interaction between collection redundancy and the information retrieval process. In Section 2.3 we survey the historical and theoretical foundations underpinning modern information retrieval systems, and discuss the way in which these principles influence search-engine design and evaluation.

In Section 2.4 we introduce the TREC initiative for evaluation of information retrieval systems. We explain the practices followed by TREC for creating robust, reusable datasets for system evaluation, and describe those tracks most pertinent to our work.

Distributed information retrieval generalises the information retrieval model to situations in which the user wishes to query across multiple autonomous search systems. This scenario introduces new complexity to the search process. We discuss the state of the art in this area in Section 2.5.

Information retrieval systems are required to operate efficiently over large quantities of information, and as such use data structures that enable rapid access to specific data. Suffix structures and inverted indexes, two common data structures that are used for this purpose, are described in Section 2.6.

In Section 2.7 we discuss a variety of approximate string matching algorithms, which are designed to efficiently calculate the degree of similarity between non-identical strings. Such algorithms are a feasible approach to the problem of redundancy identification and, significantly, form the basis of the sequence homology search tools discussed in Section 2.8. Sequence homology search is an information retrieval task over genomic sequence data, and is an important tool in many areas of biology. We introduce the basic approach to this problem, discuss its limitations, and describe more advanced heuristic systems.

2.1 Defining redundancy

In this thesis, we address the issues surrounding the detection and management of redundancy within collections. It is reasonable to conjecture that a pair of redundant documents will in most cases share a substantial portion of text. Thus, our initial stated approach to the detection of redundancy was to investigate classes of algorithms that are able to identify documents that share a reasonable amount of text. During this preliminary investigation, it became clear that this seemingly easy-to-define task presented far more difficulties than were initially apparent.

Notions such as 'redundancy', 'novelty', 'duplication', or 'near-duplication' seem so obvious and intuitive that most previous works do not define them. Manber (1994), Brin et al. (1995), Heintze (1996), Sanderson (1997), Lyon et al. (2001), and Schleimer et al. (2003) are all examples of works in which such terms are not meaningfully defined. It is frequently left to context and intuition to determine approximately what is meant by the term in question.

In cases where terms are defined, it is often in an unsatisfactory manner. Broder et al. (1997) describe two set-theoretic measures, resemblance and containment, for estimating the degree of 'similarity' between a pair of documents (see Section 3.6.4), but do not relate these measures to the notion of near-duplication. They then go on to assume that any document clusters computed by their system based on a lossy estimate of the resemblance measure are near-duplicates. This circular definition assures the apparent success of their technique, but tells us little about what it actually going on. Fetterly et al. (2003) follow Broder et al. (1997) in defining near-duplication in terms of a metric based on an estimate of resemblance.

The following is from Chowdhury et al. (2002):

> "The general notion is that if a document contains roughly the same semantic content it is a duplicate whether or not it is a precise syntactic match."

This definition is vague; worse, it is at odds with an intuitive notion of duplication. This problem is largely immaterial, however, as the definition is largely ignored in favour of a system-oriented approach as above.

It is apparent that, while notions such as text reuse and near-duplication are seemingly intuitive and straightforward, they actually elude simple and decisive definition.

What does it mean to say that a pair of documents share text? It is obvious that the notion is somehow syntactic — two passages with the same semantics need not contain any shared text — but the exact syntactic form of shared text is quite difficult to define. Must the documents contain a single contiguous sequence of text that is identical in both documents? If so, how long must it be, and why? What if the text were interspersed with other, original text? If interspersions are allowed, how significant may they be before they fundamentally alter the text?

The answers to such questions are of necessity subjective. This highlights the difficulty of defining the notion of shared text in a rigorous fashion. We argue that an alternative approach is required to defining the problem domain. It seems reasonable to consider that documents share text if it seems implausible that the text in the two documents could have been independently written; that is, shared text can be taken as evidence of a shared heritage. Thus, it makes more sense to define the problem in terms of detecting documents that share a heritage, rather than the relying on the difficult-to-define notion of shared text.

We define a pair of documents as *co-derivative* if they have not been independently written; that is, one document is derived from the other, or both are derived from a third source, either directly or indirectly (Hoad and Zobel, 2003); an illustration of possible co-derivation relationships is presentes in Figure 2.1. This definition of co-derivation is a broad one that encompasses extrinsic evidence, and thus cannot be inferred by merely inspecting the documents. A more precise definition that makes use only of intrinsic evidence states that a pair of documents is co-derivative if one document could not plausibly have been written without reference to the other, or both could not have been written without reference to a joint common source. Note that, though the definition of co-derivation is concrete, the co-derivation status of a pair of documents must ultimately be judged by a subjective process. It is important not to confuse this with the situation described earlier with reference

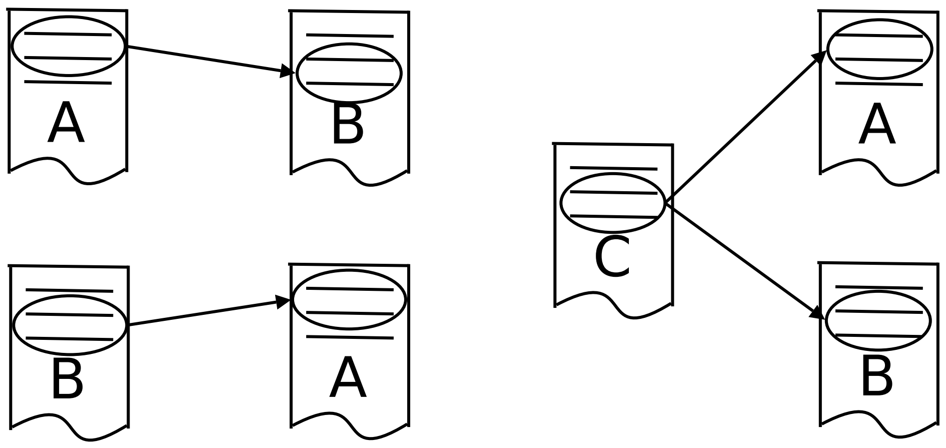


Figure 2.1: Three circumstances in which a co-derivation relationship can be said to exist between a pair of documents A and B: A contains text derived from B; B contains text derived from A; A and B both contain a portion of text derived from a third document C.

to the notion of shared text, in which it is the definition itself that is open to subjective interpretation.

The advantage of using co-derivation rather than shared text as the objective of the problem is that, while individual judgements of co-derivation are subjective, the definition is precise and objective; with shared text the definition itself is subjective. With co-derivation as the objective, the performance of a system can be judged against human judgements of a meaningful concept that reflects the application domain of the problem. This is analogous to the treatment of relevance in the query-based retrieval domain.

2.1.1 The relationship graph

The task of identifying co-derivation in a document collection is fundamentally a binary classification task over document pairs: given a pair of documents, we must either classify them as co-derived or not co-derived. We can thus represent the set of co-derivation relations in a document collection as an undirected graph, which we call the relationship graph.

Let the relationship graph R for a document collection C be represented by $< V, E >$ where $|V| = |C|$ and $e_{ij} \in E$ if and only if $i \bowtie j$, where $\bowtie$ represents the co-derivation relationship. That is, each vertex in the relationship graph corresponds to a document in the collection, and two vertices are joined by an edge if and only if the corresponding documents are co-derived. The relationship graph for a hypothetical collection of eight documents is illustrated in Figure 2.2.

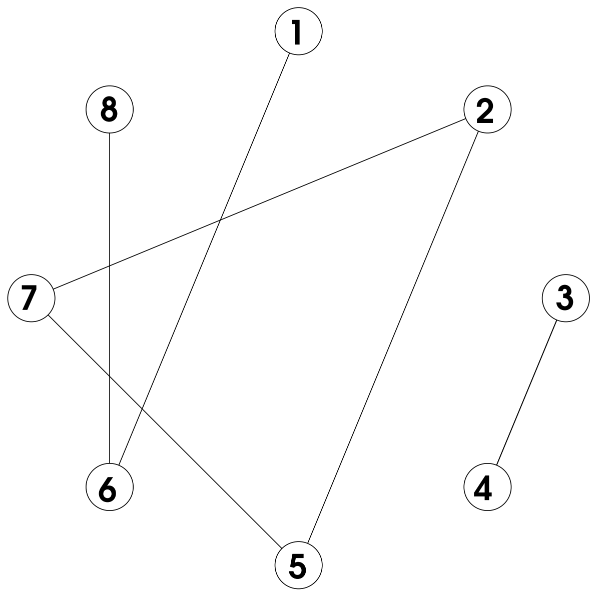


Figure 2.2: An example of a relationship graph for a collection of eight documents.

The relationship graph is a useful tool for visualising the co-derivation structure of a collection and understanding the nature of problems in the domain of co-derivative detection (see Chapter 3). It serves as the basis for several techniques for evaluating the quality of co-derivation classifications, as discussed in Chapter 4.

2.1.2 The search and discovery problems

There are two basic problem scenarios in the domain of co-derivative document detection. The first scenario is that in which there is a particular document of interest for which we wish to find all co-derivatives. Such a situation may occur in a document registry when a particular document is being checked for breach of copyright, or on an intranet when a user wishes to know whether the copy of a document they possess is the most up-to-date version. We call this the $1 \times n$ or *search* problem to emphasise that the task in this situation is to search a collection for documents that are co-derived with a target document, much as ranked search seeks to find documents that are relevant to a given query.

In the second scenario, there is no natural target document. Rather, we are presented with a collection of documents and wish to find co-derivation relationships between any pair of documents in the collection. Such a scenario may occur in the academic domain when a teacher wishes to check whether any students plagiarised or collaborated on a particular assignment, or in the search domain in order to prune an index of duplicate entries. We call this the $n \times n$ or *discovery* problem to emphasise the fact that the task is to discover arbitrary co-derivation relationships within a given collection. The problem can alternatively be described as the task of discovering the structure of the underlying relationship graph for a document collection.

Note that the discovery problem is a strict superset of the search problem. Having resolved the discovery problem for a given collection of documents, we automatically have the answers for all n possible target documents within that collection. Conversely, given an algorithm for the search problem, the discovery problem can be solved by applying the algorithm n times, each time with a different document from the collection as a query. Given this relationship between the two problems, it is obvious that the discovery problem is the more general and the more difficult to efficiently solve.

The discovery problem is not a traditional clustering task because co-derivation is not typically a transitive relation. If documents A and B are known to be co-derived and documents B and C are similarly co-derived, we are not entitled to make the assumption that A and C are co-derived. To make this more concrete, imagine that document B is a digest-type document that draws together excerpts from various independently authored sources. In such a scenario, the above situation obtains. The fact that co-derivation is not transitive makes the discovery problem a particularly difficult one to tractably solve, and it is often overlooked. Broder et al. (1997) note that document resemblance is not transitive[1] but then on the very next page proceed to calculate clusters using a union-find structure to compute transitive closures (Broder et al., 1997, p. 1160).

2.2 Hashing and hash functions

Hashing is a fundamental operation in computer science, widely documented in basic textbooks such as Sedgewick (1998). The task of a hash function is to map input from a universe $\mathcal{U}$ to a range of M values (typically integers) in a deterministic but pseudo-random fashion. Thus, a sequence of inputs should map to a reproducible sequence of values with no discernible pattern. Hashing is invaluable for many applications in which it is necessary to uniformly project a set of values onto a smaller space. Hashing is particularly prominent in implementations of search engines and document fingerprinting algorithms (see Chapter 3).

[1] "a well-known fact bemoaned by grandparents all over" (Broder et al., 1997, p. 1159)

2.2.1 Properties of hash functions

We present several properties that are, in addition to computational efficiency, desirable in classes of functions used for hashing.

Universality

Carter and Wegman (1977) describe a property of classes of hash functions known as *universality.* A class of hash functions Π can be considered universal if, for a random function $\pi \in \Pi$, and for any pair of distinct elements x_1 and x_2, the following holds:

$$P(\pi(x_1) = \pi(x_2)) \leq \frac{1}{|M|}$$

That is, over the class of functions, the collision rate between the hash values of two distinct elements should be the same as the collision rate for two random samples chosen over the same space. This provides a degree of assurance that the class of functions is well-behaved in terms of the characteristics required of a hash function. Note that universality can only be held applicable across a *class* of hash functions: guarantees cannot be made for individual instances of a class, a deficiency that leads to situations in which universal classes of hash functions contain distinctly poor hash functions (Sarwate, 1980).

Universality can in some cases be proved, though for many practical and useful hash functions it cannot be. Empirical evidence is never sufficient to demonstrate universality, as a single counter-example suffices to falsify the claim. Nonetheless, experimentation can provide a strong basis for belief that a class of functions is universal.

Uniformity

A class of hash functions Π is uniform if for any given $\pi \in \Pi$ and randomly-selected $x \in \mathcal{U}$, x is equally likely to hash to any of the possible values in M:

$$\forall_{m \in M} P(\pi(x) = m) = \frac{1}{|M|}$$

Uniformity provides an assurance that, in the average case, a hash function spreads its input evenly amongst the space of available outcomes.

Min-wise independence

A class of hash functions Π is min-wise independent (Broder et al., 1998) if, for any given set of elements X, it is equally probable that any element $x \in X$ will be the smallest element when the set is submitted to a randomly chosen function from that family, $\pi \in \Pi$. That is:

$$P(\operatorname{argmin}(\pi(X)) = x) = \frac{1}{|X|}$$

Min-wise independence states that the class of hash functions is unbiased at least with respect to the identity of the element with the lowest hash value. Min-wise independence is sufficient for applications in which the aim is to pseudo-randomly select a single element from a set. This is the case in many situations, some of which we encounter in later chapters.

2.2.2 Classes of hash function

In a situation where resources were not a constraint, a good hash function would consist of a lookup table mapping all the elements of U to random values within M. However, in such an artificial situation, hashing would in most cases be unnecessary. Thus, practical hash functions must be formulaic or algorithmic so that the function description can be compact. A wide variety of such functions exist, each with different properties; we describe two families of hash functions in this section: the *shift-add-xor* class, and Rabin fingerprinting.

Ramakrishna and Zobel (1997) describes a class of hash function called *shift-add-xor* that is able to hash any type of digital data and, by virtue of the fact that it relies solely on computational primitives, can be calculated efficiently on current computers. As the name suggests, the class uses only the shift, add, and exclusive-or (xor) primitives on the original input to compute a hash. Though the class does not yield to analysis, their experiments provide reasonable weight to the conjecture that the class is universal and uniform.

Rabin fingerprinting (Rabin, 1981; Broder, 1993) is a class of hash functions that have algebraic properties making them useful in applications that call for the hashing of long overlapping strings. As well as being universal and having reasonably efficient hardware implementations, Rabin fingerprinting is able to compute the hash for a string that overlaps an already-hashed string in time proportional to the length of the new portion, rather than the length of the entire string. This confers a significant speed advantage to applications in which hashing proceeds in this manner, such as certain string-matching algorithms (Karp and Rabin, 1987) and some document fingerprinting strategies (see Chapter 3).

2.3 Information retrieval in text collections

Users refer to documents in order to meet some sort of need for information. In general, different documents have varying levels of utility when it comes to satisfying a particular information need. When faced with a large document collection, the user has a dilemma: though the collection may contain any number of documents satisfactory to the user, they have no way of locating them. The useful information is effectively inaccessible.

The role of an *information retrieval* or *search* system is to make it as easy as possible for a user to distill information that is useful or of interest out of the vast volume of available data. In this section we discuss the practical and theoretical considerations underlying the construction of an effective information retrieval system, as well as some of the many ways for evaluating their performance.

2.3.1 Boolean search

A Boolean expression is a logical expression consisting of variables taking on the values TRUE and FALSE, combined using the binary operators AND and OR, and the unary operator NOT. In the context of search, a *Boolean query* is a Boolean expression in which each of the variables represents a term. The expression is evaluated for each document in the collection by setting the value of each variable to TRUE if the term represented by the variable occurs in the document, and FALSE otherwise. For example, the expression:

(clown OR acrobat) AND NOT Adobe

would match all documents that contained one or both of the terms ‘clown’ and ‘acrobat’, but contained no instances of the term ‘Adobe’.

Boolean search allows a searcher to specify queries of arbitrary complexity, and allows an experienced searcher with a clearly defined objective and good domain knowledge a high level of control over their search. However, it leaves much of the work in the search process to the user.

The Boolean search task effectively resolves to a classification over documents: a well-formed Boolean query defines an expression over terms where the result set should consist of all documents that satisfy this expression. What a Boolean search does not do is discriminate amongst the documents within the set of those that satisfy the query expression: all are equally valid with respect to the query. While this is satisfactory in a situation where the result set contains up to a few dozen documents, it is somewhat problematic when the result set consists of more documents than the user could hope to inspect. In such a situation, the search system may have answered the query with complete accuracy. However, to what degree has the search helped the user in their task?

Though Boolean search automates the process of finding key terms within a large index and is useful in many search situations, it places the burden of task formulation entirely on the user. In the case of casual searchers, or searchers operating in unfamilar domains, this often leads to unsatisfactory search results. In order to further advance the utility of information retrieval systems, it is necessary to more explicitly model the user and their task, so that the system can provide more targeted results.

2.3.2 A user model for information retrieval

A user's interactions with a document collection can be modeled in terms of an *information need*. That is, a user has an objective that can be satisfied by acquiring information that may be present in one or more members of some subset of documents in the collection. The 'information' sought by a user may not be informative in any formal sense — for example, a user may be searching for websites containing scatological jokes — but it is nonetheless the case that a user will find some documents in the collection more useful than others in meeting their objective. In this sense, it can be argued that the user is always interested in finding information of some sort.

Viewed from this perspective, the role of an information retrieval system is to minimise the effort expended by the user in meeting their information need.

How can this be achieved? The depth and subtlety of natural language, its semantic ambiguity, and the complex interactions between documents, coupled with the subjective and nebulous nature of a user's information need, combine to suggest that optimal performance at this task is impossible to define, let alone achieve. Furthermore, the definition above is descriptive rather than prescriptive, and thus provides no assistance in building an implementation. In order to ground retrieval in a more concrete task, most information retrieval research focuses on identifying documents with a high degree of *relevance*. By presenting the user with a set of relevant documents, it is hoped that they can satisfy their information need quickly by perusing these documents. While this is a simplification and abstraction of the base task of helping the user satisfy their information need expeditiously, it is hypothesised that the effectiveness with which a system retrieves relevant documents is highly correlated to the degree to which it helps the user meet their information need.

With the model described, one must answer the question: what does it mean for a document to be relevant? Despite seeming reasonably intuitive, relevance has proved particularly elusive to define. Mizzaro (1997) lists over 130 papers whose primary focus is the issue of relevance. Saracevic (1975) discussed the difficulties in defining relevance over three decades ago. An entire special issue of JASIST was devoted to relevance (Froehlich, 1994), one of its themes being an "inability to define relevance." Mizzaro (1998) attempts a four-dimensional taxonomy that enumerates dozens of different notions of relevance. Despite the vigorous discussion and debate, we are seemingly no closer to consensus on the issue, nor do we have a clear understanding of the relationship between the different notions of relevance, and the overarching goal of satisfying an information need.

Despite being less favoured by those researchers who have studied relevance in depth, *system-oriented* relevance remains the most commonly used formulation of relevance for practitioners in information retrieval. System-oriented relevance is perhaps the most abstract form of relevance, because neither the user nor their information need is explicitly modeled under this approach. System-oriented relevance is a matter exclusively between a document, D, and a query Q. D is relevant to Q if "there is substantial semantic overlap between the representations of D and Q"(Lavrenko, 2004, p. 4). The simplicity of this definition is particularly advantageous when evaluating the relative performance of various information retrieval systems. For more information on how system-oriented relevance is used for evaluation, see Section 2.3.5.

An assumption that is often made with regards to the system-oriented definition of relevance is that it is *binary*: either a document is relevant, or it is not. Under this assumption, there is no scope to say that one document is more relevant than another. Using a binary definition of relevance greatly simplifies the process of effectiveness evaluation. Furthermore, studies on graded relevance — in which judges are asked to give a score based on the degree of relevance a document has — have shown that there is little agreement between different judges on the relevance grade, whereas the agreement is far greater if relevance is binarised (Lavrenko, 2004, p. 6).

Because system-oriented relevance disposes of any reference to the complex and somewhat transcendental relationship between the user and their information need, instead focusing on the binary relation between a concrete document and a concrete query, it makes the task of information retrieval research significantly simpler. However, the degree to which the notion of relevance has been abstracted away from the user brings into question whether this simplified definition is meaningful in the context of the information retrieval task.

2.3.3 Ranked search

The recognition that a system for guiding a user towards relevant documents must explicitly deal with inherent uncertainties is generally credited to Maron and Kuhns (1960). Their *probabilistic indexing* system for concept-based indexing introduced fuzziness into the index, with each document given a number between zero and one for a particular concept based on the probability that the document will be relevant to a user interested in that concept. The result of a search in this system is a list of documents in decreasing order of probability of relevance to the user, based on the probabilities stored in the index.

The notion that returning documents in decreasing order of probability of relevance is an optimal arrangement is known as the *probability ranking principle.* Robertson (1997) most famously reproduces a formulation of the probability ranking principle as due to W.S. Cooper:

> “If a reference retrieval system’s response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data.”

The probability ranking principle underlies the search mechanism of most current information retrieval systems: documents are each assigned a score with relation to the query, and they are presented as a list in order of decreasing score.

When Maron and Kuhns proposed probabilistic indexing, the notion of probability of relevance was explicitly coded into the index. Modern retrieval systems use rich models and functions to produce scores by which documents are ranked; however, it is often unclear whether these scores can be legitimately interpreted as probabilities of relevance. For example, the commonly-used cosine score is based on an underlying measure of *distance* between a document and the query; it is not a probability at all. The query-likelihood score (Ponte and Croft, 1998) does produce a probability, but it is a generative probability based on a language model rather than a probability of relevance. In such cases it is difficult to see how the probability ranking principle is applicable.

2.3.4 Ranking models and the TF×IDF principle

The probability ranking principle provides a descriptive framework on which search systems are based. However, it provides no clue as to how the goal of ranking documents in decreasing order of probability of relevance might be achieved. This task is left to a ranking formula based on one of several *ranking models*, theoretical systems that attempt to model the relationship between a query and a document and thus, whether directly or indirectly, the probability that a document is relevant to a particular query.

Whichever model is used, a fundamental principle of document ranking is the TF×IDF term-weighting principle. This principle states that the contribution of a term to the relevance score for a document should be in some way proportional to the document's term frequency $f_{d,t}$ (TF) and inversely proportional to the number of documents in the collection in which it appears, the so-called document frequency f_t (IDF).

The first part of the formulation — TF — articulates the intuition that the more frequently a term occurs in a document, the more significant an indicator it is of the topic of the document. The second part — IDF — gives greater weight to terms that appear rarely across the collection and downgrades the significance of terms that appear frequently. This can be justified in a number of ways, such as by asserting that rare terms provide greater discriminating power between relevant and nonrelevant documents, or that rare terms are more likely to be signifiers of topic than common terms, which often play functional rather than topical roles within natural language. Croft and Harper (1997) also show that IDF emerges as a term-weighting parameter within the probabilstic modeling framework (see Section 2.3.4) in situations where no relevance information is known.

Witten et al. (1999) argue that the traditional TF×IDF formulation is overly strong, and propose a weaker set of 'monotonicity constraints' in the same spirit:

> "A term that appears in many documents should not be regarded as being more important than a term that appears in a few, and a document with many occurrences of a term should not be regarded as being less important than a document that has just a few."

We now briefly describe the three most commonly-used ranking models: the vector-space model, probabilistic modelling, and language modelling.

Vector-space model

The vector-space model treats each query and document as a single vector in natural-language space. In order to estimate the probability of relevance of a document to a query, the 'closeness' of the vector representing the document and the vector representing the query is computed using one of a range of standard scalar measures for vectors.

In practice, queries and documents under the vector-space model are $|V|$-dimensional vectors, where V is the collection vocabulary, the set of all terms in use in the collection. Each dimension corresponds to a particular term in the vocabulary, and the size of the vector for a document d in a dimension t is determined by the term weight $w_{d,t}$. As discussed above, a term's weight is most often a product of its TF and IDF.

An appealing candidate for quantifying the similarity of the document and query vectors is the dot product, or inner product:

$$\mathrm{dp}(q,d) = \sum_{t \in V} w_{q,t} \cdot w_{d,t} \tag{2.1}$$

Because the dot product can be geometrically interpreted as being the product of the length of one vector and the length of the projection of the other vector, it emphasizes similarity in vector direction. The more aligned a pair of vectors, the greater their potential for a large dot product.

The dot product metric is capable of producing adequate results in practice when used for information retrieval. However, it is biased; because the dot product is the product of vector lengths, it will — all other things being equal — be higher for vectors of a greater magnitude. Thus, the dot product metric tends to favour longer documents at the expense of shorter relevant documents.

The tendency of the dot product metric to favour long documents can be corrected by the application of an appropriate document-length normalisation factor. However, this has no theoretical basis. Hence, we use the *cosine distance* measure, which allows for document length to be normalised in a principled manner. The cosine distance measures the cosine of the angle θ between the query vector and the document vector. Because the $\cos(0°) = 1$ and $\cos(90°) = 0$, the cosine measure is maximised when the vectors are parallel and minimised when they are orthogonal. Thus, cosine is a measure of the degree to which the query and document vectors point in the same direction. The general formula for the cosine distance between a query q and a document d is as follows:

$$\text{cosine}(q, d) = \frac{\sum_{t \in V} w_{q,t} \cdot w_{d,t}}{W_q \cdot W_d} \tag{2.2}$$

where W_d and W_q are the document weight and query weight respectively, defined as the Euclidean distance of the respective vectors from the origin.

Because the cosine distance measures the angle between vectors, it is insensitive to their length. The functional interpretation is that the value W_d acts to normalise for document length. Thus, it does not suffer from the problems faced by the other vector-space metrics with regards to document-length bias.[2] In the body of published empirical evaluations over the years (in particular TREC; see Section 2.4) the cosine measure is generally shown to be reasonably effective, although somewhat inferior to leading metrics from the probabilistic modelling and language modelling paradigms.

Probabilistic modelling

Probabilistic modelling (Robertson and Sparck Jones, 1976) is an attempt to apply the probability ranking principle directly to the scoring process. Thus, the probabilistic modelling framework tries to estimate $P(r|d)$, the probability that a particular document is relevant to the user's information need.

The original similarity measure based on the probabilistic modelling approach required that one be in possession of a reasonable estimate of the number of relevant documents, and of the term distribution amongst the set of relevant documents (Robertson and Sparck Jones, 1976). In the absence of this knowledge, the model reverts to a rather simple IDF-based approach. Since then, the similarity metric has been enhanced by the addition of factors for term weighting, document length normalisation, and query-term weighting. Thus, probabilistic modelling remains of interest for pragmatic reasons: several similarity measures issuing from the probabilistic modelling framework show competitive effectiveness when compared to other state-of-the-art measures. In particular, the Okapi BM25 similarity measure (Robertson et al., 1995) is considered amongst the most effective of state-of-the-art similarity measures. The BM25 measure scores documents using the following formula:

[2]Singhal et al. (1996) contend that the normalisation introduced by standard cosine distance in fact penalises longer documents. The pivoted document-length normalisation measure is thus introduced to normalise for length in a more unbiased manner.

$$S(d,q) = \sum_{t \in q} \left[\log \left(\frac{N - f_t + 0.5}{f_t + 0.5} \right) \times \frac{(k_1 + 1) f_{d,t}}{k_1 \left((1 - b) + b \cdot \frac{|d| N}{\sum_{i \in N} |d_i|} \right) + f_{d,t}} \times \frac{(k_3 + 1) f_{q,t}}{k_3 + f_{q,t}} \right] \tag{2.3}$$

The three factors in the equation can be summarised as weighting each term according to its frequency in the collection (the IDF factor), according to its frequency in the document (the TF factor), and according to its frequency in the query (pertinent to situations where the query is lengthy). As well as using collection size N, term frequency f_t, document term frequency $f_{d,t}$, and query term frequency $f_{q,t}$, BM25 contains the parameters k_1, b, and k_3. These can be adjusted or tuned to suit the characteristics of the collections being searched, though ranges of recommended values have been published (Sparck Jones et al., 2000).

Ranking formulae such as Okapi BM25 bear little superficial resemblance to the original principles of probabilistic modelling. Nonetheless, BM25 consistently demonstrates excellent performance in practice, and for this reason the probabilistic modelling approach is still in widespread use.

Language modelling

Language models are generative statistical models that are designed to simulate the process by which natural-language text is created. These models can have various degrees of sophistication and fidelity, depending on the application and the data available. Language modelling has seen longstanding use in diverse areas including speech recognition (Bahl et al., 1990) and automated translation (Brown et al., 1990). Its introduction into the field of information retrieval can be traced to Wong and Yao (1989), though it was popularised and brought to its current form by Ponte and Croft (1998), with the introduction of the *query likelihood* approach to document ranking.

Language models define a probability distribution for the next term to appear in a stream given all the terms that came before it:

$$P(t_n | t_1 \cdots t_{n-1}) \tag{2.4}$$

There are no restrictions on the way in which a general language model may be formulated. In information retrieval, the models tend to be simple *unigram* models in which no consideration is given to the preceding text in estimating generation probabilities:

$$P(t_n|t_1 \cdots t_{n-1}) = P(t_n) \tag{2.5}$$

The generative probability of a particular document D under a unigram model is simply

$$P(D) = \prod_{t \in D} P(t) \tag{2.6}$$

The main reason, besides simplicity, that unigram models are favoured over more complex and realistic language models is the issue of *data sparsity* in model prediction. The more complex the model, the larger the sample required to accurately parameterise it. As we are often dealing with relatively small samples in information retrieval it is not feasible to work with higher-order models.

To estimate the model used to generate a document D, the language modelling approach treats the document as a sequence of samples from the model. These samples can then be used to calculate the document's *maximum likelihood estimator* — the distribution that yields the highest $P(D)$ of any possible distribution. In the case of unigram models the maximum likelihood distribution is built by dividing the probability mass evenly amongst all the terms in the document:

$$P(t|d) = \frac{f_{d,t}}{|d|} \tag{2.7}$$

The query likelihood measure starts with the conjecture that the likelihood of a document being relevant to the query is correlated to $P(d|q)$, the generative probability of the document given the language model used for generating the query. That is, if a document is likely to be generated by the model the user was using to create the query, it is likely to be topically aligned with, and thus relevant to, the query.

In a typical keyword search, the query may only be two or three terms in length. We are thus faced with an extreme data sparsity problem; after all, a language model induced over a two-word query will be wildly inaccurate. This problem is addressed by using Bayes's theorem to calculate the value indirectly, as follows:

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)} \tag{2.8}$$

Because $P(q)$ remains constant for any given query, it can be removed from consideration:

$$P(d|q) \propto P(q|d)P(d) \tag{2.9}$$

$P(q|d)$, the generative probability of the query from the language model used to generate the document, is the *query likelihood.* If $P(d)$, the *document prior*, is assumed to be uniform then we have

$$P(d|q) \propto P(q|d) \tag{2.10}$$

If we assume a uniform prior, we have $P(q|d)$ being a constant factor of $P(d|q)$. This means that ordering documents by $P(q|d)$ will produce the same ranking as if they were ordered by $P(d|q)$. In the practical setting this is of enormous benefit as in general we have a much larger sample from which to predict the language model of a document than that of a query.

Although using the language model induced from a document greatly alleviates the data sparsity problem faced when trying to estimate $P(d|q)$ directly, the problem does still exist. A document (particularly if it is short) remains a relatively small sample from which to estimate the entire language model. This results in a 'jagged' distribution in which the entire probability mass is concentrated on those terms that happened to appear in the sample. A maximum-likelihood distribution asserts that only those terms that have occurred are in fact possible. This is implausible in the case of documents; just because a word has not occurred in a document doesn't mean it couldn't have occurred.

From a practical perspective, the maximum-likelihood distribution results in undesirable behaviour if one of the query terms does not occur in the document. As shown in Equation 2.6, the generative probability of a query is calculated as a product of the individual probabilities for each term. Thus, if a query term t does not occur in a document, a zero is introduced into the product and $P(q|d)$ goes to zero, even if many of the other query terms occur frequently in the document.

In order to deal with the consequences of data sparsity, the maximum-likelihood distribution is generally *smoothed.* Smoothing is a general term for techniques in which some of the probability mass is redistributed away from the jagged probability peaks of a distribution in order to compensate for an inadequate sample size. The most effective smoothing techniques are those that take advantage of the fact that there is further knowledge about the document model as well as the sample represented by the document itself. While it is possible to directly incorporate knowledge about natural language into the model, an easier and more robust technique is to treat the entire document collection as a larger sample of a more generalised language model, known as the *background model.* While the background model does not capture topicality in the way that a model induced over just the document does, it contains far more data about the way in which the language is used.

The Jelinek-Mercer smoothing technique (Zhai and Lafferty, 2004) builds a simple *mixture model*, linearly combining the document and background models using a parameter λ.

$$P(w|d) = (1 - \lambda)\hat{P}(w|d) + \lambda\hat{P}(w|c) \tag{2.11}$$

where $\hat{P}(\cdot|d)$ and $\hat{P}(\cdot|c)$ are the maximum-likelihood document and background models respectively. In this way, all words appearing anywhere in the collection have a nonzero generation probability within the document model.[3] While Jelinek-Mercer smoothing is a simple and effective baseline smoothing technique, it is outperformed by some more sophisticated smoothing techniques, notably Dirichlet smoothing (Zhai and Lafferty, 2004). Research is this area is ongoing.

Query likelihood is the original and most commonly used ranking model within the language modelling paradigm, but there are other choices. These include translation models (Berger and Lafferty, 1999) and relevance models (Lavrenko, 2004).

2.3.5 Effectiveness evaluation for ranked search

In building systems that attempt to quantify the degree of relevance a document has to a query, we encounter what is known as the *semantic gap.* This principle is an acknowledgment that the task of moving from purely symbolic expressions to semantic notions — for example, from the text of a document to its meaning — is far beyond the current state of the art in computer science. Thus, we cannot hope to accurately capture the proper relationship between an information need and relevant content — both semantic notions — given only textual representations of query and documents. In fact, the problem is even more fundamental: a system performing a ranked search task is acting on incomplete information. The issue is not just that a query is a textual representation of a semantic concept, but that it is an *incomplete* representation of that concept. Even a meticulous human expert could not hope to retrieve the set of documents that best meets a user's information need, for the simple reason that the information need is not fully specified.

Under such circumstances, it is unrealistic to expect that an algorithm or system will return the precise set of documents that are considered relevant to the user. It thus becomes

[3]Mixture-model smoothing also implicitly introduces an IDF factor into the query likelihood ranking process. Because the background model will contain higher generation probabilities for commonly occurring words than for rarely occurring words, a document lacking a rare query term will be penalised more heavily in the ranking process than a document lacking a more common query term.

pertinent to evaluate not just the efficiency, but also the *effectiveness* of a system. That is, we must quantify the degree to which a system has succeeded in its task of retrieving relevant documents.

A vast range of evaluation metrics and methodologies have been proposed. Broadly, evaluation is conducted either in a user-oriented or a system-oriented fashion. In the former, an attempt is made to directly measure the level of satisfaction of real users on different systems. In system-oriented evaluation, the evaluation is directly on the system output using some model of user behaviour. User-oriented evaluation has been marginalised to some degree by the fact that meaningful experiments can be expensive to conduct, while system-oriented evaluation is accused of being too abstract to be meaningful (Saracevic, 1995). Nonetheless, system-oriented evaluation represents the most widespread form of information retrieval evaluation. Within this context, the dominant methodology for several decades has been based on the Cranfield paradigm (Cleverdon, 1991; Voorhees and Buckley, 2002).[4]

The Cranfield paradigm specifies a comprehensive methodology for the system-oriented evaluation of the effectiveness of an information retrieval system. Under the Cranfield Paradigm, a static corpus of documents of an appropriate type is specified, as well as a set of queries. For each query, the documents in the test collection are manually judged for relevance to that query. Taken together, the corpus, queries, and relevance judgements form a *test collection*. A search system is then evaluated using the confliting metrics of *recall* and *precision*. Given a set of documents returned in response to a certain query, the recall is the proportion of all documents relevant to the query that appear in that set. Precision is the proportion of documents appearing in the set that are relevant to the query.

The conflict between the two measures becomes apparent when one considers the probability ranking principle. As one descends the result list, the confidence that each subsequent document is relevant decreases; thus, one expects that (on average) precision is maximised by returning only the single document with the highest probability of being relevant as determined by the search engine. Conversely, recall increases monotonically as the size of the result list is expanded; after all, adding extra documents can never lead to a reduction in the number of relevant documents in the list.

Recall and precision operate on the assumption that a fixed-size set of results has been returned by the search engine. Furthermore, they do not take account of the ordering of results in the ranked list. In general, these are neither necessary nor desirable assumptions

[4]Though details have changed over the years, primarily to accommodate larger corpora, the vast majority of modern system-oriented evaluation methodologies can trace their lineage back to the Cranfield paradigm.

in the domain of ranked search. The task of a ranked search system is to impose an ordering on the document collection in accordance with the probability ranking principle. A user may choose to browse as far down the ordered list as they wish until the entire collection has been inspected. It is not for the system to decide a threshold below which no more documents need be returned. While recall and precision are fundamental principles of information retrieval, they are not in and of themselves good measures of performance for a ranked search system.

One way of removing the artificial need for a fixed-size result set is to graph both recall and precision against the number of documents retrieved. Thus, the overall performance of the system can be observed as one goes further down the list. The dichotomy between recall and precision discussed above, in which one is expected to decrease as the other increases, suggests a simpler alternative presentation of the same data: plotting precision as a function of recall.

Recall-precision curves provide a wealth of data about the effectiveness of a system in producing ranked lists. However, they suffer from a major weakness: they do not represent, in the general case, an ordering on the quality of ranked lists: "there is no absolute sense in which one can say that one particular pair of precision-recall values is better or worse than some other pair" (van Rijsbergen, 1979, page 129). Inasmuch as a major aim of effectiveness evaluation is to be able to discriminate between two retrieval strategies, to be able to say 'A is better than B', it is useful to be able to reduce the effectiveness of systems to a single figure. In this way, it is possible to impose an ordering on the effectiveness of a range of systems, making for much easier interpretation of results. We now present a summary of some of the more common measures for ranked lists; all but one are based directly on recall and precision.

Precision at n Often denoted as P@n, one simple expedient is to nominate an arbitrary cutoff in the ranked list at which precision is measured, and for that value to be the representative score for a ranked list. For example, P@10 is the proportion of the first 10 documents in the ranked list that are relevant to the query. Recall is not explicitly measured in P@n, but the observation holds that a higher value also represents a higher level of recall.

$\mathbf{F}_1$ measure The F measure was proposed by van Rijsbergen (1979) as a way of combining a recall-precision pair into a single value. The most commonly-used form of the F measure is the F_1 measure, which is a simple harmonic mean of the recall and precision values:

$$\mathrm{F}_1 = \frac{2rp}{r+p}$$

The F_1 measure is provides a theoretically motivated, unbiased way of combining recall and precision into a single value. However, the reduction in dimensionality renders interpretation of F_1 difficult; a single F_1 value forms a contour line across the space of recall–precision values: it is thus impossible to determine whether a system producing a given F_1 value has a high recall with low precision, or vice versa. Similar criticisms apply to many of the summary measures that follow.

n-point average interpolated precision One way of representing a recall-precision curve as a single point is to take the area under the curve, which can be estimated by sampling the curve at intervals and taking a mean of the precision values observed. The most common variant of this measure is 11-point average interpolated precision. In this variant, precision values are sampled at recall values of 0%, 10% and so on in intervals of 10% up to 100% recall. The average of these values is then the representative value for a particular run.

Interpolation of precision simply means that the precision value at a given point is the greater of the actual value at that point and all values for higher levels of recall. This practice removes noisy jitter and enforces monotonicity on the recall-precision curve.

Average precision Non-interpolated average precision (Buckley and Voorhees, 2000), generally referred to as average precision or AP, is the official metric of many of the tracks in TREC (see Section 2.4), and is consequently one of the most popular metrics in use today. The average precision of a ranked list is calculated by summing the precision of the list at each relevant document and dividing by R, the total number of known relevant documents for the query. When average precision is used to score multiple runs and the average score taken, this is usually known as mean average precision or MAP.

A criticism that is made of AP is that, while it exhibits powerful discriminating characteristics, it presents no clear user model (Buckley and Voorhees, 2004). It has thus far been impossible to describe a sensible model of user behaviour for which the measure of result list quality would resemble average precision. Another criticism is that AP is unstable in the presence of incomplete relevance judgements; given the insertion of additional judgements into the pool, the AP of a run could either increase or decrease in an unbounded fashion.

R-Precision This measure takes the precision at R, where R is the total number of relevant documents for the query in the judged set. R-precision is also know as P@R. Aslam and Yilmaz (2005) provide an analysis of R-precision explaining and supporting the commonly observed correlation between R-precision and AP.

Reciprocal Rank The reciprocal rank of a result list takes on the value $\frac{1}{n}$ where n is the rank of the first relevant document in the list. When reciprocal rank is used to score a number of runs and the average score taken, this is usually known as mean reciprocal rank or MRR.

bpref Due to the vast size of modern document collections, evaluation takes place on sets of judgements that are unavoidably incomplete. Thus, the issue of the *robustness* of an evaluation metric — the degree to which it degrades in accuracy as relevance judgements become less complete — becomes pertinent. Buckley and Voorhees (2004) define a new measure, bpref (for *binary preference*), that is claimed to be more robust under incomplete judgements. bpref is calculated by determining the proportion of judged nonrelevant documents that appear before each judged relevant document in a result list, and averaging these values. Buckley and Voorhees perform a number of experiments demonstrating the superior robustness of bpref, though the experiments are artificial and it is uncertain how applicable the results are to more realistic situations.

2.4 TREC

TREC, the Text REtrieval Conference[5] (Harman, 1993; 1995) has run annually since 1992 and is coordinated by the United States National Institute of Standards and Technology (NIST). Intended as a forum at which IR researchers could compare the approaches and effectiveness of their search software, the contribution of TREC, as well as the conferences themselves, has been an ever-growing archive of freely-available collections, query sets and their corresponding relevance judgements. This archive has provided the IR community with a common testbed for the comparison and evaluation of retrieval systems, and it is used far more broadly that merely in the context of the conference itself. In this thesis, we use data from the TREC initiative for the evaluation of a number of phenomena.

TREC is organised into various *tracks*, subareas that emphasise different information

[5] `http://trec.nist.gov`

retrieval tasks, or different aspects of those tasks. In this section, we provide brief descriptions of those tracks most directly related to our work.

2.4.1 TREC *ad hoc* search tracks

The *ad hoc* retrieval task comprises what is most often thought of as the search task: returning a ranked, document-oriented response to queries that are formulated on an *ad hoc* basis.

Over the duration of the TREC initiative, there have been several tracks that have focused on various aspects of the *ad hoc* search task.

***Ad hoc* track** The TREC *ad hoc* track ran from 1992 to 1999 and focused on the *ad hoc* search task on various collections of data sourced from news agencies such as Associated Press (AP) and the Los Angeles Times. The task was to return a ranked list of documents for each of 50 topics representing various specific information needs. Each topic consisted of a brief title, a description of the characteristics of a relevant document, and a more detailed narrative describing in precise details the nature of the information need. Submitted runs were then evaluated using various recall–precision based metrics. For a more detailed description of the task mechanics, see Harman (1993) or Harman (1995).

The *ad hoc* track focused generically on the *ad hoc* search task. As it became clear that various aspects of the task required specific attention, other *ad hoc* search tracks have been integrated into the TREC program. These tracks are described below; in general, they follow the same methodology as the *ad hoc* track.

Web track The web track (Craswell and Hawking, 2002) has been running since 1999 and is designed to evaluate the performance of *ad hoc* search systems over data drawn from the web. Web data has characteristics that make it significantly different to the collections of newswire data used for the *ad hoc* track. The fact that most web documents are created in the hypertextual, semi-structured HTML format provides rich possibilities for new ranking methodologies. The hypertextual nature of HTML data allows for *link analysis* (Kleinberg, 1999), which has proved successful as an ingredient in the ranking mix (Brin and Page, 1998; Craswell et al., 2001). The semi-structured nature of the data allows for novel weighting schemes, such as according terms in the page title greater weight, or giving those terms that appear in a large font greater weight than those appearing in a smaller font.

A collection of web data, WT100G (for Web Track 100 Gigabytes) and two subsets, WT10G and WT2G, were created for the Web track. The *ad hoc* task — as well as several other web-oriented IR tasks — was evaluated over this data, with systems encouraged to take advantage of web-specific optimisations. In 2005, the web track was merged into the terabyte track (see below).

Terabyte track The massive volume of data present on the world-wide web and in many government and corporate document repositories present serious challenges to search engine creators. Search techniques that work for small document collections may not scale well to large collections, both in terms of efficiency and effectiveness. The terabyte track (Clarke et al., 2004) is an *ad hoc* web track designed to evaluate search in the context of large terabyte-level collections. One of the legacies of the terabyte track is the GOV2 collection, a crawl of the `.gov` (U.S. Government) domain. While the aim had been to amass one terabyte of data, this was not achieved. Nonetheless, the GOV2 collection — at 426 GB and containing over 25 million documents — is significantly larger than any other collections openly available to research institutions.

Robust track At the various *ad hoc* tracks it has been observed that some queries are much harder than others, in the sense that many search systems catastrophically fail to find many (or any) relevant documents for such queries, often despite good performance on many other queries in the query set. From a user perspective, this is seen as particularly undesirable. It is hypothesised that a catastrophic failure in response to one query is not adequately compensated for by excellent performance on another. It is therefore important that a search system demonstrate the property of *robustness*. The focus of the TREC Robust track (Voorhees, 2005) is on improving the robustness of information retrieval systems.

The Robust track uses a standard *ad hoc* methodology, simply choosing some of the worse-performing topics from the *ad hoc* track. Thus, rather than optimising search systems to perform well on average across a range of queries, it became necessary in the Robust track to focus on points of failure. In the 2005 Robust track a new official evaluation metric, gMAP — for geometric MAP — was introduced. This metric was considered to be more suitable given the track emphasis, as taking the geometric mean of the AP rather than the standard arithmetic mean leads to a greater penalty for inconsistent performance. Thus, gMAP encourages participants to focus on the worst-case performance of their systems.

Pooled judgement

All the *ad hoc* tracks are evaluated using various of the recall–precision measures described in Section 2.3.5, and thus require a set of relevance judgements against the collection. That is, they must know which documents in the collection are relevant with respect to a particular query, and which are not. In general, the only way to reliably extract a set of relevance judgements for a query is for a human judge to go exhaustively through the target collection and make a determination for each document of whether it is in fact relevant to the query in question.

Exhaustive judgements were possible (though expensive) on the smaller collections used in earlier evaluation efforts such as those at Cranfield (Cleverdon, 1991). However, modern collections are enormous. The TREC GOV2 collection contains over 25 million documents. It is simply not feasible for a group of human judges to accurately evaluate this vast number of documents.

The solution used in TREC is *pooling* (Sparck Jones and van Rijsbergen, 1975). Rather than judging all the documents in the collection, the top n documents for each submitted run are added to a judgement pool; in TREC n is typically 100. Only those documents appearing in a judgement pool are manually inspected; all other documents are typically assumed to be irrelevant (although this is not the case for the bref measure). By pooling documents from a variety of search methodologies, it is hoped that not only will it be possible to calculate accurate effectiveness scores for the submitted runs, but to reuse the incomplete evaluations on other runs that were not part of the pooling process, without distorting relative effectiveness assessment (Voorhees and Buckley, 2002). Sanderson and Zobel (2005) found that, while many relevant documents had most probably not been judged, the collections did seem to demonstrate robust reusability. Further efforts have been made at finding measures that are robust in the face of pool incompleteness (Buckley and Voorhees, 2004) and modifying the pooling procedure in order to get better coverage of relevant documents (Zobel, 1998; Cormack et al., 1998).

2.4.2 TREC novelty track

In the general *ad hoc* search tracks, the notion of relevance adhered to the simplified definition given in Section 2.3.2. In particular, relevance is assumed to be a relation solely between a document and a query. Inasmuch as relevance is supposed to reflect utility in some sense, this is not a very realistic limitation: the relevance of a document to a user is a function not just of the document itself but of the state of the user. If an item of information is already known to the user — that is, the information is not *novel* — it loses a great deal of value.

While it is unfair to expect a search system to be able to know or predict the state of a user's knowledge external to their interaction with the search system, it is more reasonable to attempt to keep track of how the stream of results alters the state of the user's knowledge. In particular, it is reasonable to attempt to keep track of what information has already been imparted to the user so that redundant information is not repeated. The task of a search system is thus extended from differentiating between relevant and non-relevant (in the traditional sense) documents to additionally differentiating between novel and redundant information. This model was first explicitly expressed by Zhang et al. (2002).

The TREC novelty track (Harman, 2002; Soboroff and Harman, 2003; Soboroff, 2004) ran from 2002 to 2004 with the aim of promoting techniques that return a stream of relevant and novel information to the user. Though the task varied slightly over the three years it was run, the loose outline is this: given an ordered stream of sentences, return a sentence only if it is both relevant *and* novel.

In order to evaluate the performance of systems at the novelty task, topics were chosen from previous *ad hoc* tasks. For each of these topics, 25 documents from the relevant set were selected, ordered, and segmented into sentences. Judges were required to first traverse the list of sentences and mark those that they considered relevant to the topic. They then had to look through the set of relevant sentences they had selected and mark them as novel or redundant with respect to the sentences that had preceded them. The recall and precision of runs relative to the judged set of relevant novel sentences was evaluated. To allow a sensible averaging of results across the different topics, the F_1 measure (van Rijsbergen, 1979) was used as the evaluation measure.

In later years, the track was divided into several subtracks, allowing participants to focus exclusively on finding novel sentences as well as the full novelty task, and providing varying amounts of training data.

The results from the novelty track were not overwhelmingly strong. The formulation of the task has been criticised as providing meaningless or misleading results. For example, in order to get a sufficient volume of novel relevant sentences, the judgement sets consisted of sentences solely extracted from judged relevant documents. Allan et al. (2003) found that this bias in building test collections leads to a massive overestimate of the effectiveness of novelty-detection systems. More disturbingly, they found that relative effectiveness orderings were not preserved; in fact, systems that performed better when only relevant documents contributed to the judgement set tended to do worse when a mix of relevant and nonrelevant documents were used. This calls into question the degree to which the results of the novelty track can be interpreted to be indicative of real-world performance.

2.5 Distributed search

The information retrieval process as we have described it thus far adheres to a *single-database* model. It is assumed that the documents being searched are aggregated into a single collection that can be indexed and searched as a monolithic system. By allowing the information retrieval system to have full access to the data being searched, the single-database model makes the information retrieval task simpler and more accurate. However, there are circumstances in which this model is inappropriate or impossible: when data is physically dispersed and bandwidth is limited, or when content owners only make documents available via a search interface for copyright, operational, or privacy reasons. When wishing to search amongst such documents, it becomes desirable or necessary to use the *multi-database* model that is the basis of distributed information retrieval (Callan, 2000).

In the multi-database model of information retrieval, there is no central aggregated collection: rather, the documents composing the 'collection' are distributed amongst a number of servers, each of which presents a standard search interface. The user transacts with a system known as a *broker*, which is responsible for dispersing the query and aggregating results into a standard ranked list. The distributed information retrieval environment is usually characterised as being either cooperative, in which servers provide the broker with useful information such as collection statistics, or uncooperative, in which the servers only make available the bare search results. In either case, the broker must perform two main tasks in order to effectively help the user meet their information need: *resource selection* and *result merging*.

In distributed information retrieval it is assumed that it is not in general desirable or possible to query all databases on the network. Thus, when a query is presented to the broker, it must select a subset of the available databases to which to forward the query (Si and Callan, 2003a; 2004). The aim is to select the servers that are most likely to return relevant documents in response to the query. In order to make a decision on which servers to select, resource selection techniques make use of a *representation set* for each server. A representation set is a data structure describing the contents of the collection maintained by the server, typically detailing standard collection statistics. In a cooperative context, servers make their representation sets available directly to the broker. In an uncooperative setting, the broker must use query-based sampling techniques to estimate the representation set (Callan and Connell, 2001).

In response to the issuing of a query to the selected servers, the broker will receive a series of ranked lists. The results merging task consists of interpolating these lists in such a way as to maximise utility in accordance with the probability ranking principle. In a cooperative setting in which ranking scores are provided to the broker and representation sets are available, various techniques exist to normalise these scores to effect the merger of lists (Callan et al., 1995; Craswell et al., 1999). In the absence of such data, result merging must proceed based on the rankings assigned by the individual systems, in combination with sample data that has been collected by the broker (Si and Callan, 2003b).

2.6 Index structures for fast text querying

Searching for short strings within a large volume of text is a fundamental operation within computer science, and is particularly important for information retrieval. For a query string of length m and a collection of size n, the naïve approach to this task is to compare the query string against every substring of length m in the collection. Such an approach would take $O(mn)$ operations. Various more sophisticated methods for scanning the collection text can reduce the average-case complexity to $O(n)$.

Nonetheless, for large n, linear scanning of the collection text — even with the more refined $O(n)$ algorithms — is extremely slow. If a collection is 100 GB in size, any linear search algorithm must read the entire collection in from disk. This is not acceptable for a high-performance search system.

In this section, we discuss two index structures — suffix structures and inverted indexes — that reduce search for short substrings to speeds that are sublinear in n.

2.6.1 Suffix trees and suffix arrays

Suffix trees and suffix arrays (known, together with their variants, as *suffix structures*) are a class of index structures that allow quick access to all suffixes of a string (Gusfield, 1997). Suffix structures allow fast solutions to a wide variety common problems in string processing, such as string matching, the longest repeated substring problem, and the longest common substring problem.

A suffix tree for a string is in fact a trie constructed from every suffix in that string.

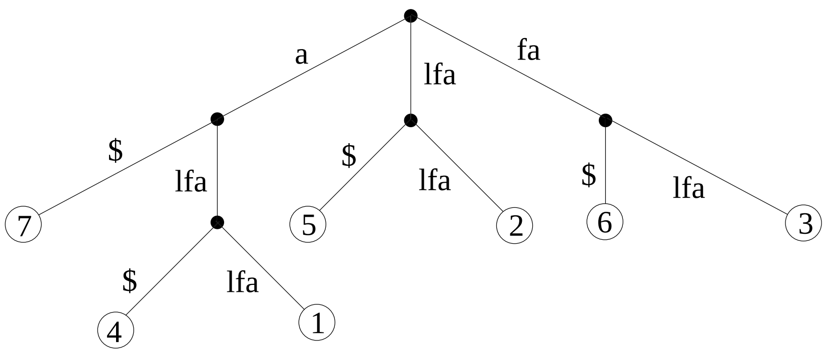


Figure 2.3: An example suffix tree for the word 'alfalfa'. The $ character represents the end of string, and the number in the leaf node represents the starting position of the suffix represented by that node.

Figure 2.3 shows the suffix tree constructed from the string *alfalfa*. Each vertex of the trie is labeled; concatenating the labels of each vertex traversed in a root-to leaf traversal produces a suffix of the string. The offset of that suffix into the string being indexed is labeled at the leaf. For example, in Figure 2.3, traversing from the root along the middle branch and then taking the right-hand branch produces the string *lfalfa* and terminates at a leaf node labeled with the offset 2. This indicates that the suffix *lfalfa* begins at an offset of 2 in the original string *alfalfa*.

In addition to many other uses, suffix trees are capable of returning offsets to all occurrences of an arbitrary query string in a time proportional to the length of the query. This can be effected by traversing along the query string from the root. If the traversal cannot be completed, the query string does not have a match; otherwise, the leaf nodes of the subtree of the node at which the traversal terminates contain pointers to the occurrences of the query. This index-lookup task is fundamental in information retrieval.

Another problem that can be efficiently solved by suffix trees is that of finding repeated strings of a certain length. The tree must be traversed in order to gather the subtrees that occur at a depth equal to the length being searched. Leaf-nodes occurring on the same subtree will point to repeated strings of at least the specified length.

Suffix trees are extremely powerful and flexible. However, in the form described, they are in-memory structures that consume an amount of memory that is significantly greater than the size of the text being indexed: for a string of size n, Manber and Myers (1993) report space usage of at least $16.6n$ for real data samples. After thorough optimisation, Kurtz (1999) managed to reduce this to an average-case of $10.1n$. This is still an onerous

`a[0]`	`7` (a)
`a[1]`	`4` (alfa)
`a[2]`	`1` (alfalfa)
`a[3]`	`6` (fa)
`a[4]`	`3` (falfa)
`a[5]`	`5` (lfa)
`a[6]`	`2` (lfalfa)

Figure 2.4: An example suffix array for the word 'alfalfa'. Sequentially accessing the array returns the position of suffixes in lexicographic order.

overhead, and limits the utility of suffix trees for larger amounts of data.

Suffix arrays are alternative structures that can be used instead of suffix trees for many applications — including those described above — but consumes less memory for a string of a given size (Manber and Myers, 1993). A suffix array consists of an array of pointers into the string being indexed. The pointers are arranged in such a way that a linear traversal of the suffix array will encounter the suffixes in lexicographic order. A suffix array for the string *alfalfa* is presented in Figure 2.4. The sorted order of the array means that all occurrences of a given substring will appear adjacently, and can be located using a binary search.

In its standard form, a suffix array consumes $5n$ space in the size of the source, a significant improvement over suffix trees. Further developments have resulted in compressed forms of suffix arrays that are able to consume significantly less memory that the standard form. Some *self-index* representations such as the FM-index (Ferragina and Manzini, 2000) are able to consume space less than n, effectively compressing the text and indexing it at the same time. In general, the more compact the representation, the slower it is.

There have recently been developments in techniques for the construction and representation of suffix structures on disk (Tata et al., 2004; Cheung et al., 2005). However, external suffix structures are still characterised by relatively long construction times, and either large on-disk size or slow search speed. For these reasons, suffix structures are not the first choice for indexing large volumes of data. A much more efficient (though less flexible) data structure that is appropriate for many information retrieval tasks is the inverted index.

2.6.2 Inverted indexes

An *inverted index* (Witten et al., 1999; Zobel and Moffat, 2006) is a data structure for the fast lookup of features in a collection of objects. Inverted indexes play a central role in information retrieval and related areas as the most important structures for the resolution of queries.

Inverted indexes are similar to the traditional indexes that appear at the back of many books. Much as a traditional index allows a reader to locate all uses of a particular term or phrase without having to browse through the whole book, an inverted index rapidly returns a list of locations within a collection where a particular feature occurs.

From the perspective of an abstract data structure, an inverted index is designed to primarily support a single operation: given a *feature descriptor*, return the list — known as the *postings list* — of all locations in the collection where that particular feature occurs; some auxiliary data may also be returned. In the most common applications of inverted indexes, the feature descriptor will be a natural language term, much as is the case with traditional indexes. However, note that arbitrary features can be indexed, so long as they have associated with them some notion of location.

While modern high-performance inverted indexes can have sophisticated implementations in order to improve performance and scalability (Zobel and Moffat, 2006), the inverted index is fundamentally a simple structure. In general, an inverted index consists of two sub-structures: the vocabulary and the postings lists.

The vocabulary is a directory of all the feature descriptors in the index, organised so as to facilitate high-speed (preferably constant-time) random access. Each entry in the vocabulary entry optionally has associated with it some feature statistics — most commonly a feature occurrence count, f_t — and a pointer into the file of inverted lists indicating the beginning of the postings list for that feature.

A postings list should contain a series of entries indicating the location of each occurrence of the feature. The exact composition of each entry is dependent on the specific use to which the index is being put. In the classic application of indexing terms in a document collection, a minimal postings list contains d, the document identifier, and $f_{d,t}$, a count of the number of times that term appears in the document. A finer-grained index might also contain the offset within the document of each term occurrence. Other applications may contain completely different identifying data depending on the nature of the features being indexed; the only requirement is that the entry should give some indication of the *location* of the feature.

```
con--quest
|||  ||| |
consequent
```

Figure 2.5: An optimal alignment between the two strings 'conquest' and 'consequent'. The dashes represent gaps, and the vertical bars represent matches in the alignment. In this case the LCS is 'conquet'.

In the context of information retrieval, the features being indexed are generally collection terms. For this task, a carefully implemented inverted index generally consumes less than one-fifth of the collection size and can be constructed at a rate of several gigabytes per hour on a standard workstation (Heinz and Zobel, 2003). Note that inverted indexes cannot be used for querying of partial-terms, or in situations where there is no such concept as a term. These restrictions do not apply to suffix structures.

2.7 Approximate string matching

Approximate string matching addresses the task of finding matches between pairs of strings that are similar, but not identical. Approximate string matching has applications in many areas including spelling correction (Glantz, 1957; Damerau, 1964), version control (Tichy, 1985) and computational biology (see Section 2.8).

Early work concentrated on practical and theoretical concerns surrounding solutions to the longest common subsequence (LCS) problem (Navarro, 2001). The LCS task is, given a pair of strings S and T, to find the longest string that is a subsequence of both strings.[6] Another way of viewing the problem is as finding the best *alignment* between the two strings: inserting gaps in the strings so that the maximal number of elements in the two strings match. The LCS is simply the string of characters that match under the best alignment. Figure 2.5 illustrates an alignment between the words 'conquest' and 'consequent'. The LCS, derivable immediately from the alignment, is 'conquet'.

Closely related to LCS is the Levenshtein distance (LD) or edit distance of a pair of strings. The LD measures the number of atomic string operations (insert, delete and substitute)

[6]Subsequences should not be confused with substrings. A subsequence may have gaps in it, whereas a substring may not. For example, `ape` is a subsequence of `apple`, but it is not a substring. Algorithms for dealing with the two are generally quite different.

required to transform a source string S into a target string T. LD and LCS are linked by a very simple formula: $\mathrm{LD}(S,T) = |S| + |T| - 2|\mathrm{LCS}(S,T)|$. Referring again to Figure 2.5, the Levenshtein distance can be calculated by summing the number of positions in the alignment that are not connected by vertical bars.

There are several approaches to solving the LCS/LD problem; for a full review see Navarro (2001). Here we discuss the dynamic programming approach.

When aligning a pair of strings, we can imagine that we have a pointer to the beginning of each. At each iteration, we can choose to advance the pointer in one or other of the strings (corresponding to an insertion or a deletion), or to advance it in both (corresponding to a match or a mismatch). We can do this computationally by choosing each of the three possible actions, summing the score incurred with the recursively calculated edit distance of the rest of the string, and returning the lowest of the three values. This process is, however, extremely inefficient as it repeatedly computes the same intermediate steps for many different permutations. Using a dynamic programming matrix such as the one in Figure 2.6 effectively collapses all the different recursive branches onto the one matrix, and requires only $O(|S| \times |T|)$ computations.

The dynamic programming matrix is filled from the top-left corner in either row-major or column-major order. A cell $C_{i,j}$ is updated according the following formula:

$$C_{i,j} = \min(C_{i-1,j-1} + \delta(x_i, y_j), C_{i,j-1}, C_{i-1,j}) \tag{2.12}$$

where $\delta(x, y) = 0$ if $x = y$ and 1 otherwise.

Hunt and McIlroy (1976) used the dynamic programming approach (coupled with some heuristic optimisations) in writing a line-oriented program that takes two files and produces a *delta*, a summary that describes how one file differs from the other. The program, *diff*, is still a core utility on UNIX systems 30 years after its creation. It has many uses, from simple inspection of changes between files to space-efficient version control and distribution of software update patches.

The LCS/LD approach can be generalised to more sophisticated types of approximate matching — such as genomic alignment — by using more complex scoring schemes (see Section 2.8.1).

While the LCS/LD paradigm is quite a successful one for identifying the differences and similarities between a pair of strings, there are some contexts in which it can be seen to overestimate the difference (or underestimate the similarity) between a pair of strings. In

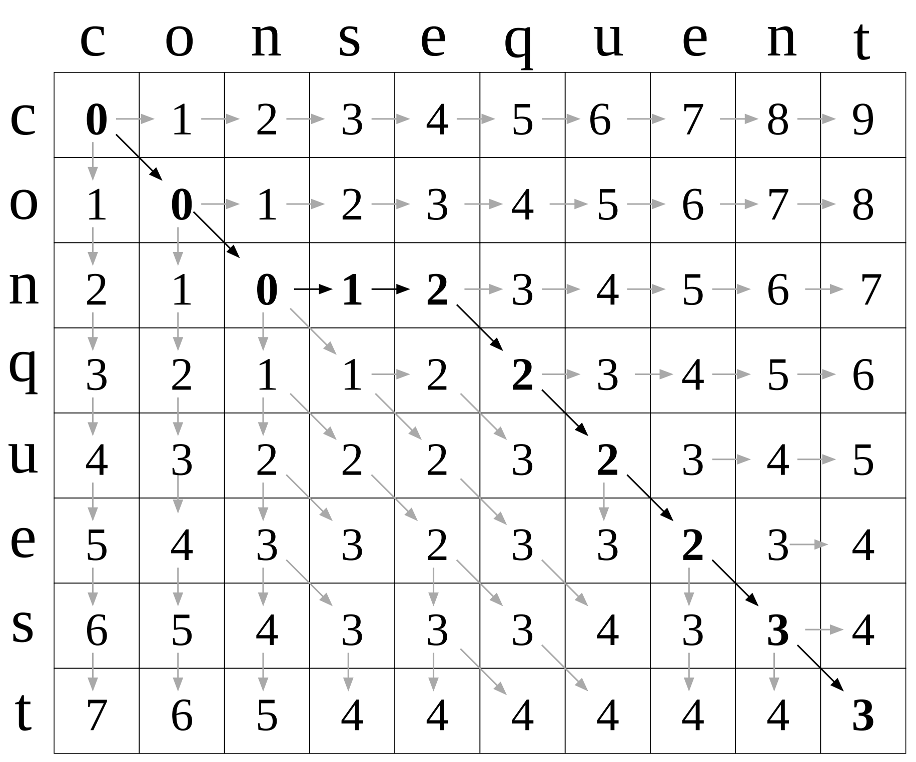


Figure 2.6: A dynamic programming matrix that calculates the edit distance between the strings conquest *and* consequent*. The optimal alignment that leads to the final score of three is in bold.*

particular, there is no provision for *block transposition* between a pair of strings. Figure 2.7 shows the alignment between the two strings `abcdefghij` and `fghijabcde` using LCS. The resulting edit distance of ten seems like an overestimate of the 'true' level of dissimilarity between the strings. In particular, it is an overestimate of the effort required to transform one string into another under a plausible user model.

Heckel (1978) proposes a simple algorithm for a *diff*-like program that takes block transpositions into account. The algorithm is, however, heuristic in nature and relies on the presence of unique tokens in the files being compared; in their absence the algorithm becomes unreliable and failure-prone.

Tichy (1984) describes a more refined algorithm based on suffix trees. A suffix tree is constructed for the source string S, and then each suffix of the target string T is queried against the suffix tree. The node at which the search terminates defines a block transposition between the strings. All remaining tokens that are not part of any block transposition can be considered to be insertions or deletions. Tichy wrote an implementation of the algorithm

```
abcdefghij-----
     |||||
-----fghijabcde
```

Figure 2.7: An alignment between two strings 'abcdefghij' and 'fghijabcde' that differ only by a single block transposition. Their Levenshtein distance is ten.

called *bdiff*. On a repository of source code, *bdiff* produced deltas that were on average 7% smaller than those produced by standard *diff* (Hunt and McIlroy, 1976).

Greedy string tiling (GST) (Wise, 1993; 1996) is another alternative for computing a transposition-aware distance metric. It differs from the approach of Tichy by creating a covering set over one string using substrings of other, and operates in its most efficient guise by using Rabin fingerprinting (Karp and Rabin, 1987; Broder, 1993). Wise claims significance for the fact that, in contrast with the *bdiff* method of Tichy, under GST the same block cannot be mapped twice onto the other string. However, it is unclear why this distinction is important.

Another class of approaches quantifies the distance between a pair of strings based on principles in algorithmic information theory (Li and Vitányi, 1997). Algorithmic information theory is based on Kolmogorov complexity, a measure of the information content of data based on the size of the smallest program able to generate that data on a universal computer such as a Turing machine. While the Kolmogorov complexity of a string is formally uncomputable, it defines the lower bound on the compressibility of a string; thus, the Kolmogorov complexity can be approximated from above by the simple expedient of compressing the string in question using some compression algorithm. The better the algorithm, the closer the approximation to the true Kolmogorov complexity of the string.

The reader may observe that many of the most effective modern compression schemes operate according to the principles of traditional information entropy (Shannon, 1948). Indeed, Kolmogorov complexity has many symmetries and equivalences with traditional Shannon entropy (Grünwald and Vitányi, 2006) but, because of its emphasis on the actual object rather than an assumed underlying model (generative distribution), it arguably has a more straightforward interpretation in the context of evaluating the difference between a pair of strings.

The transformation distance (Varre et al., 1998) is one measure of difference deriving from the principles of Kolmogorov complexity. The basic concept is simple: using a scripting language consisting of a set of operations such as insertions, deletions, transpositions and reversals, the transformation distance between two strings is the size of the smallest script that can transform one string into another. The degree of difference between a pair of strings is effectively defined under this measure by the amount of effort required to describe the difference.

Bennett et al. (1998) provide a more formal measure called *information distance*, which is motivated directly from Kolmogorov complexity theory. This approach was further refined by Li et al. (2003) to yield a measure known simply as *the similarity metric*. Li et al. make the claim that the similarity metric subsumes all other string similarity metrics in the sense that (theoretically at least) any meaningful similarity between a pair of strings will be captured by this metric.

2.8 Sequence homology search

Modern sequencing techniques allows biologists to rapidly map large quantities of nucleotide and protein information. Once biological material has been sequenced, it is usually of interest to determine the biological function of the material. Inferring this function directly from the biological code is well beyond the state of the art in biology. In order to minimise the expenditure of effort, the most common starting point for determining the function of a particular piece of DNA or protein is to find its *homologues*: items of genetic material that share an evolutionary history with the item under investigation. Homologous sequences, though they may not be identical, are in general highly similar and usually perform the same biological function, perhaps in a different species. If a homologue of the sequence under consideration has been previously studied and its function determined, this can save an enormous amount of research effort.

Because pairs of homologous sequences tend to be similar, homology search is a task that can be automated with a reasonable degree of accuracy. Superficially, homology search is much like ranked text search: a query (in this case, the sequence of interest) is submitted to the system, and the system returns a set of sequences from its underlying database, ranked in terms of decreasing likelihood of homology. The user can peruse this result list of sequences in an attempt to learn more about the query sequence. However, several features of biological sequence data and the homology search task differ fundamentally from natural language data and the relevance search task.

The main difference between sequence data and natural-language text is the total absence of structure and delimitation in sequence data; a sequence is one long string, and there is no higher-order structure in analogy with words, sentences, paragraphs, and so on. Thus, the query cannot be straightforwardly broken down into subcomponents, as is typically the case with natural-language search. Thus, standard inverted-index strategies as described earlier are not suitable this domain.[7] The sequence homology search problem is essentially one of approximate string matching; indeed, the most popular and effective strategies for searching sequence databases for homologues are based on approximate string matching techniques. These techniques adapted specifically to the domain of biological sequences using knowledge of the effects of the evolutionary process on such sequences; we discuss such strategies in the following sections.

2.8.1 Global alignment

The *global alignment* score of a pair of sequences quantifies the level of similarity between the sequences as a whole: the higher the score, the more similar a pair of sequences is considered to be. First described by Needleman and Wunsch (1970), this score is calculated in a similar way to the length of the longest common subsequence as discussed in Section 3.1. In the simplest case — generally used for nucleotide alignment — the global alignment is parameterised by three values: the *match score*, the *mismatch penalty*, and the *insertion penalty*. The match score is added to the alignment score for each point in the alignment in which the nucleotides match; the mismatch penalty is subtracted from the alignment score for each point in the alignment at which the nucleotides do not match; the insertion penalty is subtracted from the alignment score for each point in the alignment where there is an insertion (or, equivalently, a deletion). Global alignment of nucleotide sequences as described can be handled using a dynamic programming matrix exactly as described in Section 3.1, using the following update function:

[7]There have been some attempts to conduct homology search using inverted indexes. The CAFE system (Williams and Zobel, 2002) is one such implementation. However, the accuracy of such systems is generally inferior to the approaches described in this section.

$$B(i,j) = \max \begin{cases} B(i-1,j-1) + s(x_i, y_j) \\ B(i-1,j) - e \\ B(i,j-1) - e \end{cases} \tag{2.13}$$

where $s(x,y)$ is equal to the match score if $x = y$ and to the mismatch penalty otherwise, and e is the insertion penalty.

Needleman and Wunsch (1970) point out that, rather than having simple match and mismatch scores, the global alignment score can be generalised by defining a different score for each possible pairing of characters in a particular alignment. This observation is particularly important for generating a sensitive global alignment score for protein sequences. Protein alignment is more complicated than nucleotide alignment because of the chemical similarities that exist between many of the 20 amino acids that constitute the alphabet of a protein sequence (Taylor, 1986). It is not unusual for the evolutionary process to cause an amino acid in a particular location in a sequence to be substituted by another amino acid with similar chemical properties, leading to no alteration in the structure or behaviour of the protein. Thus, while a match between amino acids within an alignment always has the highest significance, the significance of mismatches varies depending on the particular amino acids that are involved. Related to this is the fact that some amino acids have more similar coding at the DNA level, meaning that it is more likely that a point mutation in the DNA would cause the amino acid substitution.

In order to deal with this issue, alignment of protein sequences is not a simple matter of determining a match score and a mismatch penalty. Rather, it is carried out with the use of a *substitution matrix*, which defines the score that should be assigned to any corresponding pair of amino acids in an alignment (Dayhoff et al., 1978; Henikoff and Henikoff, 1992). Various different substitution matrices have been developed, and the choice of which to use is generally guided by the types of homologies the searcher wishes or expects to find; discussion of the way in which the various substitution matrices are constructed is beyond the scope of this work. When a substitution matrix is used, the score $s(x,y)$ in Equation 2.13 is derived from row x and column y in the matrix. The commonly-used BLOSUM62 matrix is presented in Figure 2.8.1. As an example, a match between the residues R and K would yield a score of two; that is, $S(\mathrm{R},\mathrm{K}) = 2$.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	U
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	-1	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-1	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-1	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	-1	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	-1	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-1	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
X	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-4
U	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

Table 2.1: The commonly-used BLOSUM62 scoring matrix (Henikoff and Henikoff, 1992).

```
zzzzzzabcdezzzzz
      |||||
xxxxxxabcdexxxxx
```

Figure 2.8: Two strings with high local similarity but low global similarity. A global alignment technique may not identify the highly-conserved region in the middle of the two strings.

2.8.2 Local alignment

When using global alignment techniques such as described above, sequences that combine overall dissimilarity with local regions of high similarity will not be identified as having a high-scoring region: the penalty scores for the mismatching regions will counteract the effect of the high-similarity regions, leading to a mediocre score overall.

However, sequences of the kind described — global dissimilarity interspersed with local similarity — are common and are often of interest to researchers (Myers, 1991). It is hypothesised that highly-conserved regions are less subject to mutation precisely because of their central importance to biological function, while other parts of the sequence, in particular junk DNA, may exhibit significant genetic drift. Thus, the presence of such highly-conserved regions within otherwise divergent sequences is quite telling and should not in general be overlooked.

Smith and Waterman (1981) describe an algorithm for *local alignment*, in which the score returned is the highest score returned for any possible pair of substrings of the two sequences under consideration. That is, we consider alignments that do not span the entire length of the sequences under consideration. In this way, strongly matching local regions can lead to a high alignment score, free from the penalties incurred when aligning dissimilar regions of the sequences.

The Smith-Waterman algorithm uses a dynamic programming matrix in much the same way as the global alignment techniques of Needleman and Wunsch (1970). The main difference in computing the alignment matrix with local alignment compared to global alignment is that the score for a cell in the matrix is never allowed to be negative. Thus, the update function for cells in the alignment matrix is amended to the below:

$$B(i,j) = \max \begin{cases} B(i-1,j-1) + s(x_i, y_j) \\ B(i-1,j) - e \\ B(i,j-1) - e \\ 0 \end{cases} \tag{2.14}$$

The effect of this change is to allow the alignment to begin at arbitrary points along the sequence pair by discounting negatively scoring regions of the alignment without penalty. The other change is along the top and left boundaries of the alignment matrix, where the values of cells in the hypothetical row $B(0,\cdot)$ and column $B(\cdot,0)$ is set to zero, rather than $-\infty$ as is the case for global alignment.

The other difference is in the interpretation of the alignment matrix once computed. Instead of looking to the bottom right cell $B(|x|,|y|)$ to determine the optimal alignment score as is the case for global alignment, the optimal local alignment score is determined by the maximum value of *any* cell in the matrix. This discounts trailing regions of the alignment that make a net negative contribution to total alignment score.

2.8.3 Heuristic search algorithms

The dynamic programming approaches described in the preceding sections are excellent for finding an optimal alignment between a pair of sequences in a relatively short time. Finding the optimal alignment between a pair of sequences allows for highly accurate and sensitive scoring of similarities between sequences. However, the dynamic programming approach is computationally intensive, and does not scale well to query-based search against the large (and rapidly growing) sequence databases commonly available today. As a rough indication, use of the Smith-Waterman local alignment algorithm for a typical query sequence against the GenBank (Benson et al., 2005) nonredundant protein database (approximately 900 MB at the time of writing) takes approximately one hour on a Pentium 4 class desktop computer; searching the much larger GenBank nonredundant nucleotide database (NR) takes several days.

Apart from the frustrating latency that such long search times impose on researchers, the enormous computational resources required to complete just one such search contributes significantly to the expense of conducting biological research. In order to increase the responsiveness of and reduce the cost of query-based search against these large sequence databases, a number of heuristic approaches for search have been developed. These heuristic techniques are based on the observation that calculating a full optimal alignment score using a technique such as Smith-Waterman is a wasted effort for a majority of sequences in the database for any given query. This is the case for several reasons. First, a typical query will be highly dissimilar to most sequences in the database. No matter how a pair of dissimilar sequences are aligned, the alignment score will be no more than random noise, far below the threshold of statistical significance. Thus, exhaustive calculation of a precise optimal alignment score for such a sequence pair is entirely unnecessary. Second, even if a pair of sequences do contain significant similarities, a large portion of the alignment is infeasible; once again, computing cell entries in such areas of an alignment matrix is likely to be fruitless work.

Heuristic search techniques take advantage of the above observations to attempt to identify sequences that are highly unlikely to be similar to the query sequence (or, equivalently, identify sequences that are highly likely to be similar) using a less computationally expensive technique than dynamic programming. If they are able to do this successfully, a vast majority of the expensive alignment operations for the query can be avoided.

FASTA and BLAST

The FASTA (Pearson and Lipman, 1988) and BLAST (Altschul et al., 1990; 1997) heuristic homology search algorithms are the best-known heuristic search algorithms. They both operate on similar principles, though the latter has largely superseded the former as a result of a number of optimisations that make it significantly faster.

Both FASTA and BLAST use a multi-stage incremental search technique, using crude but efficient techniques to screen out unpromising areas in the alignment space before progressing to more sensitive techniques that are applied to the smaller remaining search space. Though there are significant variations in detail between the FASTA and BLAST implementations, the search stages can be categorised as (in increasing order of sensitivity): hit detection, hit extension (ungapped alignment), and restricted gapped alignment.

In the hit detection phase, short overlapping segments of the query sequence (for example nucleotide subsequences of length three) are inserted into a lookup table.[8] The sequences in the database are then sequentially scanned, with each chunk in the database sequences being compared to the lookup table, and a hit recorded. This stage serves to screen out database sequences that are grossly dissimilar to the query sequence, as well as identifying *diagonals* within promising sequences that may form part of a successful alignment. Thus,

[8]A small in-memory inverted index.

the search space is reduces in a twofold manner, both by reducing the number of sequences under consideration and by reducing the search space within the alignment space of those sequences that remain.

The hit extension stage varies significantly between FASTA and BLAST, but the basic principle remains the same: the hits detected in the previous stage are extended out along the same diagonal in order to create a series of ungapped local alignments, which may represent fragments of a full gapped alignment. Once again, this stage filters out many sequences that fail to demonstrate potential to produce a high alignment score with the query, while restricting the feasible area of the alignment matrix that must be searched for those sequences that do move on to the next stage.

The final stage of the FASTA and BLAST search process is a gapped alignment based on the results of the ungapped alignments. FASTA uses a *banded alignment* process (Chao et al., 1992) to restrict gapped alignment to a region surrounding the best ungapped alignment. BLAST uses seeded gapped alignment with dropoff (Altschul et al., 1997), which is another way of restricting the alignment to promising areas of the dynamic programming matrix. In both cases, the result is a list of approximations to the true optimal gapped alignment score for a variety of candidate sequences from the database. A statistical thresholding process then determines the set of results to return to the user.

Both FASTA and BLAST are several orders of magnitude faster than SSEARCH, returning results in minutes rather than hours or days. The tradeoff is a small but significant drop in overall accuracy. While BLAST in particular produces results in acceptable times even for large databases, heuristic search algorithms for homology search are still much slower than index-based keyword searches of databases of similar size; where one expects an index-based search system to return results in a fraction of a second, BLAST may take several minutes. Improving the speed of homology search is therefore still an important goal.

Iterative profile-based search

While standard alignment-based heuristic search techniques work well in many situations, they sometimes lack the sensitivity to perform effectively for distant-homology searches. It is often the case that distant homologues, while possessing distinct similarities, do not produce a statistically significant score when directly aligned. The quality of distant-homology searches can be improved by using profiles: representations of a class of related sequences. Because they allow characteristics to be inferred over a whole class of sequences, profiles are able to capture a richer set of associations and relationships than can be done by individual sequence representations. For example, highly conserved areas become clearly identified, whereas areas of genetic drift can be discounted.

Position-specific scoring matrices (PSSMs) are a common choice for profile representation (Gribskov et al., 1987). A PSSM for a group of related sequences is built by first performing a multiple alignment of those sequences. For each position in the multiple alignment, a separate scoring vector is constructed based on the elements present in that position. The scoring vector contains one entry for each element in the alphabet and specifies the score that should be received for aligning that element against the current position in the profile. In general, the vector for a given position will score highly for those residues (in the case of proteins) that are present in that position, and score worst for those residues that are unrelated to those present. The mechanics of alignment do not change substantially: instead of aligning a candidate sequence against the query using a standard alignment matrix such as BLOSUM, the score is calculated by using the appropriate scoring vector for each position in the alignment.

PSSMs are notable because the search mechanics are not substantially altered in comparison with standard alignment; as a consequence, search with PSSMs can be integrated with standard heuristic search algorithms such as BLAST with only a minor effect on search speed. Altschul et al. (1997) introduce PSI-BLAST, a modified version of the BLAST heuristic that works with PSSMs.

PSI-BLAST does not require the user to manually annotate sequences in order to build up a profile; rather it builds a profile automatically from a standard query sequence using a technique called *iterative search*. The first step of the iterative search process is to perform a standard BLAST search with the user-supplied query. The highest-scoring results from this search are assumed to be homologues of the query sequence and combined with it to form a PSSM. PSI-BLAST is then invoked to search the database with the newly-generated profile. The high-scoring results are once again combined to form a new profile, and the process is repeated. This continues so long as there are new sequences that produce statistically-significant alignments, or until a user-defined maximum number of iterations is reached. The iterative process allows more distant homologues of the query sequence to be discovered, because the profile captures common characteristics of the group that may not have been present in the original sequence. Note that the iterative process is analogous to the pseudo-relevance feedback strategy in text-based information retrieval, which similarly aims to make the search more sensitive to superficially-distant matches (Rocchio, 1971).

Iterative search can be very sensitive at finding distant homologues of the query sequence. However, there are some dangers with the technique: the first of these is that the inclusion of false-positives — non-homologues — in the profile at early iterations can compound and lead to *query drift* in which the profile starts to represent a group of sequences that are not related to the original query (Schaffer et al., 2001). A related danger, profile corruption, is caused by the presence of high levels of redundancy in search results (Li et al., 2002). A large number of near-identical sequences can heavily bias the profile and cause it to overfit, leading to a profile that is not broadly representative of the class of homologues.

2.9 Summary

In this chapter, we have presented contextual background regarding the history and state of the art in areas that are pertinent to the work described in this thesis. Material covered in this chapter includes the problem of task definition for redundancy management, hashing algorithms, the theory and practice of information retrieval, distributed information retrieval, data structures for indexing, approximate string matching, and sequence homology search.

In the next chapter we present an examination of existing algorithms for the identification of redundancy in text collections, including a detailed taxonomy and discussion of document fingerprinting techniques.

Chapter 3

Detecting co-derivation

In Chapter 2 we discussed the issue of document co-derivation, and why the task of co-derivative document discovery is interesting in the context of intra-collection redundancy management. In this chapter, we describe three classes of algorithm that can be used for detecting co-derivation: approximate string matching, term-vector algorithms, and document fingerprinting. We discuss the strengths and weaknesses of each of these approaches. Having identified document fingerprinting as the approach best suited for discovery of co-derivation relationships in large collections, we proceed with a detailed exposition of this technique, including areas of potential difficulty.

3.1 Approximate string matching

Approximate string matching (see Section 2.7) describes a class of algorithms that quantify the degree of similarity between strings. In many cases, this quantification is an attempt to approximate the amount of editing effort required to transform one string into another. As co-derivation results from an editing process, we expect them to show higher similarity according to such measures than documents that do not share any common heritage. Thus, approximate string matching techniques are a plausible and appealing approach to the problem of detecting co-derivatives.

Approximate string matching techniques have previously been used in this context. Wise (1996) used his greedy string tiling algorithm in the construction of YAP3, a plagiarism detection tool. The so-called similarity metric of Li et al. (2003) has likewise been used as the basis for a plagiarism detection tool, SID (Chen et al., 2002; 2004), and also for building a phylogeny of co-derivation for chain letters (Bennett et al., 2003). Other approximate string matching methods have not been explicitly used for the task of finding co-derivative documents, though some — such as the transformation distance metric of Varre et al. (1998) — have been used for a similar purpose for genomic search. Nonetheless it is apparent that, for example, the volume of output produced by *diff* (Hunt and McIlroy, 1976) could be used as the basis for determining the co-derivation status of a pair of documents.

A limitation of the approximate string matching algorithms discussed in Section 2.7 is that they rely on computationally expensive pairwise comparisons of fulltext strings. That is, a query of one string against a collection of strings usually requires that the full text of the entire collection be read. In situations where a collection may occupy many gigabytes and the number of matches is relatively sparse, this can be an extremely costly process. Under such circumstances massive speed gains can potentially be made by using an index-based approach.

3.2 Term-vector algorithms

Term-vector algorithms predict the co-derivation status of documents by comparing their term vectors. A document's term vector simply records the number of times that each term in the collection vocabulary has occurred in that document. Term-vectors are useful in the context of co-derivative detection as they behave in a predictable way under editing. It is apparent that two identical documents will have identical term vectors; that is, each term will occur in both documents an equal number of times. Insertions, deletions and edits will cause the term vectors to diverge gradually rather than abruptly. Thus, a high level of similarity between the term vectors of two documents is likely to be a robust predictor of co-derivation.[1]

The advantage of using term-vector algorithms for detecting co-derivation relationships between documents is that the process can be easily supported by a standard term-based inverted index such as is commonly used for query-based search. The ability to use an inverted index confers a significant speed advantage over the linear-pass approximate string matching algorithms discussed in Sections 2.7 and 3.1.

Relative-frequency models are closely related to the vector-space model for query-based search, though they do not share the same theoretical foundations. It is thus not surprising that early work in index-supported co-derivative detection used vector-space measures directly. Sanderson (1997) was motivated by a desire to find nearly identical documents in a

[1]We discuss caveats and pitfalls of this assumption later in the section.

collection of newswire documents from Reuters. In order to do this he simply fed the target document as a (very long) query into an existing search system. Sanderson found that this approach produced acceptable results, albeit on a small collection and on an easy task.

Both Shivakumar and García-Molina (1995), and Hoad and Zobel (2003) compare several vector-space scoring functions with novel term-vector measures specifically designed for the purpose of co-derivative document detection. While these new measures abandon the strict theoretical underpinnings of the vector-space model, they are designed to be more sensitive and discriminative at the particular task of co-derivative document detection. In particular, they amend a deficiency of standard vector-space measures for this task, which is that they do not penalise differences in term frequency between a pair of documents. For example, if document A contains the word 'aardvark' seven times and document B contains it just once then a vector-space measure would score this pair more highly than otherwise identical documents in which the term 'aardvark' occurred once in each. This does not match our intuition when it comes to co-derivative detection: the large discrepancy between instances of the word 'aardvark' does not increase our confidence in the common origin of the two documents; indeed, it decreases it.

The *relative-frequency model* (Shivakumar and García-Molina, 1995) uses a two-stage scoring process. In the first stage, corresponding terms between a document pair are screened. They are only retained in the *closeness set* if the difference between the relative frequencies is below a certain tolerance level specified by the user. Once the closeness set has been assembled the terms within it are combined into a score using an asymmetric measure based on the vector-space cosine measure. The higher of the two scores using this asymmetric measure is then assigned to the document pair as its similarity score. In effect, then, the relative-frequency model is very similar to the standard vector-space model except that terms with significantly different relative frequencies are discarded from the calculation.

The family of *identity measures* (Hoad and Zobel, 2003) do not have the preprocessing step of the relative-frequency model. Rather, differences in frequency between corresponding terms are directly incorporated as a penalising factor in the scoring function. The identity measures also penalise a pair of documents for the difference between their two lengths, the intuition being that a pair of co-derived documents should be of similar length. A family of measures are built within this framework by varying the presence and weighting of different factors used by the measure.

Shivakumar and García-Molina (1995), and Hoad and Zobel (2003) compared their term-vector based measures for co-derivative detection to standard vector-space measures (in particular cosine). In both studies the accuracy and sensitivity of the customised term-vector measures were clearly superior to those of cosine and other vector-space measures.

The identity measure was used at the sentence level by Metzler et al. (2005) to try to identify sentences that were part of the same information flow. This measure performed reasonably well, but did not outperform a variety of other measures for this task.

While relative-frequency techniques are reasonably robust and show graceful degradation in the face of co-derivation by revision, it is worth noting that the behaviour of these measures is less impressive when other kinds of co-derivation are considered. As an example, taking a document and concatenating it to itself will result in a new document that does not register a particularly high similarity score using the measures of Hoad and Zobel or Shivakumar and García-Molina. More disturbingly, a digest-style document that contains excerpts from various sources will most probably score poorly against all the documents from which it is derived for all of the scoring functions described above. Thus, one must be careful in evaluating the suitability of term-vector techniques for co-derivative detection. While existing experimental work suggests high accuracy in the domains investigated, this may not be the case in domains where the prevalent form of co-derivation does not preserve the correspondence between term-vectors.

The term-vector approaches described have not been used for the discovery problem in any published work. Further innovation may be possible, but at first glance such approaches are not likely to be suitable for this problem. While it is certainly possible to avoid an explicit pairwise comparison process by traversing the index and updating accumulators for all pairs of documents that appear in each postings list, this is likely to consume an infeasible quantity of resources. In an inverted index for a typical document collection, there will be some terms that have extremely long postings lists associated with them. A quadratic expansion of such lists would result in an enormous expenditure of time and memory; consider that for a term that appears in 100,000 documents, nearly 5,000,000,000 accumulators would need to be managed. While accumulator-limiting strategies may be of some help, term-vector approaches must be considered unsuitable for the discovery problem until demonstrated otherwise.

3.3 Document fingerprinting

The techniques for detecting co-derivatives described in the previous sections operate either directly on the text of document pairs (Section 3.1), or by comparing term distributions using a standard term-based inverted index (Section 3.2). An alternative approach is to attempt to extract indexable features from the collection that are individually and directly indicative of document co-derivation. The presence of such a feature in two documents is evidence of co-derivation between the two documents, irrespective of the distribution of other features. For such an approach to be successful, the features gathered must possess the following properties:

1. Relatively high probability of uniquely identifying a document as having a certain origin (discrimination)
2. Relatively low probability of being altered by minor alterations to the document (robustness)
3. Domain independence
4. Economical extraction
5. Economical storage

Document fingerprinting is the name for a loose class of techniques that aim to identify co-derivative documents using features that conform to the properties described above. In the following sections, we present a taxonomy of document fingerprinting techniques and variants.

One property that is common to all document fingerprinting techniques is that they take a *syntactic*, rather than a semantic, approach to grouping documents. It is possible, and in fact common, for two otherwise unrelated documents to be semantically highly similar. An example would be essays from two students in response to the same assignment questions. By contrast, a high correlation of syntactic features such as identically constructed sentences is overwhelmingly suggestive of a co-derivation relationship; in the case of the student essays, a number of identical paragraphs is strong evidence of plagiarism. The emphasis on syntactic similarity clearly distinguishes document fingerprinting from the wide variety of search and clustering techniques that are intended to find semantic (topical) similarity between documents.

3.4 Fingerprinting: a brief example

In this section we briefly describe a naïve but plausible approach to addressing the search and discovery problems. We then discuss the problems inherent in this naïve methodology.

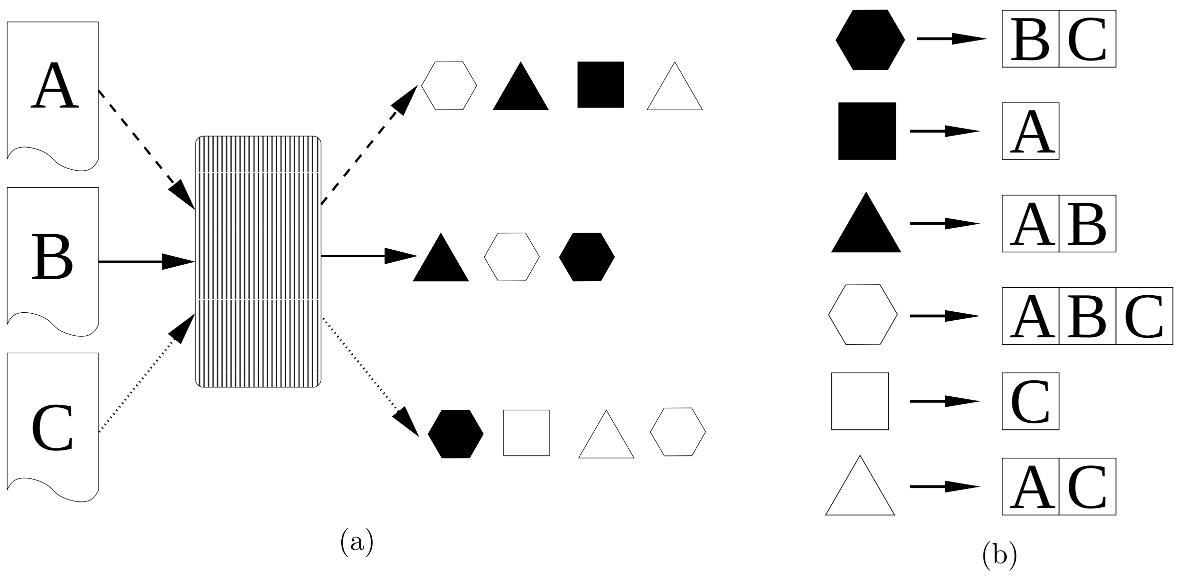


Figure 3.1: A high-level example of the generation of a fingerprint index.
(a) The features generated for three hypothetical documents.
(b) The inverted index constructed from the features extracted in (a).

Let us imagine for the moment that we are able to extract features from documents that are reasonably faithful to the properties described above in Section 3.3. For the purposes of this description, we are not concerned with the precise nature of the features, nor that of the feature extractor. We can view the system as a ‘black box’ that takes a document and returns a set of features. This abstracted view is illustrated for a hypothetical collection of three documents A, B, and C in Figure 3.1(a). Once features have been extracted from all documents in a collection, they can be arranged into an inverted index structure; see Figure 3.1(b). With an inverted index, we are easily able to access a list of documents that contain a particular feature, allowing for easy querying based on feature values.

For the search problem, one possible algorithm is as follows: given a query document, we use our black box to extract all relevant features from this document. We then initialise a counter for each document in the collection being queried.[2] For each feature extracted from the query document, the postings list for the corresponding feature is extracted from the inverted index. For each document in that postings list, the counter is incremented. After all features have been processed, the counter for each document contains a count of the

[2]In *document-at-a-time* processing we do not need to initialise an array of accumulators; however, this implementational variant is beyond the scope of this thesis.

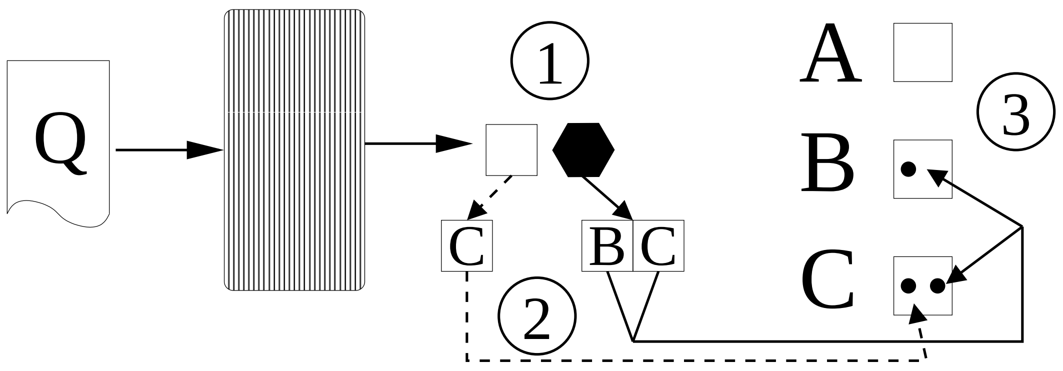


Figure 3.2: Use of a feature index for the search problem. Features are extracted from the query document (1), postings lists for each feature are retrieved from the index (2), and feature co-occurrences are counted for each document in the collection (3).

number of features the document has in common with the query. This process is illustrated in Figure 3.2. A threshold can be set whereby all documents that share more than the specified number of features with the query document can be flagged as co-derived.

In the discovery problem we wish to find all pairs of documents that are co-derived. A straightforward way of solving this problem is to take each document in the collection in turn and use the algorithm described above to solve the search problem for that document, aggregating the results at the end. However, this approach is extremely inefficient, and we now describe a more sophisticated algorithm — first proposed by Broder (1997) — that operates directly upon the inverted index.

Using this approach, each postings list in the index is considered separately. The strict upper triangle of its Cartesian product is taken to yield a set of document-pair tokens. Each token indicates that the specified document pair have one shared feature. Applying this operation on a postings list with a single entry yields an empty set; a postings list with n entries yields $\frac{n(n-1)}{2}$ document-pair tokens. Once all the document-pair tokens have been extracted from the inverted index, they are sorted so that tokens for the same document pairs are adjacent. These tokens are then aggregated to yield a count of common features between each pair of documents in the collection. Figure 3.3 illustrates the process. The count of common features can then be used as a basis for classifying document-pairs as co-derived. However, as the collection size grows, this process can become prohibitively resource-intensive, in particular as a result of the quadratic expansion of postings lists. Techniques to manage this problem are considered in Section 3.9.

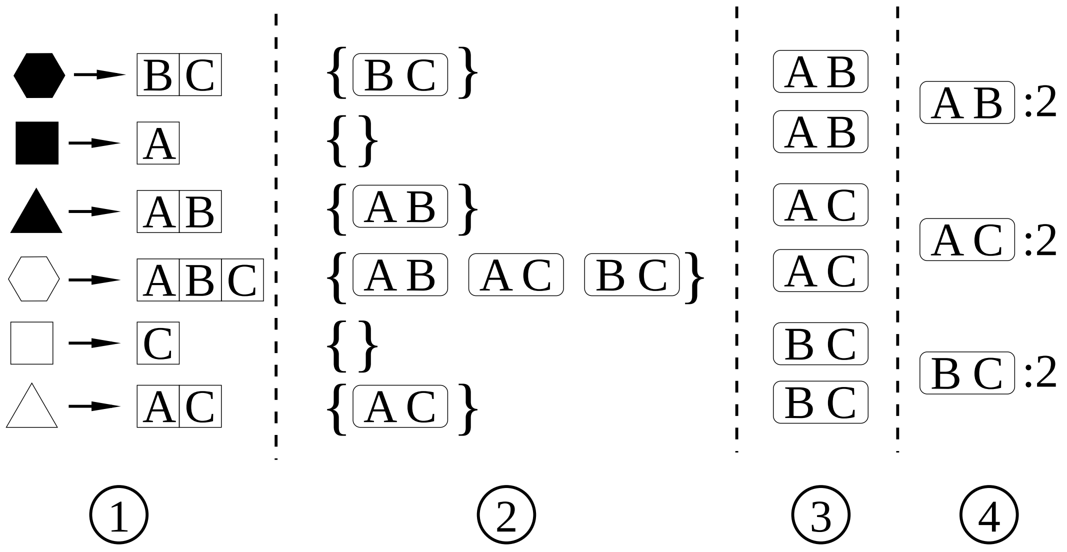


Figure 3.3: Use of a feature index for the discovery problem. For each feature in the index (1), the postings list for that feature is expanded into a set of document-pair tokens (2). Document-pair tokens are sorted (3) and aggregated (4) to produce a feature co-occurrence count for each pair of documents in the collection.

It is apparent from this brief description of the operation of document fingerprinting that at a broad conceptual level it is a simple, almost trivial process. The transition from the basic concept to an efficient, effective state-of-the-art implementation is far from trivial, however. Most particularly, we have yet to address the nature of the features extracted from a document. Even after having established a way of selecting suitable features, we must decide how best to use the evidence available to maximise the accuracy of the technique. Furthermore, the algorithms we describe above can quickly encounter resource limitations; balancing sensitivity and effectiveness with tractability and efficiency therefore becomes extremely important.

In the following sections we discuss in some depth the considerations that shape the creation of an efficient and effective document fingerprinting system, and undertake a detailed survey of existing approaches.

3.5 Feature extraction

There are two underlying approaches to harvesting robust syntactic features for co-derivative document detection. The first of these, *chunking*, is based on contiguous text segments, usually (though not always) of the same length. The other approach, which we call *deterministic term extraction*, uses heuristics to extract a subset of terms from each document such that two near-identical documents are likely to have the same subset of terms selected. We describe these two approaches to feature extraction, their variations, and implementational considerations in the following sections.

3.6 Chunking

It seems intuitive that the longer a given string of text, the less likely it is that exactly the same string appears in two unrelated documents. Probabilistic language models (Ponte and Croft, 1998) seem to support this intuition, with multiplicative factors over a large vocabulary making it improbable that an identical string of any reasonable length will be generated for two separate documents. The likelihood that several such strings will co-occur between a pair of documents by chance is vanishingly small. While it is not necessarily the case that co-derived documents must share common strings, Brin et al. (1995) show that significant editing effort is required before they do not. Given that the presence of such strings is a strong predictor of co-derivation, we assert that they are viable features upon which to base a tool for detecting co-derivative documents.

It was this observation that led Manber (1994) to use contiguous 50-byte strings as the basis of *sif*, a tool for finding similar (but not identical) files within a file system. We refer to such contiguous-text-sequence features as *chunks*, after Brin et al. (1995).

When of an appropriate length, it is easy to see how chunks satisfy conditions 1-4 laid down in Section 3.3. It is unlikely that two documents of completely independent origins would share an identical sequence of — for example — eight words; a minor change to a document leaves a large majority of chunks unaffected; the indexing of chunks is unaffected by domain issues such as language; and they can be easily parsed from most documents. We see how to make these features economical to store later in the section.

3.6.1 Chunking strategies

The dominant technique for breaking a document up into chunks is *shingling*, in which the set of chunks consists of all substrings of the documents that are of a given length with respect to the chosen unit of granularity.[3] For example, a document may be shingled by extracting — as Manber (1994) did — all substrings of the document that are 50 bytes in length, or

[3]The issue of chunking *granularity* is discussed in detail in Section 3.6.2.

all substrings of length four words. Chunks produced by the shingling strategy are often referred to as shingles. The set of shingles in a document is extracted by passing a sliding window of the specified length over the document and recording all chunks; this approach is illustrated in Figure 3.4(a).

Brin et al. (1995) define an alternative approach to chunk extraction known as *hashed breakpointing*. With this approach, chunks are not of a fixed length and they do not overlap. Instead, a chunk is built up incrementally: as a document is tokenised, each token is hashed; if the hash is not equal to 0 *modulo* some parameter m then it is added to the current chunk; if the hash does equal 0 *modulo* m then the token is considered to be a breakpoint and a new chunk is started. The length of chunks is thus not fixed but rather follows a geometric distribution with a mean length of m. Chunks do not overlap because they cannot cross a breakpoint. Chunking using hashed breakpoints is illustrated in Figure 3.4(b).

Brin et al. (1995) list several strengths of the hashed-breakpoint chunking strategy. The most important of these is a reduction in the number of chunks in a document as a consequence of the non-overlapping nature of hashed breakpointing. For fixed-length chunking and a chunk length of 10 words a document of length n words contains $n - 9$ chunks. By contrast, hashed breakpointing with m set to 10 (leading to a mean chunk size of 10) results in only $n/10$ chunks being selected on average. A similar result could be obtained by using fixed-length chunks and not allowing them to overlap, but this would make the chunks extremely unstable: the insertion of a single word in one of a pair of otherwise identical documents could cause the set of chunks extracted from each to be completely different. A similar insertion using hashed breakpoints would only corrupt a single chunk. Thus, the strategy achieves a substantial reduction in the number of chunks that need to be extracted (and thus indexed) but without compromising the fundamental robustness of chunking.

Hashed breakpointing is also robust in an adversarial situation: in order to ensure that a pair of documents share no common chunks, an adversary must rewrite every element of the document. This is superior to fixed-window chunking for which, given a window size of n, it is sufficient to change every nth element. Adversarial scenarios occur in a variety of contexts, plagiarism being the most obvious example; Brin et al. (1995) motivated their work in the context of detecting copyright infringement in digital libraries.

However, this particular strength of hashed breakpointing is also a weakness. The same unpredictability that requires an adversary to rewrite the entire document for security means that hashed breakpointing provides no guarantee of a minimum number of chunks in common between a pair of co-derived documents, unless those documents are exact duplicates. The

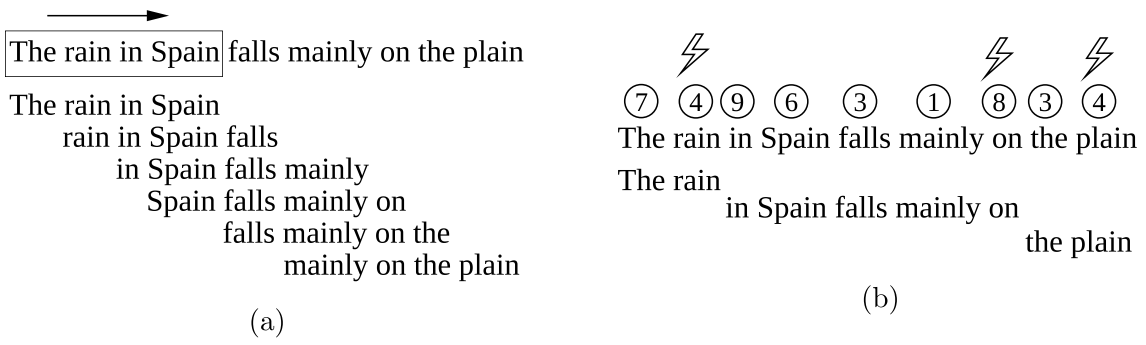


Figure 3.4:
(a) The shingling chunking strategy in which all strings of a fixed length are extracted by passing a sliding window over the text.
(b) The hashed breakpoint chunking strategy, in which chunks are non-overlapping and the boundary between chunks is determined by hashing each word and breaking when the hash evaluates to 0 modulo *some value m, in this case 4.*

variable length of chunks in this scheme also means that chunks cannot be considered to have equal weight when attempting to quantify the level of co-derivation between a pair of documents.

3.6.2 Chunking granularity

When building a feature extractor based on chunking, one must select the level of granularity at which the chunking is to operate. It would at first seem best to use the byte as the atomic unit for document fingerprinting. It is trivial to convert a document into a stream of bytes (primarily because it is a stream of bytes in the first place). In standard natural-language documents, bytes correspond to characters. As characters can be considered the atomic units of natural language, byte-level granularity seems to be a sensible choice.

Despite the simplicity of byte-level granularity, many implementations of document fingerprinting use a coarser granularity. The reason is that the finer the level of granularity, the greater the number of chunks in a document of a given size; this in turn leads to a greater resource burden during processing. This tradeoff has seen experimentation with larger atomic units, in particular words and sentences.

The number of chunks of length m in a document of length n units is $n - m + 1$. Our examination of a typical document (in this case, a conference paper) showed it to be 44,559 bytes long, and consist of 6,478 word occurrences. Thus, with byte-level granularity and a chunk length of 50, the document will yield 44,510 chunks. If word-level granularity were used instead with a chunk length of 8 words (roughly the same length) then there would only be 6,471 chunks, a reduction of over 85%. A further large saving could be gained if sentences were used instead of words.

Of course, as the granularity increases there is a commensurate loss in sensitivity: fewer changes need to be made for two documents to have no chunks in common. Another negative is a loss of domain independence. While all documents stored on a modern computer are composed of bytes, the same cannot be said for higher levels of granularity. Parsing words from a document is not a trivial task, and the results can be strange if a document diverges significantly from the typical characteristics of a natural-language document. Sentences and higher-order syntactic structures are more troublesome again: sentence parsers are easily confused and even reasonably regular natural-language documents often contain passages in which the notion of a sentence is not valid, such as tables or mathematical equations. Brin et al. (1995) report that their sentence-level fingerprinting system was rendered less effective because of difficulties in accurately parsing sentences; this point is reiterated in Shivakumar and García-Molina (1995).

In most cases, word-level granularity is a reasonable compromise between the various considerations discussed above. While words cannot be extracted from any arbitrary files, most human-readable documents can be reliably segmented into reasonable word-like units using a simple tokeniser. Furthermore, while characters are the atomic unit in phonetic-alphabet languages, it can be argued that words are the functional atoms of a natural-language document. It is unlikely that some characters in a word are changed between two co-derived documents without the entire word being changed (an exception to this is the correction of spelling and typographical errors). The distinction between characters and words is far less clear in hieroglyphic-alphabet languages such as Chinese. In these cases the conservative action is probably to use a character-level granularity, but investigation of this problem is outside the scope of this work.

3.6.3 Chunking in the literature

Having discussed the various options available when choosing how to break a document into chunks, we now review some of the choices made in the existing literature.

Several previous efforts have made use of byte-level granularity. Both Manber (1994) and Schleimer et al. (2003) used byte-level granularity with a window-width of 50 bytes.

Heintze (1996) used a variation on byte-level granularity. As his work focused on working in noisy environments such as those in which documents are produced using optical character recognition and Postscript conversion technology, it was found useful to eliminate all vowels as being significantly error-prone; subsequently, chunks were based by passing a 20-byte window over the consonants in the document. Heintze estimates that, given typical character distributions in natural language (or at least English), this corresponds to byte-level chunks of approximately 30-45 bytes.

The most common choice of granularity was word-level. The chunk size varied significantly between different groups. In decreasing order of length, the following chunk sizes were reported in the literature: Broder et al. (1997), 10 words; Hoad and Zobel (2003), eight words; Fetterly et al. (2003), five words; Pereira Jr. and Ziviani (2003), four words; Lyon et al. (2001), three words; Clough et al. (2002), one word. In many of these cases the researchers tried a range of parameters to optimise the effectiveness of their tool. The broad range of different parameterisations shows that the length of the chunk is not a particularly significant factor in the performance of a fingerprinting tool, and is dominated by other effects. It is somewhat surprising, however, that Clough et al. (2002) found single-word chunks to be most effective, as this reduces the fingerprinting scheme to a simple word-overlap measure. Clough et al. speculate that this surprising result is perhaps due to domain effects pertaining to their project, which focused on reuse of newsagency wires by newspapers.

Finkel et al. (2002) used hashed breakpointing with a m parameter of 10, meaning that the expected chunk length was 10 words.

There have been some attempts to use higher levels of granularity. Shivakumar and García-Molina (1998) used line-level granularity and non-overlapping chunks of length two or four lines. Brin et al. (1995) and Campbell et al. (2000) use sentence-level granularity and a chunk length of one sentence. Brin et al. noted that correctly parsing sentences was problematic and occasionally caused failures in the system.

3.6.4 Classification of co-derivatives using chunks

As discussed in Section 3.6, the presence of a common chunk between two documents is good evidence of co-derivation. The presence of more common chunks between the documents constitutes stronger evidence of their co-derivation. The issue is, then, how to use the evidence provided by a chunk index to most effectively classify document pairs as being co-derivative.

In general, the data about chunk co-occurrences must be combined in some way to yield a value that quantifies the degree of text reuse, and thus provides a confidence estimate regarding the co-derivation status of a pair of documents.

The most common measures used for this task are *resemblance* and *containment* (Broder, 1997). Resemblance is a symmetric measure which is maximised when two documents are identical. It is defined as:

$$R(S,T) = \frac{|\hat{S} \cap \hat{T}|}{|\hat{S} \cup \hat{T}|}$$

where $\hat{S}$ is the set of shingles extracted from a document S. Containment is an asymmetric measure which is maximised when the subject document is entirely reproduced within the object document; the following computes the containment of S in T:

$$C(S,T) = \frac{|\hat{S}|}{|\hat{S} \cup \hat{T}|}$$

While resemblance and containment are useful concepts for classifying co-derivation, there are some issues with these functions. Neither resemblance nor containment are true parameterless functions that can be mapped to a unique value because they are dependent on the parameters used for chunking. Thus, the interpretation of a resemblance or containment value is dependent on the chunking parameterisation.

Because resemblance and containment range smoothly between 0 and 1, they are frequently interpreted as 'percentage similarity' or some similar concept. However, a single edit to a document will disrupt a number of shingles due to their overlapping nature. Thus, the reported resemblance values for a pair of documents may be significantly lower than a human would intuitively expect of a 'percentage similarity'. Caveat emptor.

Resemblance and containment use only information about the proportion of chunks in a pair of documents that are shared. They take no consideration of any properties of the chunks that are used. In ad-hoc information retrieval, making use of both document and collection statistics for weighting the value of terms is of critical importance to producing a high-quality ranking. While it can be argued that this is far less important for chunks because their co-occurrence is unlikely to be coincidental, it may still be worthwhile to incorporate collection statistics into weighting the value of a chunk. We consider this possibility in Chapter 4.

When using chunks of variable length (as is the case with hashed breakpointing) it may also be worthwhile to weight the chunks by their length, as it can be claimed that a long shared chunk provides stronger evidence of co-derivation than a short one.

3.6.5 Chunk selection heuristics

In order to produce the most sensitive measure, it would be ideal to index all the chunks in every document. We call this *full fingerprinting*. In practice, however, full fingerprinting generally places an unacceptable resource burden in terms of index construction, disk storage, and search. It is generally accepted that it is necessary to index chunks selectively in order to improve the performance and scalability of document fingerprinting. The way in which a subset of document chunks is selected has a significant bearing on the overall performance of the fingerprinting algorithm. Many heuristics have been conceived and we present a survey here.

Perhaps the simplest selection heuristic is the *random* scheme in which chunks are selected based on the outcome of a Bernoulli trial. The Bernoulli parameter p can be adjusted in order to control the density of chunks selected. While this scheme has the virtue of simplicity, it represents a poor return in terms of accuracy for the resources invested. This is because while we expect that the number of chunks selected is p, the resolution of the scheme is p^2. For example, with $p = 0.1$ we would expect that approximately one in every ten chunks is selected for storage. However, given two identical documents of length 100 chunks, we expect that this selection scheme will only store one chunk that is common to both documents, and there is a nearly 35% chance that the fingerprints of the documents will have no chunks in common. We have thus traded a tenfold improvement in space-efficiency for a one-hundredfold reduction in resolution. This is not an appealing transaction.

Another family of selection heuristics which we may term positional selection heuristics are similarly simple. Hoad and Zobel (2003) describe several such schemes. One variation operates by selecting only every nth chunk in the document. Note that when n is equal to the chunk length, this is functionally equivalent to the non-overlapping chunking strategy described in Brin et al. (1995), though it occurs at a different stage in the fingerprinting process. Other positional selection heuristics include such strategies as choosing the first K chunks in the document and similar variations. One may suspect that this family of strategies is not very effective — they are extremely sensitive to edits and coarse-grained reorganisations of the document text — and indeed Hoad and Zobel (2003) find that this is the case on realistic data collections.

A common feature of the selection heuristics described above that contributes to their ineffectiveness is that the selection process is entirely independent of the content of the chunk itself. This leads to the situation where a given chunk is selected in one document

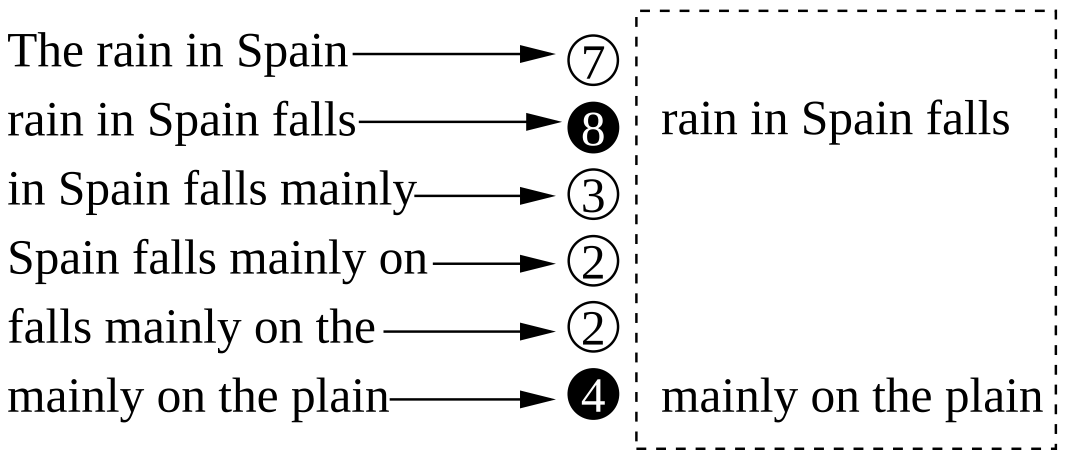


Figure 3.5: The modulo *chunk selection heuristic. Chunks are hashed and all chunks whose hashes correspond to zero* modulo *a parameter p are selected for the fingerprint.*

but not in another and is the fundamental reason why these heuristics have such undesirable performance characteristics. We say that a selection heuristic that chooses either all or none of the instances of a given chunk has the property of *synchronisation.* Synchronisation is a desirable property of any selection heuristic because it avoids precisely the sorts of problems described above. For synchronisation to be achieved the selection heuristic must make its decision deterministically and be based only on the content of the chunk.

The simplest selection heuristic exhibiting synchronisation is the *modulo* heuristic (Manber, 1994; Broder, 1997; Broder et al., 1997). This operates by taking a hash of each chunk and selecting the chunk if the hash is equal to zero *modulo* a parameter p; this is illustrated in Figure 3.5. To contrast the performance of this heuristic with the random heuristic, we set $p = 10$. As above, we expect that approximately one in ten chunks is selected for storage. However, for two identical documents of length 100 chunks, we expect an average of ten common chunks to be selected. More importantly, given a good hash function, we expect that the probability that these two documents will have no common chunks indexed is less than 0.003%.

Broder (1997) and Broder et al. (1997) demonstrated analytically that calculating resemblance and containment based only on chunks selected by the *modulo* heuristic constitutes an unbiased estimate of the true resemblance and containment between two documents. Another selection heuristic proposed by Broder (1997) is called the *min* scheme, in which all chunks in a document are hashed and the n chunks with the lowest hash values are selected; see Figure 3.6. Broder proves that this scheme provides an unbiased estimate of resemblance,

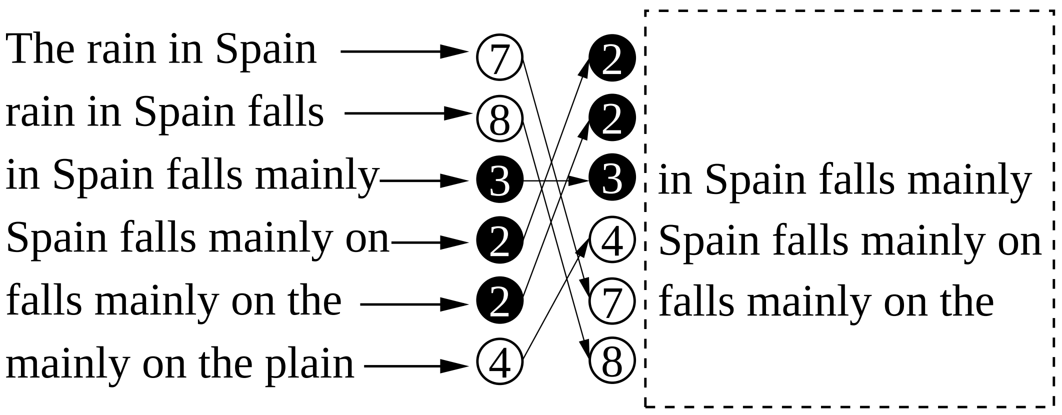


Figure 3.6: The min *chunk selection heuristic. Chunks are hashed and the hashes sorted; those chunks with the lowest hashes are selected for the fingerprint.*

though not containment. It is worth noting that the *min* heuristic does not ensure synchronisation, however. Its main advantage is that it allows the number of chunks selected from each document to be capped.

Manber (1994) — who originally proposed the *modulo* heuristic — also suggests a selection heuristic he calls the *anchor* scheme. The idea with this heuristic is to provide synchronisation by having a lexicon of short strings. A chunk is only selected if it has one of the strings in the lexicon as a prefix. Manber suggests training the lexicon by choosing terms from a collection that are “quite common but not too common”. The *anchor* heuristic possesses much of the character of the *modulo* scheme in the way chunks are selected by examining them for a particular feature, but it has several disadvantages. In particular, the domain-independent character of the *modulo* scheme is lost, as is its well understood statistical behaviour. For these reasons, the *anchor* heuristic is abandoned by Manber in favour of *modulo.* However, a similar selection heuristic is used by Heintze (1996); in this case, chunks that begin with a five-letter sequence that appears infrequently in the document are selected over those with a more common prefix. Note that this selection heuristic does not exhibit synchronisation, though Heintze claims good results in practice.

Finkel et al. (2002) describe two elegant selection heuristics based on the variable-length chunks generated by the hashed breakpoint chunking strategy. In the first of these, the n chunks closest in length to the median are selected, allowing the number of chunks selected from a document to be capped. In the second heuristic, chunks are selected if their length is within a given number of standard deviations from the median. This heuristic is illustrated

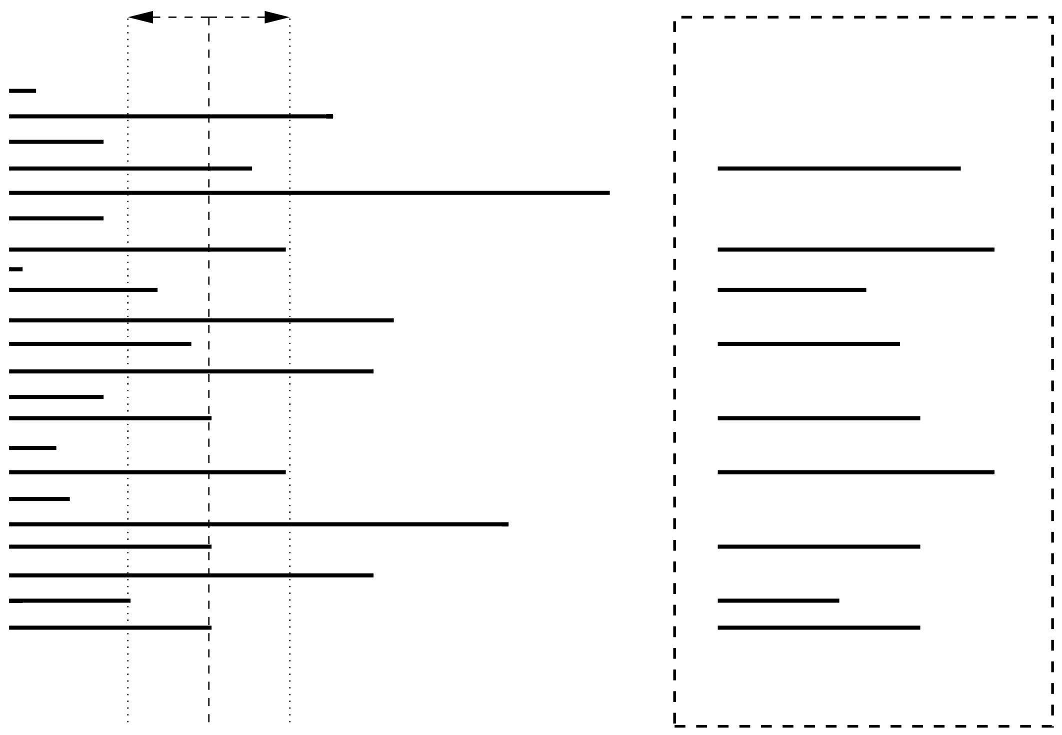


Figure 3.7: The selection scheme of Finkel et al. (2002) for the hashed breakpoint chunking scheme. All chunks of a length within a given distance of the median chunk-length are selected for the fingerprint.

in Figure 3.7. Both these heuristics take advantage of the reduced number of chunks returned by the hashed breakpoint chunking strategy while still forcing chunks to be of a reasonably uniform length, which can be considered advantageous as chunks can be assumed to be of approximately equal weight or value.

Finkel et al. (2002) do not explicitly state whether they use document statistics or analytical results for the median and standard deviation values. If the analytical values are used then the variance-based heuristic does exhibit synchronisation, although the 'closest n to median' scheme does not.

A problem with the synchronised chunk-selection heuristics described above is that while they guarantee the synchronisation property, they provide no guarantee that a chunk will be selected in a run of arbitrary length. This is, in a sense, a direct consequence of the quest to achieve synchronisation. Because the selection of chunks is based solely on the internal properties of the chunk itself without taking into consideration in any way the context in which it occurs, the chunk selection heuristic is effectively stateless. Given a well-behaved hash function, the distance between adjacent selected chunks follows a geometric distribution, which is well known as the classic 'memoryless' probability distribution. It is thus an unfortunate property of such heuristics that, no matter how long the match between a pair of documents, there is a nonzero probability that the match will be overlooked entirely. In the example above this probability is 0.003%, which is probably acceptable in most circumstances. However, the real-world parameterisations reported in some papers lead to more alarming figures.

Using the parameters suggested in Manber (1994) (byte-level granularity; *modulo* parameter $p = 256$), the probability that two documents with a one-kilobyte run of identical text do not share a single selected chunk is 1.83%. The same value obtains for an identical run of 100 words and the parameters suggested by Broder et al. (1997). For runs of text half this length, this probability increases to 13.5%. Such performance is quite likely to be unacceptable (or at least undesirable) in many application domains.

The *winnowing* selection heuristic (Schleimer et al., 2003) is designed to provide a guarantee by capping the longest common contiguous run of text between two documents before at least one common chunk is selected in both documents. It does this by maintaining a fixed length window of the last w chunks. At each increment, the chunk in the window that has the lowest hash value is marked. All chunks that have been marked at least once are then selected for the document's fingerprint; this process is illustrated in Figure 3.8. The winnowing selection heuristic guarantees that any common contiguous run of length at least w chunks has at least one common chunk selected. To see how the guarantee is enacted, consider that a sequence of text of length w will fill the entire window. Thus, because the winnowing algorithm is deterministic and does not take into account context from outside the window, the same chunk will be marked and consequently selected from this sequence of text, no matter where it occurs in a broader context.

Schleimer et al. (2003) define a property of selection algorithms that they call *locality.* A local selection heuristic has a fixed-size window of context and has a deterministic functional mapping between any given window and a particular chunk in that window. They demonstrate that winnowing is a local algorithm and that any local algorithm provides the guarantee described in the previous paragraph. They also prove a number of properties about the class of local selection heuristics and show that winnowing compares favourably with the hypothetical optimal local selection heuristic.

In adhering to the locality property Schleimer et al. have introduced a dependence on

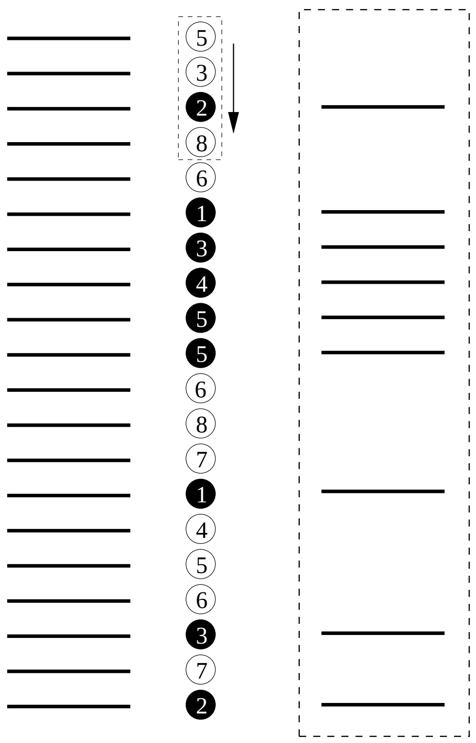


Figure 3.8: The winnowing *chunk selection heuristic. All chunks are hashed; a fixed-length window is then passed over the hash values and all chunks whose hash is minimal for some window in which it appears are marked. All marked chunks are selected for the fingerprint.*

contextual factors when selecting chunks.[4] While this is in a sense the aim of the algorithm, the dependence on context weakens the synchronisation property of the winnowing algorithm. Rather than a given chunk either being selected or not selected for all documents in which it appears, synchronisation is only assured when the local context (the window) is identical. It is perhaps best to view local selection heuristics as a compromise between full synchronisation and full contextual awareness. This highlights a fundamental conflict between contextual chunk selection and synchronisation: pursuing one of these objectives negates the possibility of achieving the other.

Fetterly et al. (2003) describe an unnamed selection heuristic that we call *minimal-chunk sampling*. This takes a fundamentally different approach to the selection of chunks and yields an *ordered* fingerprint that can be seen as a vector of separate single-chunk fingerprints. Minimal-chunk sampling requires the use of a class of hash functions that are able to act as

[4]This is not true in the degenerate case where the window size w is equal to one. Under these circumstances winnowing reverts to a full fingerprinting strategy.

min-wise independent permutations (see Section 2.2).

The algorithm makes use of a set of m hash functions selected at random from a min-wise independent class of functions. For each of the m hash functions, all the chunks in the document are hashed and the chunk with the lowest value retained. This results in a vector of m chunks as the fingerprint for each document.

In each iteration the heuristic is effectively using the *min* scheme with $n = 1$. So why not simply use the *min* heuristic with n set to the desired value? Fetterly et al. (2003) present two main reasons. The first of these is that setting $n = 1$, in combination with the fact that the hash functions are min-wise independent, provides the heuristic with several advantageous formal properties. In particular, the probability that a pair of documents will have the same chunk selected under a particular hash function is precisely equal to the proportion of common chunks the two documents have. The use of separate hash functions to select each chunk means that this probability is independent, rendering the calculation of various tolerances trivial. For example, for two documents sharing 70% of their chunks and with $m = 3$, the probability that they will have no common chunks selected can be computed as $(1 - 0.7)^3 = 0.027$ or 2.7%.

This other advantage claimed by Fetterly et al. is computational. Rather than having to perform a set-overlap operation in order to estimate the resemblance between a pair of documents given their fingerprint, each chunk in a document need be compared only to its corresponding chunk in the other document; that is, the fingerprint is ordered. However, in the context of inverted indexes this is of little benefit, as resemblance is never directly calculated in this way.

3.7 Deterministic term extraction

Deterministic term extraction algorithms are an alternative approach to the extraction of robust features for detection of co-derivative documents. Instead of using features derived from chunks, deterministic term extraction systems extract representative features by heuristically selecting a subset or several subsets of the terms appearing in each document. The intention is to choose these subsets in such a way that two unrelated documents will be unlikely to share them, but that the probability that minor alterations to the document will alter the set of extracted terms is low.

I-Match (Chowdhury et al., 2002) is a co-derivative document detection system that uses deterministic term extraction as its basis of operation. In the I-Match system, terms are selected by a series of rules governed by collection statistics, in particular inverse document frequency (IDF). For example, the 50% of terms in the document with the highest IDF may be chosen, thus screening out all the commonly-occurring terms. The terms extracted are then sorted and hashed to create a single identifying feature for each document. Two documents are considered to be 'near-duplicates' if they have the same hash. Chowdhury et al. empirically determined that the most effective heuristics were those that aggressively pruned out all but the rarest terms in each document. These rare terms can be seen as a kind of keyword set. While more common words are likely to be functional or generic, it is conjectured that it is less likely that a document undergoing minor alteration would have rare — and therefore specific and topical — terms added or removed.

Ilyinski et al. (2002) proposed a very similar system to I-Match. In this approach a predetermined, ordered lexicon of words (determined from collection statistics) is compared to each document. A document's feature vector for a given term in the lexicon is only set to 1 if the frequency of occurrence of that term in the document exceeds a specified threshold; otherwise the vector entry is set to 0. The binary string resulting from this process identifies the document; if two documents share the same vector then they are considered to be duplicates or near-duplicates. Ilyinski et al. used a lexicon of 2,000 words, resulting in document vectors 2,000 bits in length. These could be hashed to create a more compact digest, though the authors did not do this.

Another implementation that has a strong conceptual similarity to I-Match is due to Cooper et al. (2002), though that investigation was less rigorous in terms of both analysis and experimentation. In another similar approach, Conrad et al. (2003) used a hash of the six terms in each document with the highest IDF value as the feature for the detection of 'identical duplicates'. The findings of these experiments were similar to those of Chowdhury et al. (2002).

An advantage of deterministic term extraction over chunking is that the features are global to the document as a whole rather than local to a particular location in the document. For a chunking-based system to be effective, sufficient chunks must be stored to provide good coverage of the entire document so that sufficient evidence exists to make a reliable assessment of co-derivation or near-duplication. This constraint does not hold for deterministic term extraction systems, and Chowdhury et al. (2002) have reported excellent results for their I-Match system when compared to chunking-type strategies.

Having only a single representative feature for each document is attractive from a resource-consumption point-of-view. Not only is the space required by such an approach low, it is guaranteed to grow strictly linearly in the number of documents in the collection. Most importantly, it avoids the computational expense that is incurred when multiple features must be combined to compute a similarity score; see Section 3.9.4. Despite these advantages, the observation holds that a single inopportune difference between a pair of documents can cause the deterministic term extraction system to produce a different feature for each and thus render the documents completely different from the perspective of the system.

In order to make deterministic term extraction more robust, it has been suggested that more than one feature be extracted. Pugh and Henzinger (2003) proposed partitioning the input into r different buckets. This is done by hashing each term, taking its *modulo* with respect to r and assigning it to a bucket based on this value. Each of these buckets is then hashed to form a single feature, thus leading to the extraction of r features.

Kolcz et al. (2004) propose an extended version of I-Match with multiple features. They note that, by using global collection statistics, I-Match is implicitly creating a lexicon of terms to be extracted from each document to form the final feature. The proposed extension, *lexicon randomization*, takes the original lexicon and produces r new lexicons by leaving out terms from each of the lexicons with a probability p. Thus, the new lexicons are essentially random subsets of the original lexicon. Each of these lexicons are then used to induce a separate feature from the document in the same way as described for the original I-Match (Chowdhury et al., 2002).

Haveliwala et al. (2000) use locality-sensitive hashing (LSH), first introduced by Indyk and Motwani (1998), to a similar end. LSH requires the use of a family of hash functions that are min-wise independent (see Section 2.2). LSH submits each document to m different hash functions from a min-wise independent family. Under each permutation the lowest term is retained. All m terms are then concatenated in order to form the *LSH-signature*. If r features are desired, Haveliwala et al. propose repeating the process r times.

It is interesting to note that LSH is conceptually similar to the minimal-chunk sampling scheme proposed by Fetterly et al. (2003) except that instead of selecting chunks based on the output of a battery of min-wise independent hash functions, individual terms are chosen and then concatenated to form a single feature. This close methodological analogy should not disguise the fundamental fact that LSH extracts a global feature or set of global features, while the chunking heuristic of Fetterly et al. generates a set of local features much like any other chunking technique.

Both Pugh and Henzinger (2003) and Kolcz et al. (2004) analytically explore the degree to which the robustness of multiple-feature fingerprints is superior under minor edits when compared to a single-feature fingerprint and show that there is a clear advantage. However, analysis of the increase in false positives due to the multiple features is absent, though Kolcz et al. (2004) report that in empirical studies false positives did not present a problem.

Despite the relatively large number of papers listed above that make use of deterministic term extraction techniques, it is notable that there is very little in the way of a rigorous empirical analysis of their overall effectiveness at identifying near-duplicates or co-derivatives. Chowdhury et al. (2002) analyse the duplicate-detection properties of I-Match, but use machine-generated permutations of seed documents for testing. In their results, I-Match significantly outperforms shingling and super-shingling techniques. On average, over a set of ten query documents, the best parameterisation of I-Match identified 90.0% of the machine-generated co-derivatives, compared to 30.9% for shingling and 16.4% for super-shingling.

It is, however, unclear whether the results observed on machine-generated co-derivatives would transfer to actual co-derivatives in document collections. The way in which the artificial co-derivatives were created — which was by point mutation, rather than block copying — seems particularly disruptive to chunk-based techniques. It is also unclear whether the shingling and super-shingling configuations tested represented good parameter choices for these techniques.

Kolcz et al. (2004) evaluate the performance of their deterministic term extraction technique by considering all document-pairs with a cosine score above 0.9 to be near-duplicates. Recall values were approximately in the range of 0.45 to 0.65. While this evaluation shows the value of multiple-feature fingerprints, it is difficult to determine — given the arbitrary standard for comparison — whether determinstic term extraction techniques as a whole are effective at detecting co-derivatives.

3.8 Efficient feature storage

The features extracted by the fingerprinting algorithms discussed in the previous sections consist of either strings in the case of chunking, or sets of strings in the case of deterministic term extraction. With the typical parameterisations discussed in Section 3.6.3, the representation of a chunk will consume around 50 bytes of memory. With deterministic term extraction, a feature will typically be even larger (though fewer features are usually extracted). Storage of features thus exacts a substantial resource cost.

Except in the case of some specific applications, we are not interested in the actual content of a feature, but simply in the fact that a pair of documents share that feature. The observation can thus be made that the exact feature need not be stored, but only some token that represents the feature, so that documents sharing a feature can be identified. To this end, nearly all fingerprinting techniques in the literature do not store the feature itself, but rather a hash of the feature. In this way, the cost of storing a feature can be reduced to a modest fixed size of perhaps four or eight bytes.

Hashing can of course lead to collisions, in which two differing inputs are mapped to the same hash output. In the context of fingerprinting, this means that a pair documents can be erroneously reported as sharing a feature that they do not in fact share. This increases the risk of false positives, in which two documents that are not co-derived are classified as being co-derived. However, in the generally sparse space occupied by the strings, the rate of error is reasonable. For a hash of size d and a collection from which n unique chunks are obtained, the expected number of collisions is given by:

$$E(n; d) = n - d + d\left(\frac{d}{d-1}\right)^n$$

In the case of a 32-bit hash, we expect that for a set of 100,000 unique chunks, there will be only 1.16 collisions. In the context of a technique that is heuristic and thus already reasonably noisy, the introduction of this small additional level of uncertainty is unlikely to have a noticeable impact. If this is of concern, the use of a larger hash exponentially decreases the probability of collision, such that a 64-bit hash is likely to be virtually collision-free for any collection of feasible size.

The further observation can be made that in the construction of the final inverted index, not only can the features themselves be omitted, but so can their representative hashes. The co-derivative discovery process relies solely on the co-occurrence data encapsulated in the postings lists, without any reference to the feature itself. Thus, once intermediate structures have been merged, all feature identifiers may be discarded, yielding a further resource saving.

3.9 Efficiency considerations for the discovery problem

When combination-of-evidence scoring techniques are used it is necessary to perform an aggregation process such as the one described in Section 3.4. As collection sizes grow, this can become problematic from the perspective of resource consumption. In particular, the expansion of postings lists into document-pair tokens produces a number of tokens quadratic in the length of the postings list. For large collections some postings lists can easily contain thousands of entries, leading to the creation of millions of document-pair tokens from a single list. This is in general an unacceptable burden. Another problem is noise: in large collections millions of document pairs may each contribute a few tokens but still be well below the scoring threshold. This can cause a significant increase in processing load from the large volume of uninteresting document-pair tokens.

3.9.1 Chunk stopping

One simple solution to the problem of quadratic expansion is due to Broder et al. (1997). When a particularly long postings list is encountered it is simply ignored; Broder et al. use 100 entries as the threshold for ignoring a list. This is somewhat analogous to the process of stopping in query-based information retrieval (Salton and McGill, 1986), in which common words are not processed in order to make the search process more efficient. We therefore refer to this technique as stopping in the context of document fingerprinting. In the case of the discovery problem the resource saving as a result of stopping is even more significant because of the quadratic, rather than linear, asymptotic cost in the length of the postings list.

Of course, neglecting to process certain postings lists introduces an element of lossiness to the fingerprinting process. In many collections most of the long postings lists are for low-information features; some examples with chunk-based features are a portion of boilerplate copyright text in a newswire collection, part of the GNU Public License (GPL) in a collection of Unix manual pages, or HTTP header information in a collection of web documents. Such entries are conceptually similar to stopwords: co-occurrence of such features between two documents is a poor discriminator of co-derivation. However, it is important to be aware that aggressive pruning of some features can lead to a loss in fidelity of similarity scoring. For example, two identical manual pages, 20% of which consist of GPL text that has been stopped, will only report as having 80% resemblance rather than the correct figure of 100%.

A more significant problem can occur in loosely curated collections such as crawls of documents from the web. Such collections can often contain large groups of identical or near-identical documents, caused by factors such as mirroring, aliasing, or crawler failure. If the size of a particular group of near-identical documents is larger than the length threshold above which postings lists are stopped then all features extracted from these documents will be stopped. We examine this difficult problem further in Chapter 4.

3.9.2 Coarse counting

The combination-of-evidence scoring process is an example of an *iceberg query*, a type of aggregate query in which only a small proportion of attribute-combinations within a large space exceed the threshold for inclusion (Fang et al., 1998). Iceberg queries are common in data mining applications in which trends or correlations are sought from within large volumes of raw data, for example finding all combinations of products that have been sold together more than a specified number of times in the sales log of a supermarket.

Fang et al. (1998) proposed various techniques and heuristics that would allow iceberg queries to be executed faster or with lower resource consumption. One of these techniques, *coarse counting*, takes advantage of the fact that in an iceberg query, most of the accumulators that are initialised are for object combinations that occur very rarely and thus will not exceed the aggregate threshold. If it can be determined which of these accumulators are likely never to be needed then a significant resource saving can be realised.

The coarse counting technique uses a preprocessing stage in which a fixed-size hashtable of accumulators is used. During the preprocessing stage, each element is hashed and the accumulator corresponding to its hash value is incremented by the weight of that element. There is no attempt to resolve collisions. At the end of the preprocessing phase, a bitmap[5] is used to record which accumulators in the hashtable have exceeded the threshold value and the memory from the hashtable is reclaimed. At this point, the standard aggregation process takes place. However, if the hash of an element corresponds to an accumulator with a value below the threshold value, no accumulator is initialised for that combination of objects. If the initial hashtable of accumulators is sufficiently large, most of the accumulators that will not realise a sufficient aggregate value to exceed the threshold will be ignored, resulting in a significant memory saving.

Note that coarse counting does not lead to false negatives, so it is not a lossy technique. Note also that false positives do not alter the final outcome of the co-derivative discovery process. An increasing rate of false positives leads to an increase in the amount of memory used. The worst case is simply that the time used for the coarse counting stage is wasted; it is never the case that coarse counting degrades result accuracy or leads to greater memory consumption (apart from the bitmap) than if it had not been used.

Coarse counting was used for the discovery problem by Shivakumar and García-Molina

[5]A bitmap is simply a vector of bits, each of which represent a Boolean value for a particular index. They consume far less memory than a standard array as only one bit is required for each entry.

(1998). Each document-pair token extracted from the inverted index was hashed and used to update the corresponding accumulator as described above. After the preprocessing run was finished, another pass through the index was conducted and accumulators only initialised for document-pair tokens if the corresponding entry in the bitmap indicated that the pair may exceed the threshold. Shivakumar and García-Molina did not quantify the resource saving yielded by the coarse counting approach.

3.9.3 Assumption of transitivity

One of the factors that contributes to the difficulty of the discovery problem is the fact that the co-derivation relationship is not transitive. Thus, a fingerprinting system must separately track the status of every document-pair separately. The problem would be much easier if the relationship were transitive because the discovery problem would be reduced to a clustering of documents into transitively-closed subgraphs of the relationship graph. The grouping of documents into 'co-derivative clusters' also holds a great deal of intuitive appeal, as each document in a collection is only a member of a single cluster.

Broder et al. (1997) note the lack of transitivity of the resemblance relation, but then proceed to use a union-find structure to aggregate clusters as if the relation were transitive. In this case, fidelity has been sacrificed for the sake of a faster and more efficient algorithm. Fetterly et al. (2003) similarly neglect the lack of transitivity of the resemblance property. Cho et al. (2000) also choose to consider 'document similarity' to be a transitive relation, though they explicitly discuss the issue and choose to take this path "mainly because it significantly reduces the cost of computing similarity".

Several of the deterministic term extraction algorithms described in Section 3.7 only extract a single representative feature for each algorithm. Extracting only a single feature from each document implicitly makes the assumption of transitivity. If document A has the same feature as both documents B and C then, by definition, B and C also have the same feature and thus are considered near-duplicates. This property amounts to proof that such single-feature systems lack the sensitivity to capture the full complexity of co-derivation relationships. This is true no matter the richness or sophistication of such a feature.

3.9.4 Non-aggregative scoring

An alternative approach to addressing the challenges engendered by combination-of-evidence scoring is to avoid the problem entirely by not combining evidence. This can be achieved by either recording only a single feature for each document, or by setting a threshold of a single common feature for a pair of documents to be considered co-derived.

When a single feature represents each document, there is no need to perform quadratic expansions of postings lists or to maintain a large number of accumulators. All that is needed is for the list of features to be sorted and agglomerated into clusters of near-duplicate documents. When a document produces multiple features, the list of document-pair tokens needs to be recorded, but no corresponding score needs to be kept. This saving is significant, but not overwhelming. Typically such approaches are combined with the assumption of transitivity (see Section 3.9.3) so an efficient structure such as a union-find can be used.

As discussed in the Section 3.9.3, several of the deterministic term extraction algorithms produce only a single representative feature. The scoring for such approaches is inevitably non-aggregative. The remainder of the deterministic term extraction approached surveyed in Section 3.7 have a single feature-scoring threshold. Thus, these techniques are typically non-aggregative in practice.

Chunk-based approaches have typically used aggregative scoring as a way to estimate the resemblance between two documents. This is necessary because chunks are local features and do not on their own provide sufficient evidence of co-derivation. Nonetheless, several chunk-based approaches for non-aggregative scoring have been proposed. These typically work by pre-aggregating evidence into a single feature.

One such approach, *super-shingling* was proposed by Broder et al. (1997). With super-shingling, chunks are selected using a heuristic as described in Section 3.6.5. However, instead of indexing the chunks directly, the set of hashes of the selected chunks for each document is sorted to form a sort of 'meta-document'. This sequence of hashes is then itself chunked to return a set of super-shingles, which together constitute the document fingerprint. Broder et al. argue that any pair of documents that possess a common super-shingle must share a *sequence* of common shingles, suggesting with a high probability that the two documents are co-derived.

A significant flaw of super-shingling is that the ordering imposed on the 'meta-document' by sorting the chunk hashes is largely arbitrary. Two documents will only share a super-shingle if they share a sequence of common shingles in the sorted meta-document, but this sequentiality has no significance. If the common chunks for a pair of documents are interspersed with chunks that only occur in one or other of the documents then the two will not have any common super-shingles. The rate of false negatives was not analysed by Broder et al. (1997).

By using the minimal-chunk sampling heuristic to obtain an ordered fingerprint, Fetterly et al. (2003) are able to index only non-overlapping super-shingles. In the paper they extract 6 non-overlapping super-shingles of length 14 from a fingerprint of 84 chunks. This economy is not possible with super-shingling as described in Broder et al. (1998) because a small number of insertions into the fingerprint will cause two otherwise highly similar documents to have no common super-shingles. By contrast, the ordered nature of the fingerprint generated by minimal-chunk sampling means that small differences between two documents will only corrupt some of the super-shingles, without affecting the alignment between the remaining chunks in the fingerprint.

Super-shingling does not give much control as to the threshold at which documents should be considered co-derivatives. Figure 3.9 shows the probability that a pair of documents with a given resemblance will have at least one common super-shingle for the parameters described in Fetterly et al. (2003). The shaded areas below and above the line correspond to areas of false positives and false negatives respectively, given a resemblance threshold of 0.85. Ideally, such a graph would be a perfect step function with the transition at the desired threshold in order to eliminate both false positives and false negatives. The gentleness of the curve in Figure 3.9 is a direct result of the significant lossiness of the super-shingling technique.

Increasing the threshold of super-shingles to a higher value than one has the effect of making the transition in the graph steeper while at the same time moving the transition point to a higher value, thus reducing the level of classification error. This is illustrated in Figure 3.10. This makes it quite tempting to increase the common super-shingle threshold in order to get more trustworthy results. The only problem is, of course, that super-shingling was conceived of in the first place to avoid the combination-of-evidence phase of the discovery process. Increasing the threshold to any value greater than one negates this advantage.

The solution to this dilemma proposed by Fetterly et al. was to add a further stage to the chunk aggregation process, creating features known as *megashingles*. With megashingling, the set of super-shingles is exhaustively combined to yield every possible combination of a certain number of super-shingles. The example given in Fetterly et al. (2003) is of six super-shingles being combined into sets of two, yielding ${}^6\mathrm{C}_2 = 15$ super-shingle pairs, each of which is then hashed to form a single megashingle. It is apparent that the probability of a single megashingle matching between a pair of documents is the same as that of a match between any two of the original super-shingles.

Megashingling pre-aggregates evidence to negate the need for expensive combination-of-evidence techniques when solving the discovery problem. The tradeoff is clear: an increase

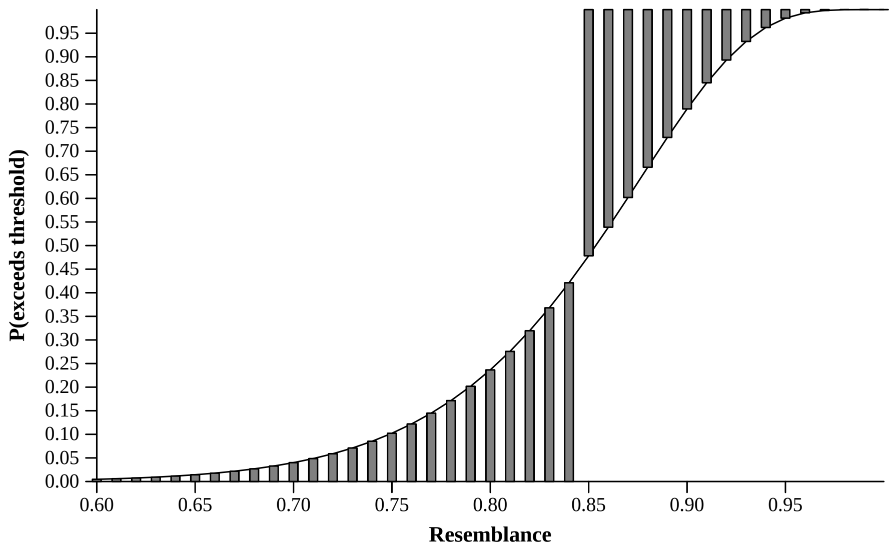


Figure 3.9: The probability that a pair of documents with a given resemblance will have at least one common super-shingle for the parameters used in Fetterly et al. (2003). The shaded area below the curve corresponds to false positives, while the shaded area below the graph corresponds to false negatives.

in index size (in the case above, storing 15 megashingles instead of 6 super-shingles) in return for a potentially significant increase in processing speed and a reduction in memory requirements. Note that megashingling is only feasible for fairly small combinatorial spaces. As an example, increasing the number of super-shingles to 15 and the evidence threshold to 4 super-shingles would lead to ${}^{15}\mathrm{C}_4 = 1,365$ megashingles for each document. Clearly, this is a far less appealing proposition.

3.10 Summary

In this chapter, we have surveyed the three techniques commonly used for the detection of co-derivation between documents.

Approximate string matching techniques for computation of metrics such as the Levenshtein distance are one possible approach for predicting whether a pair of documents are co-derived, with the intuition being that a shorter distance corresponds to two documents containing less divergent text. While some such techniques are certainly capable of reasonable effectiveness at this task, their computational cost tends to be prohibitive for use on larger collections.

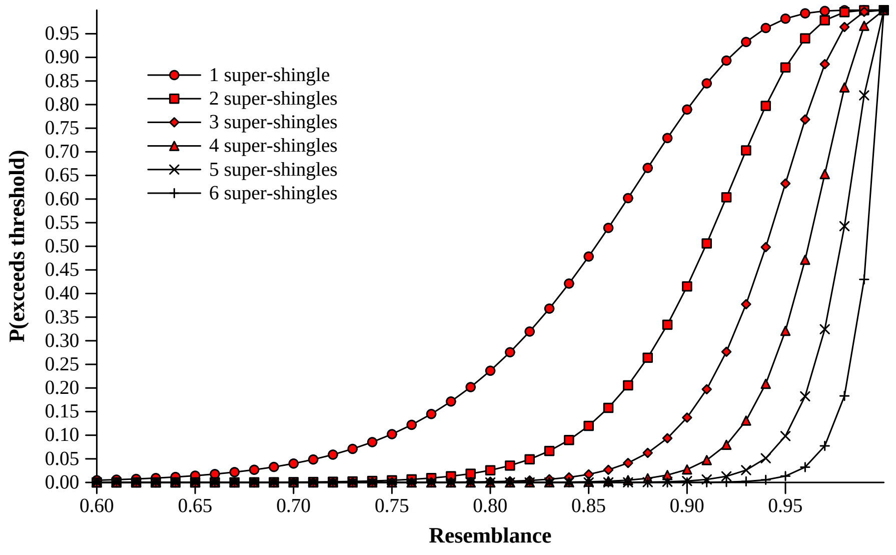


Figure 3.10: The probability that a pair of documents with a given resemblance will have at least a certain number of common super-shingles for the parameters used in Fetterly et al. (2003).

Term-vector based techniques measure a document-pair's similarity based on the level of correspondence between individual terms in the documents. While the term-vector approach draws inspiration from the vector-space model of information retrieval, scoring formulae have been modified to provide greater accuracy for co-derivative detection. The term-vector paradigm provides good efficiency for the search problem, but not for the discovery problem.

Document fingerprinting is the most practical approach for the discovery problem. It is based on the extraction of features from documents that are believed to be likely to co-occur between co-derived documents but unlikely to co-occur between two unrelated documents. The co-occurrence of such features between documents can then be used to accurately predict whether the documents are co-derived.

There are two main approaches to extracting features for document fingerprinting. The first of these, *chunking*, operates by breaking documents up into sequences of contiguous text called chunks. If sufficient chunks co-occur between two documents, there is strong evidence of co-derivation. The second technique, *deterministic term extraction*, works by choosing only certain terms in a document and then combining these into a single feature. Two co-derived documents are likely to produce the same feature if the terms are carefully selected, whereas the probability of two unrelated documents producing the same feature are quite low.

The choice of fingerprinting technique is a matter of compromise, and of making appropriate choices based on the resources available and the proposed application. Algorithms based on deterministic term extraction tend to require fewer resources but provide little in terms of either practical or theoretical assurances, though limited experimentation suggests they can be effective. Techniques based on chunking operate in a more predictable manner, but tend to require significant resources. Thus, chunking techniques tend to use a selection heuristic to reduce the size of the index. This introduces an adjustable level of lossiness to the process, and requires a careful tradeoff between accuracy and efficiency. Many commonly-used selection heuristics provide no worst-case bounds of any sort.

The index processing required for the discovery problem again requires the making of compromises. Non-aggregative discovery techniques are crude but efficient, whereas aggregative scoring techniques provide subtle levels of differentiation but can encounter serious computational bottlenecks. In order to overcome these bottlenecks, various measures such as coarse counting, chunk stopping and assumption of transitivity can be used. The latter two of these once again reduce the fidelity of the algorithm. At present there is no entirely satisfactory solution to this problem.

In the next chapter, we present our original contributions to document fingerprinting.

Chapter 4

Efficient and effective document fingerprinting

This chapter contains material that first appeared in Bernstein and Zobel (2004) and Bernstein and Zobel (2006).

Chunk-based fingerprinting strategies are an appealing choice for discovering co-derivative documents in a large collection. Furthermore, chunk-based fingerprinting techniques that use shingles behave in a predictable way with reference to intuitive measures such as resemblance and containment, and maintain a degree of robustness in the face of attacks. These same claims cannot be made for deterministic term-extraction algorithms, that — while empirically promising — can offer no guarantees in terms of performance.

If all chunks in a collection are processed — a strategy known as full fingerprinting — then resemblance and containment values for document-pairs can be computed directly.

However, full fingerprinting is a resource-intensive strategy, and is rarely practical for large volumes of data. It therefore becomes necessary to use chunk-selection heuristics in order to keep storage requirements at a reasonable level (see Section 3.6.5). Unfortunately, current chunk selection heuristics introduce *lossiness* to the fingerprinting process. Some selection heuristics, notably *modulo*, allow for unbiased estimates of resemblance and containment; however, the variance of these estimates can be high. There is a significant possibility that two documents sharing a large portion of text are passed over entirely, as discussed in Section 3.6.5.

In this chapter we introduce SPEX, an efficient *lossless* chunk-selection algorithm. SPEX allows a system to precisely compute the resemblance and containment of all documents pairs in a collection, thus providing functional equivalence to full fingerprinting, but at a fraction of the resource cost for most collections. The chapter is organised as follows: in Section 4.2, we discuss the operation of the SPEX algorithm; in Section 4.3, we describe our DECO software for document fingerprinting, which implements SPEX; in Section 4.4 we describe our experimental methodology; and in the followin sections we present an empirical analysis of the performance of the algorithm in practice. Our experimental results demonstrate that the SPEX algorithm allows for sensitive detection of co-derivative documents, while in typical cases consuming a modest quantity of resources. Experiments with the highly redundant GOV2 collection suggests that SPEX works best on collections with only a modest degree of duplication.

4.1 Lossless chunk selection

How is it even possible for chunks to be selectively retained in a document's fingerprint, and yet for this process to be lossless in terms of the ability to detect co-derivatives? We make the observation that *singleton* chunks — those that occur only once in the entire collection — make no contribution to the detection of co-derivatives. Any technique for detecting co-derivatives based on chunks operates by calculating some function of chunk co-occurrence. By definition, a singleton chunk cannot co-occur, and thus makes no contribution to any of the metrics for assessing the likelihood of co-derivation between a pair of documents.

In order to get a sense of the quantity of resources that are wasted on processing singleton chunks, we analysed the *LATimes* newswire collection (see Section 4.4). Out of a total of 67.8 million chunks of length 8, only 2.8 million were in fact instances of duplicate chunks, which is less than 4.5% of the overall collection. The number of distinct duplicated chunks is 0.9 million, or less than 1.5% of the collection total. As can be inferred from these figures, the inability of current chunk-selection heuristics to identify singleton chunks leads to an enormous amount of waste on many typical document collections.

A selection heuristic that retains only non-singleton chunks would provide functional equivalence to full fingerprinting, but at a fraction of the resource cost for most collections. The challenge is to find a way of efficiently and scalably discriminating between duplicate and singleton chunks.

4.1.1 Algorithms for lossless chunk selection

The simplest way to eliminate singleton chunks from a chunk index is as a postprocessing step: first perform full fingerprinting, then scan through the resulting index to eliminate all entries that contain just one posting. The problem with this technique is that although the resulting index is minimal, the peak resource requirement is as high as it is for full fingerprinting. Clearly, we need a method that is able to identify duplicate chunks at an earlier stage in the fingerprinting process.

Hierarchical dictionary-based compression techniques such as SEQUITUR (Nevill-Manning and Witten, 1997) and RE-PAIR (Larsson and Moffat, 2000) are primarily designed to eliminate redundancy by replacing strings that occur more than once in the data with a reference to an entry in a ruleset. Thus, passages of text that occur multiple times in the collection are identified as part of the compression process. This has been used as the basis for phrase-based collection browsing tools such as PHIND (Nevill-Manning et al., 1997) and RE-STORE (Moffat and Wan, 2001). However, the use of these techniques in most situations is ruled out by their high memory requirements. For example, the PHIND technique needs about twice the memory of the total size of the collection being browsed (Nevill-Manning et al., 1997). To keep memory use at reasonable levels, the input data is generally segmented and compressed block by block. However, this negates the ability of the algorithm to identify globally duplicated passages. Thus, such algorithms are not useful for large collections.

Suffix structures can also be used for duplicate-chunk identification, and are used for similar tasks in computational biology (Gusfield, 1997). However, as discussed in Section 2.6.1, suffix structures are resource-intensive and thus infeasible for larger collections. We therefore seek a more efficient and scalable technique for this task.

4.2 The SPEX algorithm for lossless chunk selection

Our contribution in this work is the SPEX algorithm, a resource-efficient technique for lossless chunk selection. The SPEX algorithm is a novel hash-based method for the identification of singleton chunks in a collection and has far more modest and flexible memory requirements than the algorithms discussed in Section 4.1. It is thus the first selection algorithm that is able to provide *lossless* chunk selection within large collections. In the case of large collections, the memory needs of SPEX are usually many times smaller than the size of the collection. SPEX identifies repetition by use of repeated passes over the data, and thus is slower than some of the alternatives, but is, as we show, more accurate for a given space overhead.

The fundamental observation behind the operation of SPEX is that, if any subchunks of a chunk are singletons, then the chunk as a whole must too be singleton. Thus, we only need to demonstrate the uniqueness of one subchunk in order to discount the possibility that a chunk as a whole is non-unique. For example, if the chunk 'quick brown' occurs only once in the collection, there is no possibility that the chunk 'quick brown fox' is repeated. SPEX uses an iterated hashing approach to discard unique chunks and leave only those that are likely to be duplicates. This iterative approach shares a similarity of approach with data mining algorithms such as Apriori (Agrawal and Srikant, 1994) that find various data relationships by iteratively refining from more general relationships.

Our iterated approach begins with subchunks of length one — that is, individual terms. The collection is linearly parsed and each word is added to a large hash-based accumulator. The hash-based accumulator (or hashcounter) is simply a large array of counters. When an item is added to the hashcounter, the location in the table corresponding to the hash of the item is incremented. Two particulars are noteworthy: first, collisions are not resolved, guaranteeing constant-time performance at the expense of the possibility of false positives; second, the counters need take on only three distinct values (zero, one, and 'greater than one'), This means that a large hashcounter can fit into a relatively modest quantity of memory. Hashcounters are closely related to Bloom filters, which are data structures for determining the set membership of an object (Bloom, 1970). In both cases hashtables are used in such a way as to allow the possibility of false positives, but not of false negatives.

The number of words in a collection of documents is in most cases relatively modest relative to the overall size of the collection. The highly skew distribution of word occurrence is a well-studied phenomenon (Witten and Bell, 1990). This, combined with the large size of the hashcounter, means that the number of collisions at this stage will be minimal. After the hashcounter has been fully populated, querying it on a particular word in the collection will return either one or 'greater than one'. If the returned count is one, then that word is unique in the collection. If it is greater than one, then the word is assumed to occur multiple times, though false positives are possible due to hashing collisions.

Once the initial hashcounter for chunks of length one has been fully populated from the collection, a new hashcounter is initialised. A second pass of the collection begins, this time sequentially extracting all two-word chunks. Rather than simply inserting each chunk into the new hashcounter, it is first broken down into two subchunks of size one. Each of these subchunks is used as a query against the first hashcounter. If either of the words is marked as unique in the collection, then the chunk is not inserted into the second hashcounter: as

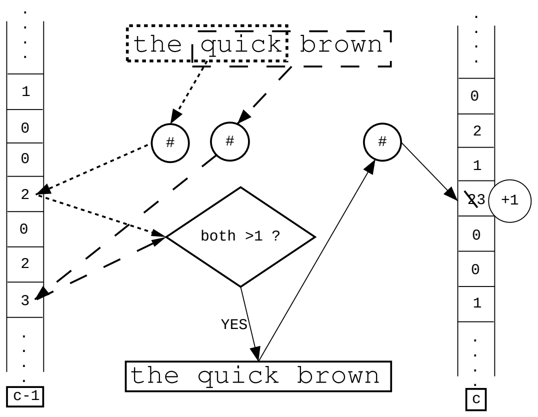


Figure 4.1: The process of inserting a new chunk into the hashcounter in SPEX*. The chunk “the quick brown” is divided into two subchunks: “the quick” and “quick brown” They are each hashed into the hashcounter for length $c - 1$; in this case, two. If the count for both subchunks is greater than one, the full chunk is hashed and the counter at that location in the new hashcounter is incremented.*

one of its subcomponents is a singleton, it too must be a singleton.

After the second pass, the process is repeated for chunks of length three. Each of these chunks is broken into two subchunks of length two, which are queried against the hashcounter for chunks of this length. Note that the memory for the hashcounter for subchunks of length one can now be reclaimed, as the information it contains is no longer needed. At any stage there need be no more than two hashcounters in memory: that for the current subchunk length c and that for the previous subchunk length $c - 1$. Figure 4.1 provides an illustration of one iteration of the SPEX process.

The SPEX process is iterated until we reach the desired chunk length; we use a length of eight terms. When the desired length is reached, an inverted index of non-singleton chunks can be created by only indexing chunks whose subchunks all hash to fields that indicate multiple hits. The entire SPEX process is described more formally in Algorithm 1.

In each iteration of the SPEX algorithm, the chunks become longer, and thus the cost of hashing each chunk grows. However, a larger factor in determining execution time is the number of chunks that must be hashed. We have empirically verified that the execution time of each iteration closely mirrors the number of hashcounter insertions made.

Figure 4.2 illustrates how SPEX works. The figure tracks the occupancy of the hashcounter after each pass of the algorithm for the *all-newswire* (see Section 4.4) collection. The solid line

Algorithm 1 The SPEX algorithm

```
for chunkLength ⟵ 1 to finalLength
   foreach sequence in the collection
      foreach chunk of length chunkLength in sequence
         if chunkLength = 1 then
            increment lookup[chunk]
         else
            subchunk1 ⟵ chunk prefix of length chunkLength - 1
            subchunk2 ⟵ chunk suffix of length chunkLength - 1
            if lookup[subchunk1] = $2^+$ and lookup[subchunk2] = $2^+$ then
               increment lookup[chunk]
```

shows the number of fields in the hashcounter that recorded just a single hit (corresponding to unique chunks), whereas the dashed line shows the number of fields in the hashcounter that recorded multiple hits (non-unique chunks or hash collisions). As discussed earlier, the number of single-word chunks (both unique and duplicated) is low. This means that SPEX does not yet have much discriminative power. Only chunks that contain a singleton subchunk are discarded by the algorithm and there are not many singleton subchunks at this stage. However, as the number of singleton chunks grows in subsequent passes, more can be discounted. The compounding effect of discounting chunks at each pass means that eventually the overall number of occupied hashcounter entries begins to drop dramatically, keeping the hashcounter from becoming flooded.

Figure 4.3 shows what happens if the SPEX process of stepwise refinement is not used and all chunks are inserted into the hashcounter at each stage. At the first couple of passes, the lines track quite closely. However, at chunk size three, they begin to diverge dramatically. Where SPEX is not used, the unique chunks are not weeded out. They therefore begin to occupy more and more space in the hashcounter, thus increasing the possibility of collisions. At a chunk size of seven, the hashcounter where SPEX was not used shows a significantly higher number of fields that have recorded more than one hit. As the number of actual non-singleton chunks remains the same, these extra fields are false positives created by hash collisions. This directly leads to the creation of a larger index.

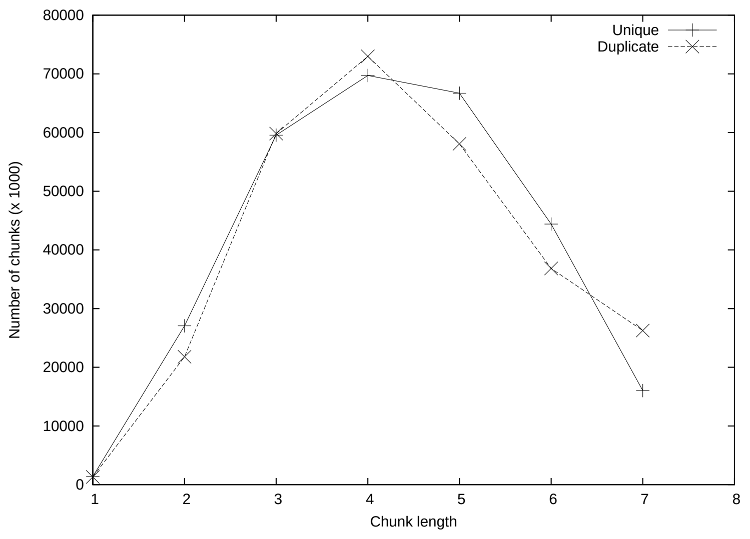


Figure 4.2: Number of hashcounter entries containing single and multiple hits at each iteration for the all-newswire *collection* (2^{26} *elements total).*

4.3 The DECO package

Our DECO system for co-derivative detection is a high-performance software package that combines the SPEX fingerprinting algorithm with advanced indexing techniques, a modular design, sophisticated optimisations, and other previous innovations in the field such as chunk stopping and coarse counting to improve the performance of relationship-graph generation (see Section 3.9).

DECO operates in two phases: index construction and relationship graph generation.

4.3.1 Index Construction

DECO uses a multi-stage 'selection pipeline' during index construction in order to minimise index size and to allow the user to adjust the tradeoffs between build time, size, and reliability. The first stage is to cluster documents that are *exact* duplicates of each other. These are identified by taking one hash of each entire document and then sorting these hashes. Documents with the same hash are considered to be identical, and only one of these documents is indexed. All documents that are co-derived with the 'representative' document are also flagged as co-derived with its duplicates. This exact technique was used by Broder et al.

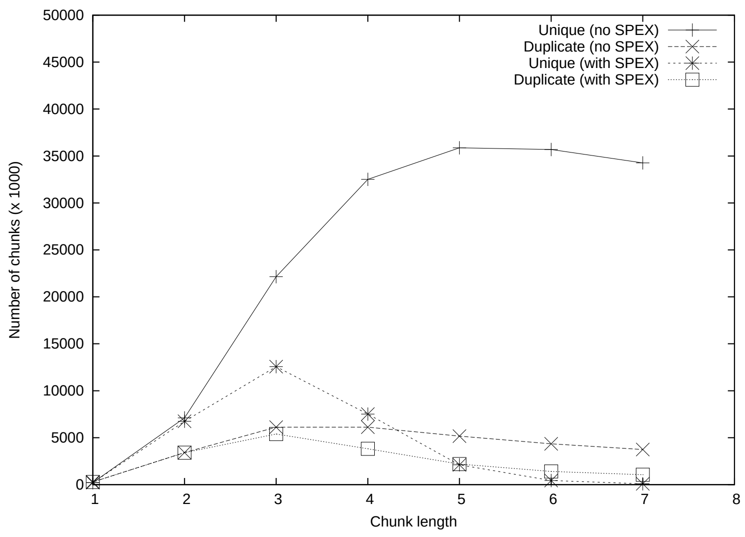


Figure 4.3: Number of hashcounter entries containing single and multiple hits at each iteration for the LATimes *collection with and without* SPEX *iterative refinement (*2^{23} *elements total).*

(1997). In order to minimise the possibility of collision, we use the reasonably secure MD5 message digest algorithm (Rivest, 1992), which produces hash-keys 128 bits in length.

Deco currently supports the SPEX and *modulo* (see Section 3.6.5) selection schemes, which can be used individually or together in the selection pipeline. By default, SPEX is used. However, it can be combined with the *modulo* heuristic so that a chunk is only indexed if it is selected by both algorithms. This allows for the creation of compact indexes that still support identification of co-derivatives. The selection pipeline code was written in such a way as to facilitate the creation of additional 'plug-ins' for the pipeline. For example, one could easily write a new selection filter for the *anchor* heuristic, or a filter that selected chunks only if they contained certain words.

4.3.2 Relationship Graph Generation

Several parameters must be specified to guide the process of generating a relationship graph from the common-chunk inverted index generated by the SPEX process. The most important of these are the *scoring function* and the *inclusion threshold*. Deco calculates the co-derivation score for a pair of documents u and v using one of the following formulae:

$$\begin{aligned}
S_1(u,v) &= \textstyle\sum_{c \in u \wedge c \in v} 1 \\
S_2(u,v) &= \textstyle\sum_{c \in u \wedge c \in v} 1/\min \bar{u}, \bar{v} \\
S_3(u,v) &= \textstyle\sum_{c \in u \wedge c \in v} 1/\operatorname{mean} \bar{u}, \bar{v} \\
S_4(u,v) &= \textstyle\sum_{c \in u \wedge c \in v} \frac{1/f_c}{\operatorname{mean} \bar{u}, \bar{v}}
\end{aligned}$$

where $\bar{u}$ is the length (in words) of a document u, and f_c is the number of collection documents a given chunk c appears in. Function S_1 above simply counts the number of chunks common to the two documents. Functions S_2 and S_3 attempt to normalise the score relative to the size of the documents, so that larger documents do not dominate smaller ones in the results. They are similar to the resemblance measure of Broder (1997) but are modified for more efficient computation. Function S_4 gives greater weight to chunks that are rare across the collection. These scoring functions are all simple heuristics. Further refinement of these functions and the possible use of statistical models is a topic for future research.

The inclusion threshold is the minimum value of $S(u,v)$ for which an edge between u and v will be included in the relationship graph. We wish to set the threshold such that pairs of co-derived documents score above the threshold while pairs that are not co-derived score below the threshold.

4.3.3 Implementation considerations

During the course of the implementation of the DECO system, a number of optimisations were made, the more noteworthy of which are described in this section.

Text tokenisation

The DECO package performs fingerprinting with word-level granularity. Thus, the incoming text stream must be tokenised into individual word-tokens before it can be processed. This process is reasonably computationally expensive. Furthermore, certain elements of the stream — for example, most HTML tags — may be ignored by the fingerprinting algorithm. Because the SPEX algorithm operates by performing multiple linear passes over the text collection, the collection is tokenised multiple times, leading to significant inefficiencies.

DECO overcomes this inefficiency by performing an initial tokenising pass over the plain-text collection, and storing the resulting stream of token IDs. As each token is returned by the parser, it is assigned a four-byte ordinal token ID. Thus, the first token in the stream will have ID 0, the next will have ID 1 and so on. A hashtable is maintained, so that if a token has appeared earlier in the stream then it will be associated with the same ID. Thus, each token in the stream has a one-to-one correspondence with a numeric token ID. For example, the following text:

```
Peter Piper picked a peck of pickled peppers
A peck of pickled peppers Peter Piper Picked
```

would represented as the following sequence of preparsed token IDs (assuming case-folding):

```
1 2 3 4 5 6 7 8
4 5 6 7 8 1 2 3
```

Once DECO has built the preparsed token file, all further processing (including SPEX) proceeds on this file rather than on the plaintext document collection. It should be apparent from the above example that if a pair of documents share chunks in the plaintext representation, they will likewise share chunks in the preparsed representation, and — in general — vice versa. This optimisation thus has no effect on the final output of the system. Note that it is possible to save one pass over the preparsed token file by piping the output of the tokenisation process directly to the input for the SPEX algorithm. However, this requires that the hashtable for building the token file be in memory contempereaneously with the hashcounters for SPEX. Depending on the available memory resources, this may or may not be a worthwhile tradeoff. We have not implemented this optimisation.

Text preparsing leads to speed improvements for two reasons. First, by only parsing the collection once, significant computational resources are saved; reading a stream of preparsed ordinal tokens is a far more efficient process than repeatedly parsing the original text. Second, the collection as represented by the preparsed token sequence is generally smaller than the plaintext representation of the same collection, for the following reasons: the four-byte ordinal tokens are typically somewhat smaller than the terms they replace; whitespace is not necessary; and ignored objects such as HTML tags may be omitted from the file. As disk read operations make a significant contribution to the overall execution time of DECO, a smaller collection leads directly to a speed improvement.

Text preparsing also has two potential drawbacks. The first of these is that the actual text of chunks that are shared between a pair of documents is lost. All that is left are sequences of token IDs. However, it is unlikely that the actual text of the chunks is a particularly valuable resource. Apart from debugging, the only use for these values is for visually presenting the overlap between documents to a user. With positional data in the index, however, this can be efficiently recovered directly from the documents. The second drawback is the possibility of collision due to overflow. If the number of unique tokens in a collection exceeds the bounds of the primitive used for storing the token ID — 2^{32} in the case of a four-byte integer — then some tokens will end up mapping to the same ID due to overflow, leading to false positives. The response to this is that as we are comparing *sequences* of tokens, the chance of a collision with chunks of a reasonable length in a space this large is exceedingly slim. Given that fingerprinting is not a precise science, this should not cause undue concern.

Hashcounter packing

Each entry in the hashcounters used for the SPEX algorithm must be able to distinguish between three values: zero, one, or 'greater than one'. Thus, entries in the hashcounter need only be two bits wide in order to store their values. This allows us to create hashcounters with a large number of entries within main memory.

However, we can do even better than this. Consider that one of the four states allowed by a two-bit field goes unused in the above implementation. From an information-theoretic perspective, we can calculate that each entry contains $\log_2 3 \approx 1.58$ bits of data. Thus, we are able to pack five hashcounter entries into a single eight-bit byte.

When inspecting or updating a particular field in the hashcounter, DECO first determines the byte in which the field occurs. Simple lookup tables are then used to either increment or return the current value of the particular field requested. While this system notionally requires an additional memory access for each operation, the lookup tables are sufficiently small and frequently accessed that they should be largely cache-resident. Thus, the additional cost is not onerous.

Using this implementation allows a hashcounter of 128 MB to contain 640 million separate fields, thus allowing SPEX to be effectively used even on large collections.

Index structure

DECO indexes chunks in a high-performance compressed index. Postings lists are document-ordered and document entries (with offsets) are stored as d-gapped, variable-byte compressed integers (Zobel and Moffat, 2006). As noted in Section 3.8, the discovery task does not make use of the actual value of the chunks that occur in multiple documents, but only of the co-occurrence information encapsulated in the postings lists for each chunk. As such, vocabulary and chunk values are omitted from the final inverted index. Note however that the chunk values are still required before partial indexes are merged into the final index, as they constitute the key upon which lists are merged.

Some of the low-level index operations in DECO are based on code from the Zettair[1] search engine developed at RMIT University.

Long postings lists

At present, the problem of long postings lists as discussed in Section 3.9 is managed by a combination of chunk stopping (see Section 3.9.1) and coarse counting (Section 3.9.2). While these measures prove satisfactory in many situations, long postings lists continue to present an efficiency bottleneck. New, more sophisticated measures for dealing with this issue are a topic for future research.

4.4 Experimental methodology

We seek to experimentally investigate two facets of the DECO package: the accuracy and reliability of the package in identifying co-derivative document pairs, and the scaling characteristics of the system.

4.4.1 Document collections

In this section we describe the six document collections that we use in our experiments:

webdata+xml The *webdata+xml* collection was created by Hoad and Zobel (2003) and consists of 3,307 web documents (approximately 35MB) extracted from the TREC web collection (Harman, 1995). Into this background collection have been seeded nine documents (the *XML documents*), all of which are co-derivatives. This set of co-derivatives was created by distributing a single document discussing XML technology to nine separate authors, and asking them to edit it substantially. In some cases the edits are so substantial that the final document shares little text with the original. Nonetheless, by our definition, all these documents should be classified as co-derived.

linuxdocs The *linuxdocs* collection was created by Hoad and Zobel (2003) and consists of 78,577 documents (720 MB) drawn from the documentation included with several distributions of RedHat Linux. While the *webdata+xml* collection serves as an artificial but

[1]`http://www.seg.rmit.edu.au/zettair/`

easily-analysed testbed for co-derivative identification algorithms, the *linuxdocs* collection, rich in duplicate and near-duplicate documents, is a larger and more challenging real-world collection.

all-newswire The *all-newswire* collection is an aggregation of the newswire collections gathered for the TREC project (Harman, 1995). This collection includes a large number of articles from newswires originating from sources including the Wall Street Journal, the Los Angeles Times, Associated Press, and the Financial Times. This collection is about 5.3 GB in size, and is used to investigate the scaling properties of the DECO package.

LATimes The *LATimes* collection is a subset of the *all-newswire* collection, consisting of 476 MB of articles from the Los Angeles Times. This collection is used to investigate the index growth we may expect from a typical collection of documents.[2]

GOV1 & GOV2 The *GOV1* and *GOV2* collections were created for the TREC project. Both are crawls of the `.gov` domain, the former consisting of 18.1 GB of data, the latter of 426 GB of data, making it the largest publicly available standard collection of web documents.

4.4.2 Metrics for evaluation

We suppose that each collection of documents has associated with it an underlying *reference graph*, the 'true' relationship graph for that document. Note that this is an idealised construct: at the fringes, the question of whether a given pair of documents is co-derived is a subjective one. Nonetheless, it is reasonable to assume that a relationship graph carefully constructed by an expert, unbiased human judge provides a good approximation of the reference graph.

We define the *coverage* of a given computer-generated relationship graph to be the proportion of edges in the reference graph that are also contained in that graph, and the *density* of a relationship graph to be the proportion of edges in that graph that also appear in the reference graph.

While coverage and density are in many ways analogous to the standard recall and precision metrics, we choose the new terminology to emphasise that the task is different to

[2]Though in one sense there is no such thing as a 'typical' collection of documents, *LATimes* is neither cleaned of co-derivatives nor deliberately contrived to contain them, in contrast to the *webdata+xml* and *linuxdocs* collections.

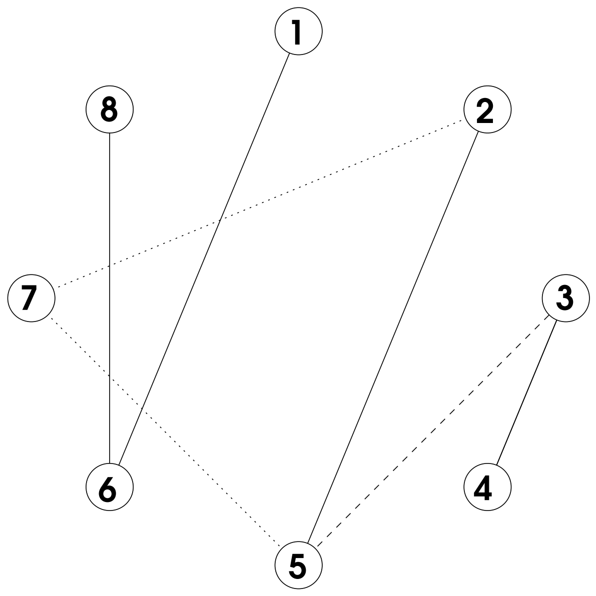


Figure 4.4: An example of a computer-generated relationship graph. The dashed line indicates a spurious inclusion; the two dotted lines show co-derivation relationships that were not detected by the algorithm.

querying. We are not trying to meet an explicitly defined information need, but are rather attempting to accurately identify existing relationships within the collection.

Figure 4.4 shows a hypothetical example of a computer-generated relationship graph for the same collection as Figure 2.2. The dashed line shows a spurious co-derivation judgement between documents 3 and 5. The dotted lines show the omission of co-derivation relationships between documents 2 and 7, and between documents 5 and 7. Given that the algorithm identified four of the six co-derivation relationships in the reference graph, it has a coverage of 67%. As four of the five co-derivation relationships identified existed in the reference graph, it has a density of 80%.

While the notion of a human-generated reference graph is a useful one, it is infeasible to construct one for any but the smallest of document collections. Doing so would require side-by-side judgements of every document pair in the collection to be made. For a relatively modest collection of 10,000 documents this amounts to nearly 50 million judgements. Given the impracticality of generating a full reference graph, it is not possible to directly determine the coverage and density of the relationship graph generated by an algorithm. We must therefore resort to estimates.

To estimate the density of a relationship graph, we take a random selection of edges from the graph and judge whether the documents they connect are in fact co-derived. To estimate the coverage of a relationship graph, we select some representative documents and manually determine a list of documents with which they are co-derived. The coverage estimate is then the proportion of the manually determined pairings that are identified in the relationship graph. A third metric, average precision,[3] is the average proportion of co-derivative edges to total edges for the documents selected to estimate coverage. That is, given a target document, we consider the manually-assigned list of co-derivatives to be relevant, and all other documents irrelevant. The average precision simply computes the precision of the list returned by DECO with reference to this relevant set. While average precision is arguably biased, and consequently is to be considered an inferior measure to density, it plays a role in experimentation because it is far less time-consuming to calculate.

4.5 Efficiency considerations

In this section we empirically evaluate the efficiency and scalability of the SPEX algorithm, and the effect of adopting certain optimisations on improving these values

4.5.1 Index growth rate

In order to investigate the growth trend of the chunk index produced by DECO as the source collection grows, we extracted subcollections of various sizes from the *LATimes* and *linuxdocs* collections, and observed the number of duplicate chunks extracted as the size of the collection was increased.

This growth trend is important for the scalability of SPEX and by extension the DECO package. For example, if the growth trend were quadratic, this would set a practical upper bound on the size of the collection that could be submitted to the algorithm, whereas if the trend were linear or $n \log(n)$ then far larger collections would become practical.

We found that for the tested collections at least, the growth rate follows a reasonably precise linear trend, as illustrated in Figure 4.5. While further testing is warranted, a linear growth trend suggests that the algorithm has potential to scale extremely well.

4.5.2 Scalability to larger collections

In order to test the ability of SPEX and DECO to scale to larger collections, we ran DECO on the *all-newswire* collection. This took a little over four hours on a lightly-loaded machine

[3]Not to be confused with the average precision calculated over ranked lists, as discussed in Section 2.3.5.

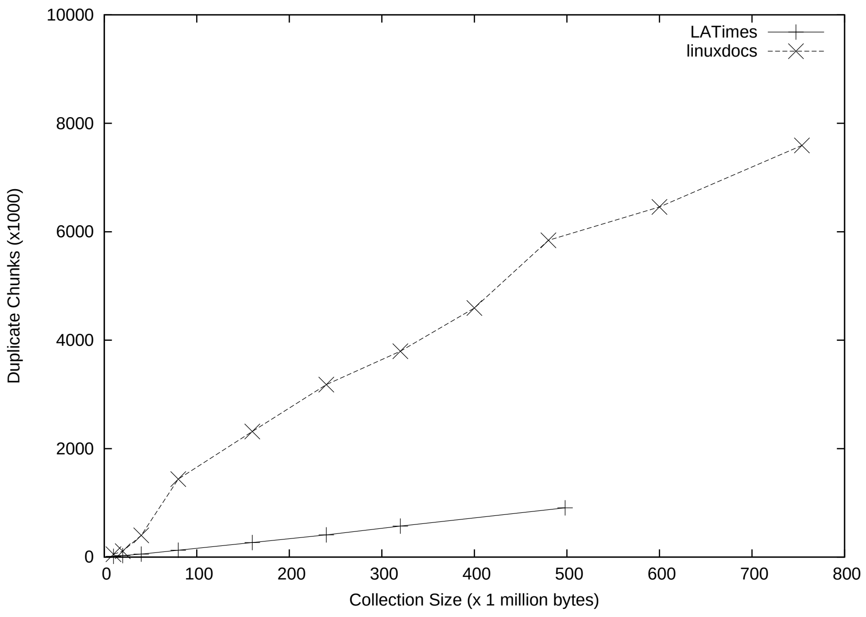


Figure 4.5: The total number of duplicate chunks found as the size of the LATimes *collection is increased.*

with dual Intel Pentium III processors and 768 MB of RAM. We consider an indexing speed in excess of one GB per hour on an older machine such as the one used to be acceptable for this application. The index generated by DECO was approximately 1.0 GB in size, or less than 20% of the size of the source collection.

DECO has also been used to successfully index the 18.1 GB GOV1 collection, with similar performance to that described for the *all-newswire* collection above. However, indexing the 426 GB GOV2 collection has thus far proved impossible using DECO on a standard production workstation. This is due less to the collection's size than the extremely high level of duplication within GOV2. The degree of duplication causes the collection to violate the working assumption of the SPEX algorithm, namely that a large majority of the chunks occurring in a collection are singleton.

Our experience with GOV2 shows that the SPEX algorithm is most suited to collections with a modest level of internal duplication. The relationship between collection size and memory requirements is ultimately dependent upon the level of redundancy in the particular collection being indexed. The level of redundancy at which the use of SPEX ceases to be worthwhile is dependent on various factors, particularly the amount of memory and disk available, and the time-sensitivity of the fingerprinting job. As a general rule, however, a strong case can be made for using SPEX on collections in which is overall level of redundancy is at or below 25%.

Despite this caveat, our investigations demonstrate that the algorithm is able to scale to quite large collections without encountering resource limitations.

4.5.3 Effect of identifying and excluding exact duplicates

As discussed in Section 4.3.1, DECO includes an optional preprocessing stage in which exact duplicate documents are omitted from the indexing process. The justification for this is that it is relatively easy to identify exact duplicates and that they consume a disproportionate amount of space in the chunk index.

Employing this preprocessing step for the *all-newswire* collection reduced the size of the resultant shared-chunk index by approximately 10%. While this is not a spectacular result, it is nonetheless a worthwhile saving.

When investigating the reason for the modesty of the saving, we found that a large number of documents were nearly identical, but differed by just a few terms. In many newswires, for example, the same article was released multiple times for different editions. As the edition heading differed, the documents were not considered identical and thus no exclusions resulted. While this problem might be especially characteristic of newswire collections, there are many cases where one can imagine such trivial differences thwarting this simple preprocessing step.

One possibility for improvement is to use a single-fingerprint approach such as I-Match (see Section 3.7) in order to capture documents that are not exact duplicates. However, I-Match is not as fast as simply hashing entire documents, and using a more aggressive scheme for identifying duplicates introduces the possibility of false positives. In Chapter 5 we discuss a more conservative improvement that first *canonicalises* documents before hashing them in order to eliminate trivial differences.

4.6 Effectiveness in identifying co-derivatives

In this section we empirically investigate the effectiveness of fingerprinting using DECO and the SPEX algorithm under a range of parameterisations.

For testing purposes, we established a range of five inclusion thresholds for each scoring function, making for a total of 20 combinations. These thresholds are named T_1 to T_5 in order of increasing value. Because of the different properties of the various scoring functions, the values are not the same for each of T_1 to T_5. Note that the thresholds are not therefore

	T_1	T_2	T_3	T_4	T_5
S_1	20	100	200	500	1000
S_2	0.03	0.05	0.10	0.15	0.30
S_3	0.02	0.05	0.10	0.15	0.30
S_4	0.02	0.05	0.20	0.50	1.0

Table 4.1: The values of thresholds T_1 through T_5 for the various scoring functions

designed to be directly comparable across scoring functions, but are intended to exhibit a broad range of possible choices for each of the scoring functions S_1 to S_4. Exact values are listed in Table 4.1.

In this section, we use the notation $S_x/T_y/P_z$ to denote a run with scoring function S_x, threshold T_y and modulo parameter p of z (see Section 3.6.5). For example, run $S_3/T_2/P_{16}$ indicates the run using scoring function S_3, threshold T_2 and a modulo parameter p equal to 16.

4.6.1 Effectiveness on the *webdata+xml* collection

Because the *webdata+xml* collection contains the nine seed documents for which we have exact knowledge of co-derivation relationships, it makes a convenient collection for proving the effectiveness of the DECO package and determining good parameter settings. Using DECO to create a chunk index with a chunk size of eight took under one minute on a 2.67 GHz Intel Pentium 4 PC with 512 MB of RAM. For this collection, we tested DECO using the four scoring functions described in Section 4.3.2. In all cases the *modulo* selection function was not used; that is, p was equal to one.

Combining the four scoring functions with the thresholds T_1 to T_5 led to the generation of 20 relationship graphs.‘ Each of these was then examined for the presence of the 36 edges connecting the XML documents to each other.

As can be seen in Table 4.2, the estimated coverage values strongly favour the lower inclusion thresholds. Indeed, for all scoring functions using the inclusion threshold T_1, 100% of the pairings between the XML documents were included in the relationship graph. In all cases the average precision was also 100%. These values — 100% coverage and 100% average precision — suggest a perfect result, but are certainly overestimates. The nature of the test collection — nine co-derived documents seeded into an entirely unrelated background

	T_1	T_2	T_3	T_4	T_5
S_1	100.0	97.2	36.1	8.3	0.0
S_2	100.0	100.0	83.3	58.3	25.0
S_3	100.0	91.7	72.2	52.8	16.7
S_4	100.0	97.2	91.7	58.3	22.2

Table 4.2: Coverage estimates, as percentages, for the webdata+xml *collection calculated on the percentage of XML document pairings identified ($p = 1$). The average precision was 100% in all cases.*

collection — made it extremely unlikely that spurious edges would be identified. This not only introduced an artificially high average precision estimate but also strongly biased the experiments in favour of the lower inclusion thresholds, because they allowed all the correct edges to be included with very little risk that incorrect edges would likewise be admitted.

4.6.2 Effectiveness on the *linuxdocs* collection

In order to test DECO in a less artificial environment, we repeated our experiments on the *linuxdocs* collection. We again used DECO to create a shared-chunk index with a chunk size of eight. This took approximately 30 minutes on a 2.67 GHz Intel Pentium 4 PC with 512 MB of RAM. For generation of relationship graphs we used the same range of scoring functions and inclusion thresholds as in the previous section. Once again, p was in all cases set to one.

To estimate the coverage of the relationship graphs, we selected ten query documents from the collection representing a variety of sizes and types. Given that the collection was a collection of online documentation, these queries included both long and short man pages, as well as HTML and info style documentation. For each of these query documents, we manually collated a list of co-derivatives. As it was infeasible with the resources available to undertake a comprehensive manual investigation of all the documents in the collection, the list of co-derivatives was compiled with the aid of some standard tools and by making use of domain knowledge. In particular, the nature of the collection gives us very reliable keywords for search: in the evolution of the documentation for a command, one thing that is very unlikely to change is the name of the command itself. Also, the naming conventions for the documentation files were relatively consistent. Thus, with the aid of the standard Unix `ls` and `grep` commands, we were able to compile lists of candidate co-derivatives for each query document with relative ease. Given this methodology, the lists of co-derivatives for each query document cannot be asserted to be entirely comprehensive. However, the nature of the collection gives us a reasonable degree of confidence that not too many of the co-derivation relationships were missed.

Estimated coverage and average precision results for this set of experiments are given in Table 4.3.

The results of these experiments demonstrate that document fingerprinting using the SPEX algorithm is a practical and effective approach to the discovery problem for co-derivatives. Several of the combinations of scoring function and inclusion threshold yielded accurate results. In some cases, coverage was estimated at 100% while maintaining high average precision, which in other cases average precision was maintained at 100% and nearly all co-derivative pairs were identified. While none of the combinations yielded a perfect result, a few came very close.

In general, the pattern of results demonstrates a classic tradeoff between recall (coverage) and precision (density/average precision). By reducing the inclusion threshold, the likelihood of finding a majority of co-derived pairs increases, but so does the likelihood of false positives. This suggests that there is no ideal combination of scoring function and inclusion threshold, but that the particular choice depends upon the requirements of the anticipated application. The very different spread of results for the *webdata+xml* and *linuxdocs* collections suggests that the choice of threshold is likely to be data-dependent and thus require tuning for optimal performance.

It is interesting to note that the coverage/average precision values that were observed in these experiments are far higher than the recall/precision values typical of topical information retrieval experiments. This confirms that the syntactic-level task of co-derivative detection is indeed a less difficult problem than the semantic one of finding documents relevant to a user's information need. The relative tractability of the problem and corresponding high accuracy are encouraging and bode well for the use of document fingerprinting for the management of digital collections.

The scoring functions S_2, S_3, and S_4 were more effective than the simple chunk-counting S_1 scoring function. This demonstrates the importance of normalising results to the size of the documents in question. However, there is little in terms of effectiveness to distinguish between the latter three scoring functions. In terms of selecting which of these scoring functions to use, then, the choice is somewhat arbitrary. One could argue that S_4 should not be chosen because there is no performance improvement in return for its increased complexity.

	T_1	T_2	T_3	T_4	T_5
S_1	100/ 70	89/ 71	56/ 93	36/ 95	34/100
S_2	100/ 57	100/ 75	100/ 92	89/ 94	57/100
S_3	98/ 75	96/ 84	94/100	84/100	47/100
S_4	99/ 83	96/ 91	94/100	78/100	30/100

Table 4.3: Coverage and average precision estimates, as a pair X/Y of percentages, for DECO *applied to the* linuxdocs *collection, using a full shared-chunk index ($p = 1$).*

In future chapters we choose to use S_3 as the scoring function for use with the discovery problem; however, S_2 could have done just as well.

We had insufficient human resources to complete an estimate of density for all of the relationship graphs generated. Instead, we selected a range of configurations that seemed to work well and estimated the density for these configurations by picking 30 random edges from the relationship graph and manually assessing whether the two documents in question were co-derived. The results corresponded closely with the average precision for the same runs: $S_2/T_3/P_1$, $S_3/T_2/P_{256}$, and $S_4/T_3/P_{16}$ all scored a density of 93.3% (28 out of 30) while $S_4/T_3/P_1$ and $S_2/T_1/P_{16}$ both returned an estimated density of 100%. This suggests that average precision is a good predictor of the true density value.

4.6.3 Effects of introducing lossy selection

In order to test the relationship between the lossiness of a selection heuristic and the degradation in the reliability of co-derivative identification, we performed the same experiments on the *linuxdocs* collection as above, this time with the *modulo* heuristic added to the selection pipeline. We experimented with the p operator set to 16 and 256. A value of 16 represents a value that is similar to many in the literature, while 256 studies the effect of a more aggressive pruning. The only change to the experimental parameters was that the inclusion thresholds for these experiments were adjusted downward commensurately with the *modulo* operator so as not to bias the results. In other words, the thresholds were divided by 16 and 256 respectively, as this is the expected decrease in the number of shared chunks between a given pair of documents. While this does not precisely mimic the behaviour of the case when $p = 1$ — we are now dealing with probabilities rather than certainties — these are nonetheless the most appropriate thresholds. A document meeting the minimum threshold

	T_1	T_2	T_3	T_4	T_5
Fingerprinting modulo *16*					
S_1	90/ 72	88/ 76	56/ 94	36/ 96	34/100
S_2	90/ 75	90/ 75	80/ 94	78/100	57/100
S_3	88/ 82	86/ 91	74/100	74/100	47/100
S_4	88/ 85	86/ 93	86/ 93	69/100	60/100
Fingerprinting modulo *256*					
S_1	54/ 95	54/ 95	54/ 95	54/ 95	34/ 97
S_2	54/ 97	54/ 97	54/ 97	54/ 97	44/ 97
S_3	54/ 97	54/100	54/100	51/100	42/100
S_4	54/ 97	54/100	54/100	44/100	31/100

Table 4.4: Coverage and average precision estimates, as a pair X/Y of percentages, for DECO *applied to the* linuxdocs *collection for indexes that store chunks only if their hash-key equals zero* modulo *16 and 256.*

T in the case of $p = 1$ would be expected to have a score of $\frac{T}{z}$ when $p = z$.

The results for these experiments are presented in Table 4.4. When the *modulo* heuristic is used with $p = 16$, the results are noticeably inferior to those using the full lossless chunk index generated by SPEX. Nonetheless, many of the value-pairs represent a compromise that is acceptable in many practical situations. In an application where finding every single co-derived document pair is not critical, the loss of a few percentage points in reliability in return for a sixteen-fold reduction in the number of chunks indexed might be an attractive trade-off.

For the *modulo* 256 index, no configuration was able to find more than 54% of the relevant edges. This is almost certainly because the other 46% of document pairs simply do not have any chunks in common that evaluate to 0 *modulo* 256 when hashed using our particular hash function. This risk exists to a certain degree with any lossy selection scheme, but especially so when the lossiness is too aggressive. Two documents with a significant number of shared chunks could be overlooked even at the lowest thresholds because not one of their shared chunks is included in the index. In many situations this would not be acceptable.

Interestingly, though the *modulo* 256 configurations were unable to achieve a high coverage value, in many of the configurations the average precision was near 100%. This ability to accurately (though not comprehensively) identify co-derivative documents even in this very lossy configuration demonstrates the robustness of document fingerprinting, and goes some way to explaining why experiments with highly selective fingerprinting schemes such as in Broder et al. (1997) are still able to meet with reasonable success.

4.7 Summary

Document fingerprinting using chunking is an appealing choice for detecting co-derivative documents, and has been successfully used in the past for a diverse range of such applications (Manber, 1994; Brin et al., 1995; Shivakumar and García-Molina, 1995; Broder, 1997). However, high resource usage of the full chunking approach has led to the use of selection heuristics to only retain a subset of chunks from each document. We have argued that, with existing chunk-selection heuristics, one can have either reliability or acceptable resource usage, but not both at once.

In this chapter we have introduced the SPEX algorithm for efficiently identifying and discarding the singleton chunks in a collection. Singleton chunks represent a large proportion of all chunks in many collections — over 98% in one of the collections tested — but play no part in discovery of co-derivatives. Identifying and discarding these chunks means that document fingerprints only contain data that is relevant to the co-derivative discovery process. In the case of the *LATimes* collection, this allows us to create an index that is functionally equivalent a complete index of all chunks but contains only one fiftieth of the postings lists. Such savings allow us to implement a system that is effective and reliable yet requires only modest resources.

Experiments on our DECO system (which uses the SPEX algorithm) on a variety of test collections have demonstrated that the package is capable of reliably discovering co-derivation relationships within a collection, and that introducing heuristic chunk-selection strategies degrades reliability.

There is significant scope for further work and experimentation with DECO. Although we have demonstrated that the system is able to scale to quite substantial collection sizes, it is important to continue to investigate additional optimisations to the algorithms and systems. This is both to increase the speed of the system and to allow it to scale to the extremely large multi-terabyte collections that are being managed in many domains.

We found that collections such as GOV2 that contain an extremely high level of document duplication cause SPEX to perform poorly in terms of resource economy. This is most likely because such high levels of redundancy violate the working assumption of the SPEX algorithm that a large proportion of all chunks in the collection are singletons. Increasing the robustness of SPEX so that it can work effectively when this assumption is violated must be an important objective of future work, although it is unclear how this can be achieved without fundamentally altering the nature of the algorithm.

A limitation of SPEX is that it is a one-shot algorithm: it must have access to the entire collection in order to build a shared-chunk index for it. It is not possible to later add additional documents to the collection without rebuilding the entire index. The difficulty of extending the index is the one major defect of SPEX compared to many other fingerprinting selection heuristics. We believe that it is possible to write an adjunct to SPEX that, while it would carry some overhead, would allow for incremental building of the shared-chunk index as new documents are added to the collection.

Chapter 5

Managing redundancy in web search

This chapter contains material that first appeared in Bernstein and Zobel (2005).

Web search engines try to present information to users in such a way as to allow them to minimise the effort required to satisfy their information needs. Broadly, they attempt to meet this aim by following the probability ranking principle, as discussed in Section 2.3.3. While this approach is in general reasonably robust, the various simplifications made in formulating the probability ranking principle can lead to distortions in which success according to this principle does not accord with success in terms of the fundamental goal of minimising user effort.

In this chapter, we concentrate on one such distorting factor: redundancy of information between documents. It should be obvious that presenting a user with a ranked list in which the top-ranked documents, though relevant, have a high degree of mutual redundancy, will not in general meet the goal of allowing the user to satisfy their information need with a minimum expenditure of effort. The problem of redundancy is compounded by the fact that most commonly-used effectiveness evaluation methodologies do not take it into account. This means that the problem is not evident in system-oriented evaluation of performance. It is therefore important that the level of redundancy be explicitly taken into consideration both when ranking documents and when evaluating search engine performance.

There have been several attempts to address the issue of redundancy in ranked lists. The maximal marginal relevance (MMR) model (Carbonell and Goldstein, 1998) iteratively considers the degree of redundancy of documents with respect to the ranked list when selecting documents to add to the list. The TREC novelty track (Harman, 2002; Soboroff and Harman, 2003), the aim of which is to extract novel and relevant sentences from an ordered list of relevant documents, has provided evaluation of various techniques for decreasing redundancy in result lists. However, it is unclear whether it would be feasible to add any of the methods evaluated in the track to a standard search engine (Allan et al., 2003). Fundamentally, the generalised novelty task as defined in the novelty track suffers because it is intractable: novelty as a concept is difficult to define, recognize, and evaluate.

In this chapter, we define and explore a restricted form of the redundancy relation that is more tractable both in terms of ranking and evaluation. *Content equivalence* is a query-independent relation between pairs of documents whereby they are considered content-equivalent if they have the same information content. Pairs of exact duplicates are content-equivalent, but various factors — such as formatting — can lead to significant superficial variations between content-equivalent documents. Nonetheless, we hypothesize that content equivalence is a relation that leaves significant syntactic evidence. Thus, we apply both a simple document canonicalization technique and the DECO fingerprinting package to the task of identifying content-equivalent document pairs in a document collection.

This chapter is structured as follows. In Section 5.1 we discuss the issue of inter-document redundancy in search results, and existing approaches to management of redundancy. In Section 5.2 we define content equivalence and justify the usefulness of such a definition. In Section 5.3 we report on the results of a user experiment undertaken to set a score threshold for content equivalence using the DECO system. We then report on experiments investigating the prevalence of content equivalence in the GOV1 and GOV2 web data collections (Section 5.6), and on ranked result lists from the 2004 TREC terabyte track (Section 5.7). Finally, in Section 5.8 we evaluate and discuss the level of inconsistency in judgements for the topics in the 2004 TREC terabyte track.

Our approach is deliberately conservative, because in this application an error-prone algorithm may inadvertently conceal essential information from the user. Nonetheless, we show that we are able to identify large numbers of content-equivalent document pairs in the TREC GOV1 and GOV2 collections. More significantly, our experiments show that content equivalence is frequent between documents co-occurring in ranked lists submitted to the 2004 TREC terabyte track. This finding has two consequences: first, that the effectiveness of search systems is being significantly overestimated, because the degree of redundancy in the ranked lists is not taken into account; and second, that the effectiveness of many search systems could be substantially increased by detecting and removing content-equivalent documents from ranked lists as a postprocessing step.

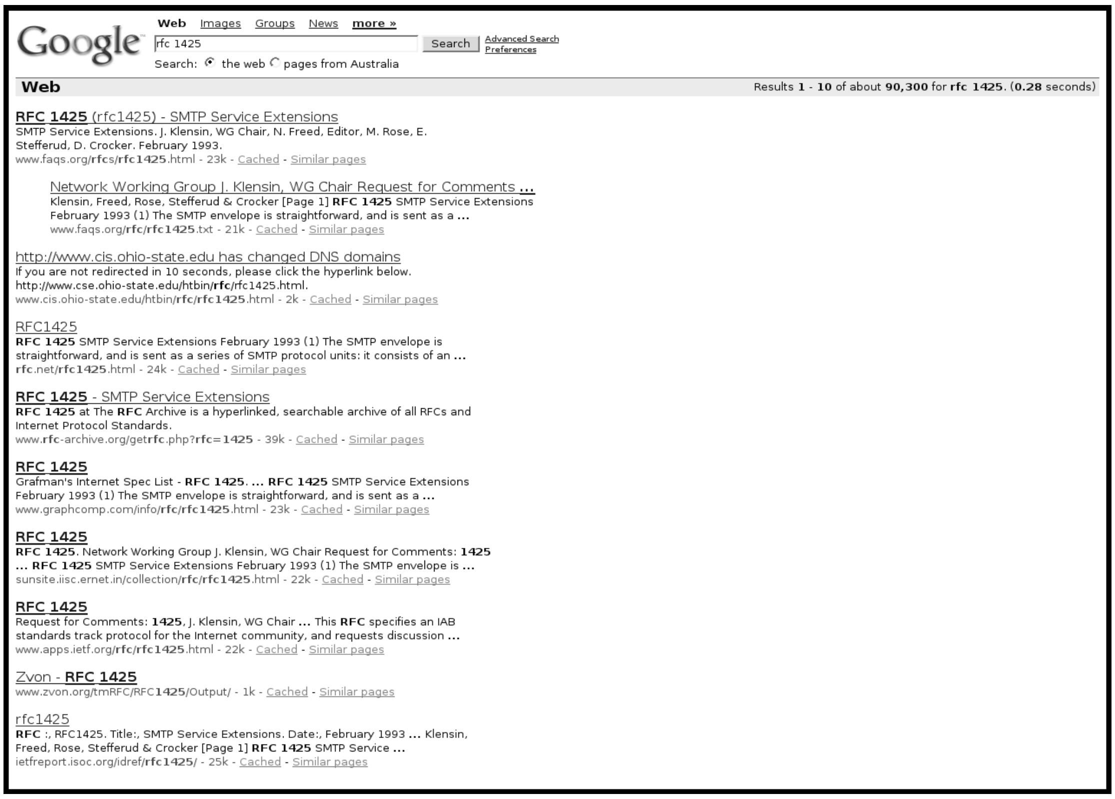
Another interesting result to come from our analysis concerns the degree of inconsistency in the relevance judgements for the 2004 TREC terabyte track. Discovery of content-equivalent document pairs gave us an opportunity to automatically estimate how consistent judges have been in assigning relevance values to documents. Our results show a significant number of contradictory relevance judgements, in which the documents in a content-equivalent pair were given opposing relevance judgements. This raises difficult-to-answer questions regarding system-evaluation methodology.

In this chapter we have focused on evaluation of the redundancy problem in web data. While the particulars may vary, the issue of redundancy exists in many other types of document collections, and must similarly be addressed. Many of our observations in this chapter can be generalised to other types of document collections.

5.1 Redundancy in search results

In Chapter 2 we asserted that the operation of search engines is guided by the principle of utility, and that in this context the task of the search engine is to help a user satisfy their information need with a minimal expenditure of effort. As this goal is too abstract to implement directly, most search systems are guided by the probability ranking principle, which states that documents should be ranked in order of decreasing probability of relevance to the user. Relevance is, in turn, generally defined as a relation solely between a document and the user's query or information need. Evaluation metrics such as MAP and R-precision similarly emphasize the probability ranking principle by rewarding a concentration of relevant documents amongst the top rankings in a result list.

One deficiency of the probability ranking principle as it is used is that it takes no account of the widespread phenomenon of inter-document redundancy. It is often the case that a document, though relevant in isolation, contains nothing new or interesting by the time it has been viewed by the user. It follows that, for typical querying, the presentation of such a document should not be considered beneficial. For some information needs, the existence or location of redundant documents may be of value to the user, but such cases are probably rare. In most cases, it is the content itself that the user seeks. It follows that in order for


Web Images Groups News more »
rfc 1425
Search
Advanced Search
Preferences
Search: the web pages from Australia

Web Results **1** - **10** of about **90,300** for **rfc 1425**. (**0.28** seconds)

RFC 1425 (rfc1425) - SMTP Service Extensions
SMTP Service Extensions. J. Klensin, WG Chair, N. Freed, Editor, M. Rose, E. Stefferud, D. Crocker. February 1993.
www.faqs.org/**rfcs**/**rfc1425**.html - 23k - Cached - Similar pages

Network Working Group J. Klensin, WG Chair Request for Comments **...**
Klensin, Freed, Rose, Stefferud & Crocker [Page 1] **RFC 1425** SMTP Service Extensions February 1993 (1) The SMTP envelope is straightforward, and is sent as a **...**
www.faqs.org/**rfc**/**rfc1425**.txt - 21k - Cached - Similar pages

http://www.cis.ohio-state.edu has changed DNS domains
If you are not redirected in 10 seconds, please click the hyperlink below. http://www.cse.ohio-state.edu/htbin/**rfc**/rfc1425.html.
www.cis.ohio-state.edu/htbin/**rfc**/**rfc1425**.html - 2k - Cached - Similar pages

RFC1425
RFC 1425 SMTP Service Extensions February 1993 (1) The SMTP envelope is straightforward, and is sent as a series of SMTP protocol units: it consists of an **...**
rfc.net/**rfc1425**.html - 24k - Cached - Similar pages

RFC 1425 - SMTP Service Extensions
RFC 1425 at The **RFC** Archive is a hyperlinked, searchable archive of all RFCs and Internet Protocol Standards.
www.**rfc**-archive.org/get**rfc**.php?**rfc=1425** - 39k - Cached - Similar pages

RFC 1425
Grafman's Internet Spec List - **RFC 1425**. **... RFC 1425** SMTP Service Extensions February 1993 (1) The SMTP envelope is straightforward, and is sent as a **...**
www.graphcomp.com/info/**rfc**/**rfc1425**.html - 23k - Cached - Similar pages

RFC 1425
RFC 1425. Network Working Group J. Klensin, WG Chair Request for Comments: **1425** **... RFC 1425** SMTP Service Extensions February 1993 (1) The SMTP envelope is **...**
sunsite.iisc.ernet.in/collection/**rfc**/**rfc1425**.html - 22k - Cached - Similar pages

RFC 1425
Request for Comments: **1425**, J. Klensin, WG Chair **...** This **RFC** specifies an IAB standards track protocol for the Internet community, and requests discussion **...**
www.apps.ietf.org/**rfc**/**rfc1425**.html - 22k - Cached - Similar pages

Zvon - **RFC 1425**
www.zvon.org/tmRFC/RFC**1425**/Output/ - 1k - Cached - Similar pages

rfc1425
RFC :, RFC1425. Title:, SMTP Service Extensions. Date:, February 1993 **...** Klensin, Freed, Rose, Stefferud & Crocker [Page 1] **RFC 1425** SMTP Service **...**
ietfreport.isoc.org/idref/**rfc1425**/ - 25k - Cached - Similar pages

Figure 5.1: An example of a search result that has high measured effectiveness but is unsatisfactory to the user due to redundancy.

a document to be of value to the user, it must not just be relevant to the information need but must also contain an element of *novelty*.

Figure 5.1 shows an instance of a search in which a popular commercial web search engine has returned many copies of the same document in response to a search for that document. The ranked list in the figure is not likely to be perceived by the user as an optimal arrangement. The documents are highly redundant with respect to each other, and the user would gain little from viewing all of them. However, the result would be considered highly effective with reference to the probability ranking principle: the top ten results are all unquestionably relevant to the query according to the commonly employed system-oriented definition of relevance. Furthermore, this result would be awarded a high score by any of the effectiveness metrics discussed in Section 2.3.5.

5.1.1 Redundancy and the cluster hypothesis

The cluster hypothesis (Jardine and van Rijsbergen, 1971) conjectures that similar documents are likely to be relevant to the same information needs. In other words, relevance tends to occur in clusters. This makes intuitive sense: one expects that documents addressing the same topic (and hence relevant to that topic) are more likely to have high similarity than a randomly selected pair of documents. The cluster hypothesis is generally accepted, and has been the subject of several empirical evaluations (Jardine and van Rijsbergen, 1971; Voorhees, 1985).

The cluster hypothesis has influenced search at several levels. Most directly, it has been the motivation behind cluster-based retrieval (Jardine and van Rijsbergen, 1971; Voorhees, 1985; Hearst and Pedersen, 1996; Liu and Croft, 2004), a search paradigm in which the user is presented with a series of document clusters rather than a ranked list of documents. Diaz (2005) proposed *regularization*, in which the cluster hypothesis is directly incorporated into the document scoring model, demonstrating in the process that the cluster hypothesis underlies successful techniques such as pseudo-relevance feedback (Ruthven and Lalmas, 2003).

The link between cluster hypothesis and query expansion techniques such as pseudo-relevance feedback suggests that their use has the potential to increase redundancy in ranked lists: highly redundant documents (and particularly duplicates and near-duplicates) will naturally cluster very well. Thus, both the simple fact of the cluster hypothesis and its application to increase the level of clustering in result lists suggest that redundancy is implicitly encouraged under the current independent-relevance model.

Such a divergence between the fundamental objective of information retrieval and the principles along which we design and evaluate our retrieval systems is a cause for serious concern. We can easily fall into the trap of pursuing objectives that are beneficial in terms of the simplified principle but are detrimental to the fundamental objective. Thus, when such a divergence is noted, it is necessary to rectify the model in order to bring it into line with the fundamental objective.

5.1.2 Modelling redundancy

There have been various attempts to incorporate redundancy into the model used for ranking documents. We describe the more pertinent of these attempts in this section.

Maximal marginal relevance

The maximal marginal relevance (MMR) technique aims to promote diversity by iteratively scoring documents according to their *marginal relevance* with respect to the current ranked list (Carbonell and Goldstein, 1998). The marginal relevance can be interpreted as an estimate of the 'relevant novelty' of a document: the degree to which the relevant information contained in the document is also novel. MMR builds the ranked list iteratively by selecting for each successive ranking the document that has the highest marginal relevance MR with reference to those documents already in the list. The iterative process selects the document to occupy position $i + 1$ in the list as follows:

$$r_{i+1}(q) = \operatorname*{argmax}_{d \in C} \left[\lambda \cdot S_1(d, q) - (1 - \lambda) \cdot \max_{n=1}^{i} S_2(d, r_n) \right] \tag{5.1}$$

where C is the collection being searched, S_1 and S_2 are any similarity scoring functions, and λ is a tuning parameter. When $\lambda = 1$, the ranking generated will be a standard relevance ranking according to S_1. When $\lambda = 0$ the algorithm will attempt to maximise the diversity of the ranked list according to S_2 without any reference to the estimated relevance of documents. For all other values of λ a document will be scored according to a combination of its relevance to the query and its novelty with respect to those documents that precede it. Thus, if a document is highly similar to a document that already occurs in the ranked list, its score will be significantly degraded.

Maximal marginal relevance is a robust framework for incorporating redundancy considerations into the ranking process, and has the convenience of being able to be applied on top of existing scoring models that have shown themselves to be effective. MMR does have deficiencies, however. In particular, the fact that the marginal relevance of a document is calculated as a linear combination of relevance and novelty scores belies the name of the model: it does not truly attempt to calculate a marginal relevance, as relevance and novelty are treated as orthogonal. Furthermore, MMR imposes a significant computational burden during ranking. For each rank position, the algorithm must recompute the scores for all documents. In a standard ranking process, scores must only be calculated once.

Novelty

The TREC novelty track (Harman, 2002; Soboroff and Harman, 2003; Soboroff, 2004) was discussed briefly in Section 2.4.2. The track focused on a sentence-level retrieval task in which, ideally, each sentence was returned only if it was relevant and contained some novel information given the sentences that preceded it.

Participants in the novelty track proposed a wide variety of techniques for modeling novelty, ranging from simple term-based counting metrics, through traditional metrics such as cosine similarity, to custom-designed metrics based on statistical language models and hidden Markov models. While some of these techniques seemed to work reasonably well, the novelty track brought to light serious concerns regarding the underlying task, both with the difficulty of the task and with the evaluation process.

One fact that became very clear during the years in which the novelty track operated was that accurately detecting relevant, novel sentences is far from easy. Semantic concepts such as novelty — or, indeed, relevance — are notoriously difficult to reliably detect with automatic means. This is reflected in the low recall and precision scores prevalent in most areas of information retrieval; high accuracy is not currently achievable. The problem is that, while inaccuracy is acceptable for relevance, it is less so for novelty. Because systems in the novelty track aim to highlight only novel information, all other information — that which is determined to be redundant — is effectively discarded from the search result. Thus, misclassification of novel information as redundant could lead to critical information being overlooked. It is therefore not clear whether the current level of performance at the novelty task actually improves the user experience or in fact degrades it.

A major problem that has been faced in the novelty task is that effectiveness evaluation has proved extremely difficult. The fact that the novelty of a sentence depends on every sentence that precedes it means that the task has had to be evaluated on a small, predefined set of documents presented in a specific order. Furthermore, in order to have a larger pool of relevant sentences on which to base the assessments, all documents in the list were strongly relevant to the query. However, this practice may have introduced a bias in the results. Allan et al. (2003) suggest that this practice of using only relevant documents could mean that existing results do not predict performance in more realistic search environments.

The novelty track also recorded a high level of disagreement between human judges regarding the novelty status of sentences. This suggests that even at the fundamental human level novelty is a vague notion. Thus, computers cannot be expected to achieve an acceptable level of accuracy at an inherently difficult and loosely-defined task.

For the reasons outlined above, the general novelty task is currently intractable both in terms of accurate detection of novelty and in terms of evaluation of system effectiveness. It is apparent that it will be quite some time before there is a general technique for identifying redundant documents that has satisfactory precision. New evaluation models that allow for redundancy to be taken into account in a reusable and cost-effective manner seem similarly remote. In the meantime, it is important that we do what is possible to manage the serious problem of redundancy in search-engine results. It is in this spirit that we define and investigate a restricted definition of redundancy that is largely free of such concerns.

5.2 Content Equivalence

We refer to a pair of documents as being *content-equivalent* if they convey the same information as each other. A functional interpretation of this relation is that an information consumer, having viewed one document, would gain no new information by viewing the other. Content-equivalent documents are therefore informationally redundant with respect to each other.

Content equivalence is a semantic notion; however, one would expect a strong correlation between document-pairs that are content-equivalent and those that are considered to be duplicates or near-duplicates. As such, we hypothesize that, despite being a semantic concept, content equivalence can be effectively detected using syntactic approaches, in particular document fingerprinting.

It is obvious that two identical documents are content-equivalent. However, documents that are not bytewise identical can still convey precisely the same informational payload. A common scenario in the TREC `.gov` data is the presence of two identical documents, one in HTML and the other converted to plaintext from PDF. These documents are identical on a word-by-word level but vary significantly in the way they are stored. In other cases, documents vary in their formatting and hyphenation. Another kind of case is where a document is updated without affecting most of the text, such as alternative forms of the same press release or different versions of the same policy document.

Duplication or near-duplication of documents has been noted as a special-case or extreme-case example of document redundancy (Zhai et al., 2003; Zhang et al., 2002). We show through the case of content equivalence that document duplication and near-duplication is neither special nor extreme; rather, it is widespread and has a significant impact on search results.

Note that we are also advocating a change in emphasis from that of the TREC novelty track. Rather than attempting to identify and promote cases of novelty — which can be difficult to establish and damaging if inaccurate — we argue for the more conservative approach of identifying obviously redundant documents and eliminating them or presenting them as a single list item. While with perfect detection accuracy the two ultimately amount to the same thing, the difference in emphasis is significant in its practical outcome because it encourages a more conservative approach to the management of redundancy.

5.3 Classification of equivalence

We apply the DECO document fingerprinting system described in Chapter 4 to identify document-pairs that are content-equivalent. Just as we used a scoring threshold in Chapter 4 to determine whether a document-pair was or was not co-derived, we now use a threshold to determine the content equivalence status of a document pair. As discussed in Section 4.6.2, the results of our experiments on the effectiveness of the DECO system suggested that there was nothing in the characteristics or the performance of the S_2 and S_3 scoring functions that allowed a meaningful distinction to be made between them. Both functions were simple, cheap to compute, and gave excellent performance. Given this absence of a discriminative criterion, we have chosen largely arbitrarily to make use of the S_3 scoring function.

In order to set a threshold that allows us a good combination of accuracy and sensitivity, we conducted a user study to study the nature of the correlation between the S_3 score of a document-pair and the likelihood that the pair is indeed content-equivalent. The user study is described in this section; we find that, as well as giving information on the correlation between content equivalence and S_3 score, the user study provided us with valuable additional insight into the nature of content equivalence in web document collections. Based on this insight, we define and discuss a modified form of content equivalence that we call *conditional equivalence.*

Further user experiments are then conducted to select and justify a suitable S_3 score threshold for reliably differentiating between document-pairs that are content-equivalent and those that are not.

5.3.1 Initial experiments

In our initial user study we required participants to classify document-pairs, presented side-by-side in a browser window, into one of the following four categories:

Level 3. The documents have **identical** content; the documents have no visible differences.

Level 2. The documents have **completely equivalent** content; any differences between the documents are trivial and do not differentiate them with respect to any reasonable query.

Level 1. The documents have **nearly equivalent** content; any differentiation between the documents is minor with respect to any reasonable query.

Level 0. The documents are **not equivalent**; differences between the documents are significant enough to differentiate them with respect to reasonable queries.

We avoid defining the term 'reasonable' too precisely. A reasonable query is a best-effort attempt at expressing an information need that one might plausibly have. A reasonable search engine makes sensible use of available information in order to make a best-effort attempt at answering the information need expressed in a given query.

The QRELS collection (described more fully in Section 5.7.1) consists of all documents that were judged for at least one of the topics used for the 2004 TREC terabyte track. It is our hypothesis that these documents are on the whole of a higher quality than the collection-wide average. DECO was run on this collection using the S_3 scoring function and with the *modulo* selection heuristic disabled. From the relationship graph generated by DECO, a pool of 420 document-pairs was randomly sampled such that their S_3 scores were evenly distributed between 0.4 and 1.0[1]. Document-pairs from this pool were presented to participants in a random order. The S_3 scores of document-pairs being assessed were not disclosed to the participants.

A group of four people participated in the pilot study, each providing assessments for 105 document-pairs from the pool. The results, particularly for levels 2 and 3, were not sufficiently robust to be truly useful in an operational setting. Failure analysis revealed that there were many pairs for which content was equivalent in all but very particular circumstances. For example, two documents may have had the same body but different navigational links. Although the navigational links would not differentiate the documents with respect to most queries, some queries may directly address these navigational links. In such cases the documents would not be equivalent.

We made the observation that, if we partitioned many of these document pairs into those parts that were content equivalent and those that were not, the partitions would be topically disjoint. In the above example of a document-pair with the same body text but different navigational links, the links address a very distinct topic — the structure of the website — from the topic which is addressed by the text in the body of the page. Thus, it is not possible to formulate a coherent query for which both the partitions are relevant to the information need.

[1]In order to make the best use of our assessors we decided to classify all pairs with lower scores as not being content-equivalent based on prior inspection.

5.3.2 Conditional equivalence

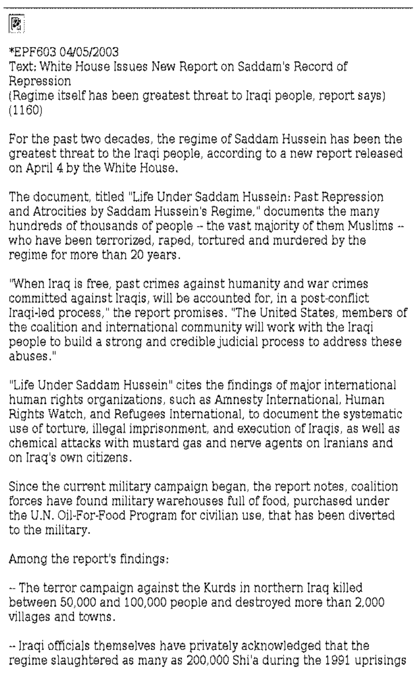
The insight gained from the pilot study suggested the concept of document-pairs possessing a *conditional equivalence*. Conditionally equivalent pairs of documents are those for which the space of queries can be partitioned into two categories: those in which the two documents are equivalent with respect to the information need, and those in which only one of the documents would be returned by a reasonable search engine. In other words, there are no queries for which both documents may reasonably be considered relevant to the query and yet at the same time not be considered equivalent. Another interpretation can be derived from the MMR model discussed in Section 5.1.2: a document pair is conditionally equivalent if both documents have a marginal relevance of zero with respect to the other in all cases, either because it is not relevant to the query or because it is no longer novel. An example of a pair of documents that can be considered to be conditionally equivalent in presented in Figure 5.2. The two documents conatin the same body text, but one has a navigational frame while the other does not. Any query for which both documents may rank highly will naturally be addressing the body of the text, and thus the navigational matter will not be relevant in such a case.

The significance of conditional equivalence is that within a given result list, documents of this class can be considered content equivalent. The presence of both documents in the result list implies they are equivalent for that query. Thus, knowledge of conditional equivalence relations within a collection can be used for postprocessing of ranked lists to remove or cluster documents from the list that are conditionally equivalent with documents that are higher up the list.

It may be argued that such a definition of equivalence is contrived and designed primarily to salvage our experimental results. However, it is our belief that in this context all definitions must ultimately be functionally driven. We assert that the definitions of content equivalence and conditional content equivalence are suitable for the domain of web search, and have presented arguments and use-cases above demonstrating that this is the case. In different situations, or under other sets of assumptions, alternative definitions may be more suitable.

5.3.3 Further experiments and threshold choice

For the main study, a group of 12 participants assessed a total of 964 document pairs for equivalence using the above criteria. We retained the same methodology as in the pilot study, but changed the classification categories based on our concept of conditional equivalence



*EPF603 04/05/2003
Text: White House Issues New Report on Saddam's Record of Repression
(Regime itself has been greatest threat to Iraqi people, report says)
(1160)

For the past two decades, the regime of Saddam Hussein has been the greatest threat to the Iraqi people, according to a new report released on April 4 by the White House.

The document, titled "Life Under Saddam Hussein: Past Repression and Atrocities by Saddam Hussein's Regime," documents the many hundreds of thousands of people -- the vast majority of them Muslims -- who have been terrorized, raped, tortured and murdered by the regime for more than 20 years.

"When Iraq is free, past crimes against humanity and war crimes committed against Iraqis, will be accounted for, in a post-conflict Iraqi-led process," the report promises. "The United States, members of the coalition and international community will work with the Iraqi people to build a strong and credible judicial process to address these abuses."

"Life Under Saddam Hussein" cites the findings of major international human rights organizations, such as Amnesty International, Human Rights Watch, and Refugees International, to document the systematic use of torture, illegal imprisonment, and execution of Iraqis, as well as chemical attacks with mustard gas and nerve agents on Iranians and on Iraq's own citizens.

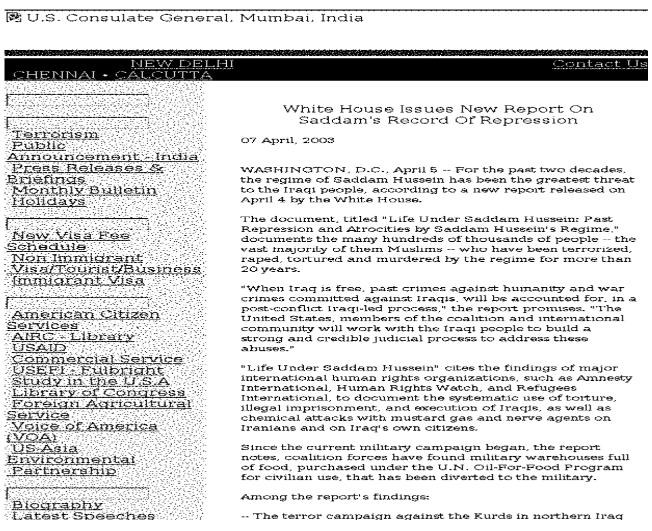
Since the current military campaign began, the report notes, coalition forces have found military warehouses full of food, purchased under the U.N. Oil-For-Food Program for civilian use, that has been diverted to the military.

Among the report's findings:

-- The terror campaign against the Kurds in northern Iraq killed between 50,000 and 100,000 people and destroyed more than 2,000 villages and towns.

-- Iraqi officials themselves have privately acknowledged that the regime slaughtered as many as 200,000 Shi'a during the 1991 uprisings

(a)



U.S. Consulate General, Mumbai, India

White House Issues New Report On Saddam's Record Of Repression

07 April, 2003

WASHINGTON, D.C., April 5 -- For the past two decades, the regime of Saddam Hussein has been the greatest threat to the Iraqi people, according to a new report released on April 4 by the White House.

The document, titled "Life Under Saddam Hussein: Past Repression and Atrocities by Saddam Hussein's Regime," documents the many hundreds of thousands of people -- the vast majority of them Muslims -- who have been terrorized, raped, tortured and murdered by the regime for more than 20 years.

"When Iraq is free, past crimes against humanity and war crimes committed against Iraqis, will be accounted for, in a post-conflict Iraqi-led process," the report promises. "The United States, members of the coalition and international community will work with the Iraqi people to build a strong and credible judicial process to address these abuses."

"Life Under Saddam Hussein" cites the findings of major international human rights organizations, such as Amnesty International, Human Rights Watch, and Refugees International, to document the systematic use of torture, illegal imprisonment, and execution of Iraqis, as well as chemical attacks with mustard gas and nerve agents on Iranians and on Iraq's own citizens.

Since the current military campaign began, the report notes, coalition forces have found military warehouses full of food, purchased under the U.N. Oil-For-Food Program for civilian use, that has been diverted to the military.

Among the report's findings:

-- The terror campaign against the Kurds in northern Iraq

(b)

Figure 5.2: Two documents that are conditionally equivalent: document (b) contains the same text as document (a), as well as containing a navigational frame. On a query for which both documents are likely to be returned, the relevant information will be equivalent.

discussed above. Participants in the main study were requested to classify document pairs into one of the categories below:

Level 3. The documents have **completely equivalent** content; any differences between the documents are trivial and do not differentiate them with respect to any reasonable query.

Level 2. The two documents are **conditionally equivalent**; with respect to any query for which both documents may be returned by a reasonable search engine, the documents are equivalent. Any query for which the documents are not equivalent would only return one or other of the documents.

Level 1. The documents have **nearly equivalent** content with respect to any query for which both documents may be returned by a reasonable search engine; for those queries where the documents are differentiated, the differentiation is minor.

Level 0. The documents are **not equivalent**; differences between the documents are significant enough to differentiate them with respect to reasonable queries.

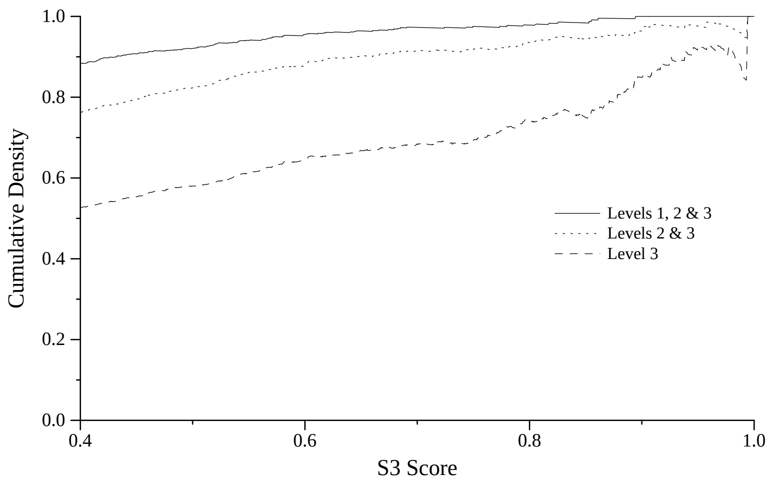


Figure 5.3: The proportion of document pairs judged to exceed various equivalence levels, for S_3 threshold values from 0.4 to 1.

Note that we no longer asked participants to differentiate between identical and content-equivalent documents as we did in the pilot study. This distinction was concluded to be unnecessary in a context where we are concerned entirely with the information content of the documents.

Results from the study were collated and analysed. Figure 5.3 shows the classification precision as the S_3 threshold increases; that is, the values at a particular S_3 value indicate the level of accuracy for all document pairs at and above that score. The graph indicates a reasonably strong linear relationship between the S_3 score for a document and the likelihood that it is content-equivalent. Note the relatively low accuracy for level 3 document pairs. This result means that many document pairs identified as content-equivalent by DECO do have some point of difference (albeit a very small one in many cases). This suggests that it may not be prudent to purge documents from a collection at index-time, as there may be particular situations in which a given document may be needed despite its near-equivalence to other documents. However, the S_3 score shows high levels of accuracy for level 1 and level 2 document pairs.

Given the lack of an obvious step-point at which to place a threshold, we has to make a somewhat arbitrary choice. We determined to choose a threshold such that 95% of assessed document pairs with an S_3 score exceeding the threshold had been rated as equivalent at level 1 or above. On the data collected, the threshold at which level 1 accuracy first exceeds 95% is 0.58. We thus use this threshold value of 0.58 to signify conditional equivalence in our further experiments this chapter. The proportion of documents at this threshold that were categorised at level 2 or above was 87%.

The choice of a classification threshold with a predicted accuracy of 95% must be borne in mind when interpreting results in later sections. When figures are reported for the number of document-pairs classified as being conditionally equivalent, one would expect that 5% of such classifications are erroneous using the current system.

Our results show that document fingerprinting using DECO and SPEX with the S_3 measure is able to accurately identify content equivalence between documents. While not failure-proof, the method appears robust. Given that for any pair of documents identified as being content-equivalent the one that is more highly ranked with respect to the query is returned to the user, and that the other document in the pair can be made available to the user on request, the incidence of information loss is likely to be low.

5.4 Retrieval equivalence

In addition to using the DECO document fingerprinting system to automatically identify content-equivalent documents, we also experiment with *retrieval equivalence*, an easy-to-compute restricted form of content equivalence motivated by the operation of search engines. A set of documents is retrieval-equivalent if once they are canonicalized — stripped of formatting and other information that is not considered during indexing — they are identical. This means that the documents will be indistinguishable to a search engine at retrieval time. Retrieval-equivalent document sets provide a very strong assurance of being content-equivalent, as their textual content is virtually identical. As the computation cost for retrieval equivalence is far lower than that of document fingerprinting, we use it as a baseline to test whether the additional effort of document fingerprinting is worthwhile.

The method for detecting retrieval equivalence follows trivially from the definition: documents in the collection are canonicalized and then hashed using the MD5 algorithm (Rivest, 1992). The list of hash values is sorted and all sets of documents sharing an identical MD5 hash are considered to be retrieval-equivalent. The asymptotic cost is thus $O(n \log n)$ for n documents but in practice the dominant cost is reading the documents, which exhibits $O(n)$ complexity.

We experimented with six levels of canonicalization, where each level adopts all the measures of previous levels. The six levels are as follows:

1. whitespace normalized

Collection	Year crawled	Size (GB)	# documents	# duplicates removed
GOV1	2002	18.1	1,247,753	267,370
GOV2	2004	426.0	25,205,179	2,950,950

Table 5.1: Statistics for the GOV1 and GOV2 collections.

2. tags removed
3. punctuation removed
4. case folded
5. stopwords removed
6. words stemmed

Note that the possibility of hash collisions is negligible: MD5 is a 128-bit hash, meaning that the space of possible values is 2^{128}. On a collection of one billion documents the likelihood of a single collision is about 10^{-22}.

5.5 Document collections

The following sections report on a series of experiments arising from the presence of content-equivalent documents in web collections. We make use of three document collections for these experiments: GOV1, GOV2, and QRELS, which are collections of web documents from the `.gov` domain created for TREC. The GOV1 and GOV2 collections were previously described in Section 4.4.1. Further information regarding these collections is presented in Table 5.1. According to the TREC track information, exact duplicate documents have already been removed from these collections. The QRELS collection is a subset of GOV2 and is described later.

5.6 Equivalence in the .gov domain

A question that immediately arises in the context of redundancy in web search results concerns the level of duplication in web collections. Naturally, if the level of redundancy in the web itself is sufficiently low then we need not worry much about the issue of redundancy when searching it. If, however, we can demonstrate that there are many equivalent documents in web collections then the issue merits further investigation. In this section we attempt to quantify the level of redundancy in two collections of web data.

We use DECO to determine both content equivalence and retrieval equivalence for the GOV1 collection. As previously discussed in Section 4.5.2, we are not currently able to process the GOV2 collection using SPEX on our current hardware. As identifying content equivalence requires knowledge of S_3 scores derived from the fingerprinting process, we were only able to identify retrieval-equivalence for this collection.

It is important to be careful when interpreting the results of these experiments. We are attempting in this section to quantify the degree of content equivalence one may expect to find on the web. The difficulty is that the web is, in a sense, subjective. For various technical reasons, the constitution and shape of the web is dependent to a large degree on the entity observing it. Whether a document occurs once or thousands of times is often a matter of interpretation. Thus, the influence of the crawler — the agent responsible for compiling web collections — must not be neglected. If a different crawler were to be used, different results may be observed. Given that they have been compiled by state-of-the-art crawlers, however, GOV1 and GOV2 can be considered representative of the sort of web snapshots with which modern search and indexing systems must deal.

5.6.1 Equivalence in GOV1

We explored the level of retrieval equivalence in the GOV1 collection under each of the six degrees of canonicalization defined in Section 5.4. We found that when only whitespace and HTML tags were removed, there were 21,840 sets of retrieval-equivalent documents for a total of 97,048 documents. When all further transformations were applied, this increased to 22,870 sets for a total of 99,227 documents. In other words, most of the documents composing the clusters became identical as a result of having HTML tags removed; few additional documents were identified as a result of further transformations. In light of this outcome, all further experiments applied the highest level of canonicalization, as the precise degree of canonicalization does not have a significant impact on the final result. In general, documents in these sets can be considered completely content-equivalent (level 3) and in many cases it is reasonable to eliminate all but one of these documents from the collection prior to indexing.

It is interesting to note that despite the removal of exact duplicates in the construction of the GOV1 collection (as shown in Table 5.1), we have been able to cluster a large number of effectively identical documents. This demonstrates the limitation of check-summing or other exact hashing techniques for redundancy management. Non-content elements such as tags,

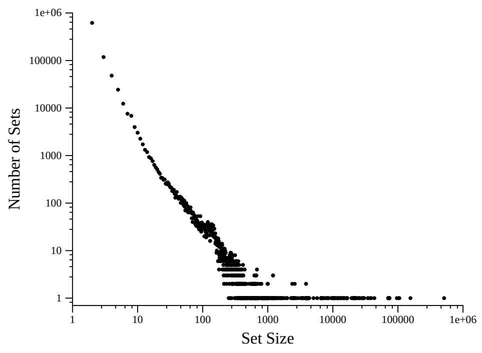


Figure 5.4: Distribution of retrieval-equivalent set sizes in the GOV2 collection, on log-log axes.

scripts, or filenames can all render entirely redundant documents non-identical at the binary level.

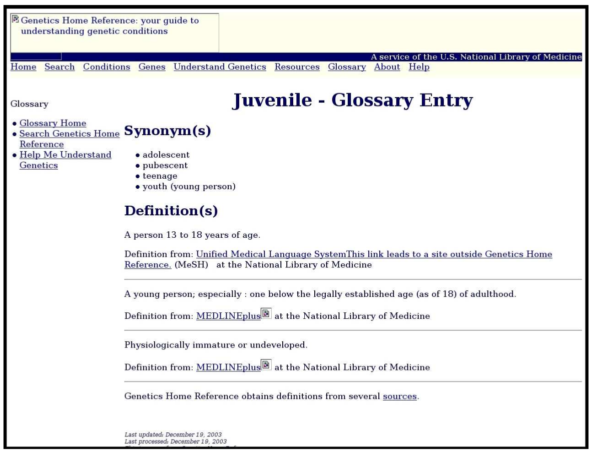
We then use fingerprinting to try to find further content-equivalent documents in this collection. As discussed in Section 5.3.3, we have chosen based on user experiments to use the S_3 scoring function with a threshold of 0.58. In these experiments, the chunk length has remained at eight words, and the *modulo* selection function was not used.

When we use fingerprinting on the collection, we find a total of 215,314 documents that are participating in a conditional equivalence relationship, or 116,087 additional documents compared to retrieval equivalence only. In total this means that 17.3% of documents were non-unique within this relatively small collection, already a relatively high figure. As collections grow larger and more comprehensive, one would expect this proportion to grow.

As discussed in Section 5.3.3, it is not advisable to prune documents from the collection being indexed based on conditional equivalence, as different documents from a conditionally equivalent set may better serve different sets of queries. The recommended action is to keep a record of conditional equivalence relationships and use this information to postprocess ranked lists. If a document in a ranked list is conditionally equivalent to a document that appears higher in the list then it may be removed from the list, reducing the overall redundancy in the final ranked list returned to the user.

5.6.2 Retrieval equivalence in GOV2

Our analysis of retrieval equivalence in the GOV2 corpus discovered a total of 6,943,000 documents in 865,362 separate retrieval-equivalent clusters. Thus, a total of 6,077,638 documents in the collection are entirely redundant for retrieval purposes, in addition to the 2,950,950



Genetics Home Reference: your guide to understanding genetic conditions

A service of the U.S. National Library of Medicine

Home Search Conditions Genes Understand Genetics Resources Glossary About Help

Glossary

- Glossary Home
- Search Genetics Home Reference
- Help Me Understand Genetics

Juvenile - Glossary Entry

Synonym(s)

- adolescent
- pubescent
- teenage
- youth (young person)

Definition(s)

A person 13 to 18 years of age.

Definition from: Unified Medical Language SystemThis link leads to a site outside Genetics Home Reference. (MeSH) at the National Library of Medicine

A young person; especially : one below the legally established age (as of 18) of adulthood.

Definition from: MEDLINEplus at the National Library of Medicine

Physiologically immature or undeveloped.

Definition from: MEDLINEplus at the National Library of Medicine

Genetics Home Reference obtains definitions from several sources.

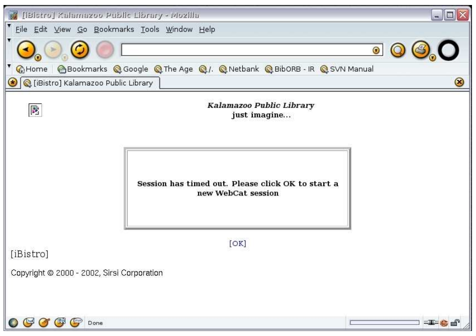
Last updated: December 19, 2003
Last processed: December 19, 2003

Figure 5.5: A document that appears 300 times in the GOV2 collection.

duplicate documents that had already been removed at crawl time. These 6,077,638 documents represent nearly 25% of the documents in the official corpus, which is consistent with previous investigations of text duplication in web crawls (Broder et al., 1997; Fetterly et al., 2003), which reported figures in the range of 25%–30%. Based on our results for GOV1, we speculate that further large numbers of document-pairs will be classified as being conditionally equivalent using document fingerprinting.

The results on GOV2 show several extremely large sets of documents that are retrieval-equivalent. The largest set encompasses 512,030 documents — 2% of all the documents in the GOV2 collection. Figure 5.4 shows the frequency of occurrence of sets of various sizes. The linear character of the distribution on double-log axes is indicative a power-law distribution. This is consistent with the results of Fetterly et al. (2003), with the graph showing a high degree of similarity to graphs constructed from a crawl of 150 million documents from the general web. There is even a similar curious artefact at set sizes of about 100, which Fetterly et al. attribute to mirroring. There were many large sets that contained reasonably information-rich documents. An example of a document that occurred 300 times is shown in Figure 5.5.

We wished to investigate whether the extremely large clusters of retrieval-equivalent documents were the result of some anomaly within the SPEX system, null documents — documents without any non-tag content — or genuine duplicates. We selected ten documents at random from each of the ten largest clusters and examined them manually in order to determine what type of documents these clusters comprised. In all cases, the ten randomly


[iBistro] Kalamazoo Public Library - Mozilla
File Edit View Go Bookmarks Tools Window Help
Home Bookmarks Google The Age /. Netbank BibORB - IR SVN Manual
[iBistro] Kalamazoo Public Library
Done

Kalamazoo Public Library
just imagine...

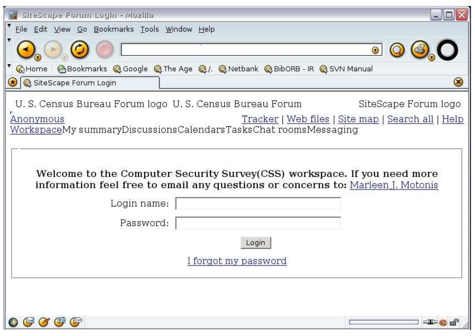
Session has timed out. Please click OK to start a new WebCat session

[OK]

[iBistro]

Copyright © 2000 - 2002, Sirsi Corporation

(a)


SiteScape Forum Login - Mozilla
File Edit View Go Bookmarks Tools Window Help
Home Bookmarks Google The Age /. Netbank BibORB - IR SVN Manual
SiteScape Forum Login
Login

U. S. Census Bureau Forum logo U. S. Census Bureau Forum SiteScape Forum logo

Anonymous Tracker | Web files | Site map | Search all | Help

WorkspaceMy summaryDiscussionsCalendarsTasksChat roomsMessaging

Welcome to the Computer Security Survey(CSS) workspace. If you need more information feel free to email any questions or concerns to: Marleen J. Motonis

Login name:

Password:

I forgot my password

(b)

Figure 5.6: Representative documents from the two largest clusters in the GOV2 collection: (a) 512,030 documents (b) 155,684 documents.

selected documents were identical, validating that the clusters were not in any way anomalous but rather were genuine content-equivalent documents. Figure 5.6 shows examples of the documents composing the two largest clusters, which were of 512,030 and 155,684 documents in size respectively.

5.7 Equivalence in search results

Not all documents are created equal; some documents contain much more information than others, or information of a higher quality, or information that is pertinent to a wider variety of queries. Document access patterns are extremely skewed, such that some documents are accessed extremely frequently and others rarely or not at all (Garcia et al., 2004). For example, the documents that appeared in the largest sets discussed in Section 5.6.2 would be relevant to few queries and as such accessed rarely.

Thus, in order to truly assess the impact of content equivalence on the search experience, we must examine its prevalence amongst the types of documents that are actually returned in response to typical queries. Particularly, we are interested in the level of content equivalence amongst documents that are judged *relevant*, as significant redundancy amongst such documents could contribute to overestimates of system effectiveness.

In order to make the above investigations, we made use of the relevance pool of the 2004 TREC terabyte track (see Section 2.4.1). We also use the 70 official runs submitted to the TREC 2004 terabyte track to estimate the impact that redundant documents are having on the quality of real search results.

5.7.1 The QRELS collection

The primary task in the TREC 2004 terabyte track (for which the GOV2 collection was compiled) was a standard *ad hoc* search. Participants were given 50 queries; a run consisted of a ranked list of 10,000 GOV2 documents for each of the queries. The top 85 documents from each ranked list submitted were added to a judgement pools (see Section 2.4.1) for the 50 queries (TREC topics 701–750). These documents were assessed by human judges.

We compiled all the documents from the 2004 judgement pools to create a subcollection of the GOV2 collection that we call the QRELS collection. It consists of 58,078 documents and is 2.8 GB, or nearly one-sixth of the total size of GOV1. Note that the QRELS collection does not contain only relevant documents; rather, it contains both relevant and non-relevant documents for each of the queries, as judged by the assigned judges.

The documents in the QRELS collection were retrieved by real search systems in response to the carefully formulated TREC topics and as such have demonstrated that they are likely to be documents of high worth, retrieved frequently. The signal-to-noise ratio is expected to be much higher in the QRELS collection than amongst the GOV2 collection as a whole. We note that the average size of a document in QRELS is 49.3 KB, or 2.8 times larger than the collection average of 17.7 KB.

5.7.2 Equivalence in the QRELS collection

A total of 23.9% of all documents in the QRELS collection were found to exhibit content equivalence with at least one other document, but for some queries the figure was far higher. Only a minority of the content-equivalent documents are retrieval-equivalent. This shows that content equivalence has a real effect on search results. Compared to the GOV1 collection, the ratio between retrieval-equivalent and conditionally equivalent documents is far higher. We hypothesize that this is because the extremely large retrieval-equivalent clusters inflate the degree of retrieval equivalence in the collection. Documents in such clusters are likely to be machine-generated and of low value with respect to most common queries.

Figure 5.7 shows the degree of document redundancy — the proportion of documents that can be eliminated for a given query because they can be represented by an equivalent document — on a per-topic basis. The lower histogram shows the percentage of all relevant documents for each topic that are redundant, while the upper histogram displays the same data for irrelevant documents. Topic 703 was not included in the final TREC judgements, accounting for the absence of results for that query. Taken across all queries, the mean

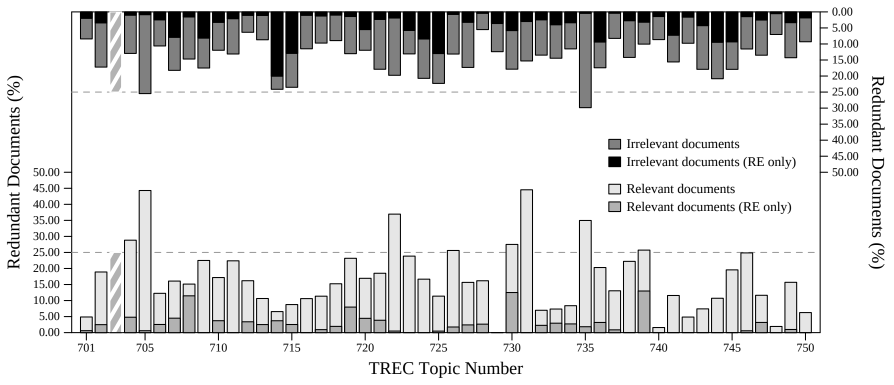


Figure 5.7: The proportion of redundant documents for each topic amongst the relevant (lower graph) and nonrelevant (upper graph) judged pools. Note that query 703 is absent from the TREC judgements.

percentage of redundant documents amongst the relevant set was 16.6%, while the mean amongst the irrelevant set was 14.5%. If we consider only retrieval-equivalent documents, mean redundancy amongst the relevant and irrelevant sets is 2.3% and 4.0% respectively. Note that these percentages exclude the first document in each cluster; only subsequent, and hence redundant, documents are counted.

The very large difference between the proportion of redundant relevant documents detected by the retrieval equivalence and the fingerprinting methods suggests that many of these document pairs do have some point of difference, in contrast to the large clusters of low-value documents captured by the retrieval equivalence technique. This outcome serves to emphasize the importance of using a sensitive technique like fingerprinting even more emphatically than the results in Section 5.6.1.

We note that the observed incidence of content equivalence in the QRELS collection is lower than in the GOV2 collection as a whole. This is not surprising, as much of the content equivalence in the GOV2 collection occurred on low-value documents, such as the server-generated error pages and search forms that occurred in the large sets discussed in Section 5.6.2. Despite this, redundant documents still represent a significant proportion of all documents in both the relevant and nonrelevant judged sets of the QRELS collection. The fact that, on average, 16.6% of documents that have been judged to be relevant are redundant means that the user will in many cases be viewing relevant documents that they have seen before. These documents, though relevant to the topic, are no longer useful to the user as they are entirely lacking in novelty. This suggests that effectiveness figures calculated for the terabyte track runs could be substantial overestimates of the user experience, and that the user experience could be significantly improved by removing instances of redundant documents from result lists.

5.7.3 Equivalence and search performance

We have claimed that document redundancy has a negative effect on the effectiveness of search. In this section we attempt to quantify the extent of this effect by examining the level of inter-document redundancy occurring in the official runs submitted to the 2004 TREC terabyte track.

We reiterate the novelty principle discussed earlier: a document, though relevant in isolation, should not be regarded as relevant if it is equivalent to a document the user has already seen. At the very least, a system should not be rewarded for presenting multiple copies of the same document, even if it is relevant. In order to model this principle, we modified the official judgements for each run so that documents appearing in a result list after another document with which they were content-equivalent were marked as irrelevant, regardless of their judged relevance status.

In order to discount the effect of poorly performing runs we adopted the methodology used by Voorhees and Buckley (2002) and Sanderson and Zobel (2005) and discarded the bottom 25% of runs, leaving us in this case with 54 runs. We used the `trec_eval` tool for evaluation and recorded the MAP for each of the runs.

Evaluating the TREC runs as submitted resulted in an average MAP across the runs of 0.201. When the novelty principle was modelled as described above, the average MAP fell to 0.161, a relative reduction in MAP of 20.2%. This is a substantial difference, and demonstrates that the assumption of independent relevance is inflating effectiveness scores, and that redundant documents almost certainly have a significant impact on the user experience of search. In a scenario where we are exploring the effect of eliminating redundant documents, it is clear that 0.161, not 0.201, is the correct baseline for comparison.

To simulate an information retrieval system that is aware of content-equivalent documents, we modified the runs so that documents appearing after another document with which they were content-equivalent were removed from the result list. The average MAP increased to 0.186, a relative 16.0% improvement compared to our baseline, demonstrating that an equivalence-aware retrieval system should be able to substantially improve the user's search experience.

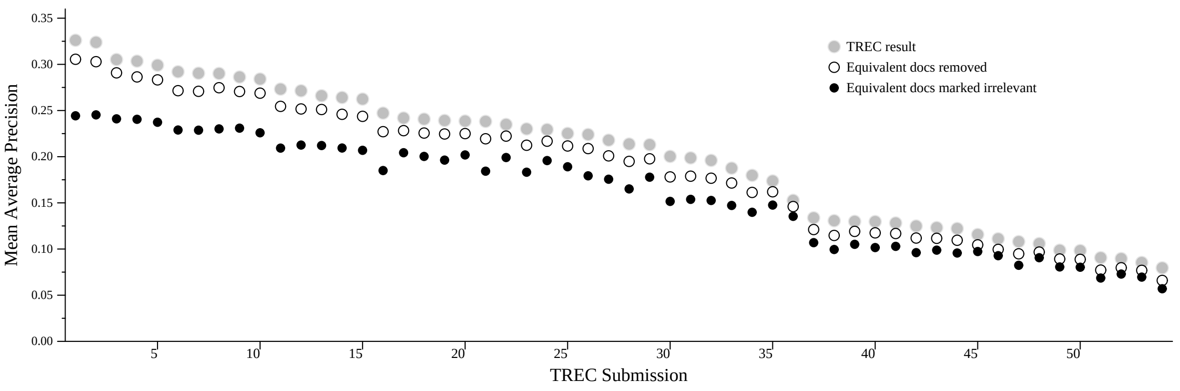


Figure 5.8: MAP on TREC queries 701-750 for the 54 most successful runs submitted to the 2004 terabyte track. The first column is the official MAP result, the second column shows MAP if subsequent content-equivalent documents are marked nonrelevant, and the third column shows MAP with the same documents removed from the run.

That the improvement is, by construction, observed in every query with redundant answers and that the degree of improvement is closely coupled to the degree of redundancy observed in the QRELS collection does nothing to invalidate this result. We note that redundant answers can add no weight to a user's confidence in information. We also note that this improvement is obtained on the same underlying collection and on the same set of redundancy-modelled relevance assessments.

These results are presented in graphical form for each of the individual runs in Figure 5.8. In all cases the official MAP result overstates the effectiveness of the run. Interestingly, both the degradation in MAP from rendering redundant documents irrelevant and the improvement brought by removing duplicates was greater for runs that showed overall better effectiveness. We are unable to account for this phenomenon with a convincing explanation.

5.8 Inconsistency in relevance judgements

Equivalent documents should not vary with respect to their relevance; thus, either the pairs were incorrectly classified as content equivalent or one of the documents in the pair was erroneously judged. This provides a convenient opportunity to assess the overall consistency of relevance assessments for the TREC 2004 terabyte track.

All documents that were connected by content-equivalence relationships were aggregated into a single group. Any group in which not all documents received the same relevance judgements was identified as containing an inconsistency. We manually examined 20 randomly selected groups that had been inconsistently judged and found that in all cases the judgements ought not to have disagreed.

In total across all queries there were 465 groups where judgements were inconsistent. Within these groups, 791 documents were judged relevant and 681 were judged nonrelevant. Assuming for the sake of argument that the majority of judgements in each group were correct, on average across all groups 3.8% of the judgements (522 of 13,854) were incorrect. If we consider only the set of *potentially relevant* documents — that is, the set of all documents that are either relevant or form part of an inconsistent group — we have 522 erroneous judgements out of 4,013 documents, an error rate of 13.0%.

This evidence seems incontrovertible: content-equivalent documents have been frequently classified by the same judge as being both relevant and irrelevant. We have no evidence of assessment quality for the documents that do not have a content equivalence relationship with any other documents, but there is no reason to suppose that the general degree of judgement accuracy for these other documents is any higher.

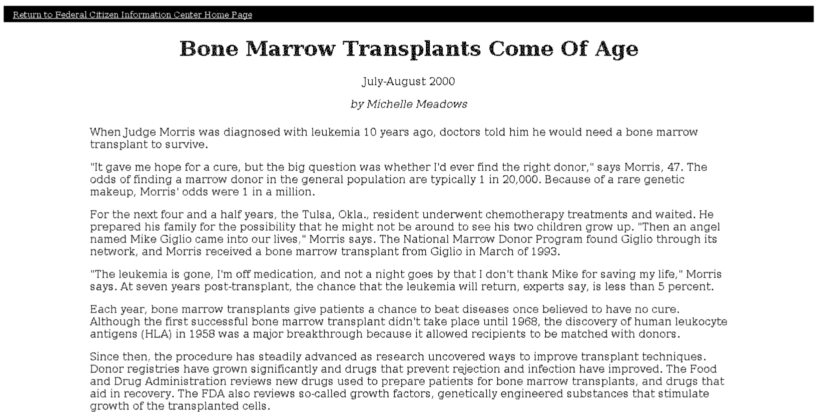
5.9 Conclusions

Inter-document redundancy in ranked lists is a significant problem for search engines, leading to a degradation in the user experience. However, the failure of common ranking or evaluation models to take redundancy into account in determining effectiveness has led to a distorted situation in which redundancy and duplication are implicitly encouraged. Current models under which redundancy is taken into account during the ranking and evaluation processes are too complex and error-prone for general use.

In this chapter we have defined and explored a restricted class of inter-document redundancy that we call content equivalence. We contend that the presence of content-equivalent documents in a result list is not in general of benefit to the user. Document fingerprinting was shown via a user study to be a robust method for identifying content-equivalence between documents. Using our method, we found that over 17% of documents in GOV1 were non-unique under content equivalence, more than double the figure of the stricter retrieval equivalence. Almost 25% of GOV2 was redundant under retrieval equivalence. Our analysis of judged documents and extrapolation from GOV1 suggests that as many documents again may be non-unique under content equivalence.

We also showed that content-equivalence has a significant impact on actual search results from a wide variety of search methodologies, with the failure of current evaluation methodologies to model redundancy meaning that the effectiveness of runs submitted to the 2004 TREC terabyte track was overestimated by at least 20%. The corollary to this is that purging content-equivalent documents from results lists ought to lead to an improvement in effectiveness. However, the failure of current evaluation metrics to take the negative effect of redundancy into account means that removing content equivalent documents from result lists actually lowers measured effectiveness. In order to negate this unbalanced situation, we introduced a variant of MAP in which redundant documents were considered to be irrelevant. A new set of experiments using this redundancy-aware evaluation metrics revealed that management of redundancy improved measured effectiveness by an average of 16.0% over the 54 best-performing runs submitted to the 2004 TREC terabyte track.

Disturbingly, our study exposes a significant degree of inconsistency in the human relevance judgements used for evaluating the performance of search algorithms in the TREC terabyte track.



Return to Federal Citizen Information Center Home Page

Bone Marrow Transplants Come Of Age

July-August 2000

by Michelle Meadows

When Judge Morris was diagnosed with leukemia 10 years ago, doctors told him he would need a bone marrow transplant to survive.

"It gave me hope for a cure, but the big question was whether I'd ever find the right donor," says Morris, 47. The odds of finding a marrow donor in the general population are typically 1 in 20,000. Because of a rare genetic makeup, Morris' odds were 1 in a million.

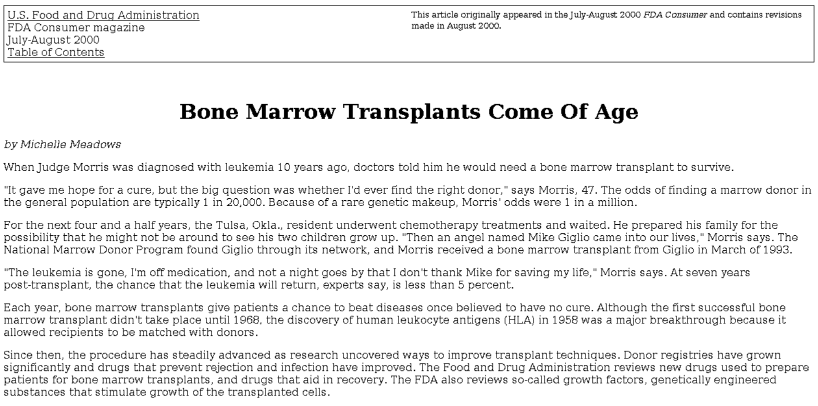
For the next four and a half years, the Tulsa, Okla., resident underwent chemotherapy treatments and waited. He prepared his family for the possibility that he might not be around to see his two children grow up. "Then an angel named Mike Giglio came into our lives," Morris says. The National Marrow Donor Program found Giglio through its network, and Morris received a bone marrow transplant from Giglio in March of 1993.

"The leukemia is gone, I'm off medication, and not a night goes by that I don't thank Mike for saving my life," Morris says. At seven years post-transplant, the chance that the leukemia will return, experts say, is less than 5 percent.

Each year, bone marrow transplants give patients a chance to beat diseases once believed to have no cure. Although the first successful bone marrow transplant didn't take place until 1968, the discovery of human leukocyte antigens (HLA) in 1958 was a major breakthrough because it allowed recipients to be matched with donors.

Since then, the procedure has steadily advanced as research uncovered ways to improve transplant techniques. Donor registries have grown significantly and drugs that prevent rejection and infection have improved. The Food and Drug Administration reviews new drugs used to prepare patients for bone marrow transplants, and drugs that aid in recovery. The FDA also reviews so-called growth factors, genetically engineered substances that stimulate growth of the transplanted cells.

GX013-60-6825144: Irrelevant



U.S. Food and Drug Administration
FDA Consumer magazine
July-August 2000
Table of Contents

This article originally appeared in the July-August 2000 *FDA Consumer* and contains revisions made in August 2000.

Bone Marrow Transplants Come Of Age

by Michelle Meadows

When Judge Morris was diagnosed with leukemia 10 years ago, doctors told him he would need a bone marrow transplant to survive.

"It gave me hope for a cure, but the big question was whether I'd ever find the right donor," says Morris, 47. The odds of finding a marrow donor in the general population are typically 1 in 20,000. Because of a rare genetic makeup, Morris' odds were 1 in a million.

For the next four and a half years, the Tulsa, Okla., resident underwent chemotherapy treatments and waited. He prepared his family for the possibility that he might not be around to see his two children grow up. "Then an angel named Mike Giglio came into our lives," Morris says. The National Marrow Donor Program found Giglio through its network, and Morris received a bone marrow transplant from Giglio in March of 1993.

"The leukemia is gone, I'm off medication, and not a night goes by that I don't thank Mike for saving my life," Morris says. At seven years post-transplant, the chance that the leukemia will return, experts say, is less than 5 percent.

Each year, bone marrow transplants give patients a chance to beat diseases once believed to have no cure. Although the first successful bone marrow transplant didn't take place until 1968, the discovery of human leukocyte antigens (HLA) in 1958 was a major breakthrough because it allowed recipients to be matched with donors.

Since then, the procedure has steadily advanced as research uncovered ways to improve transplant techniques. Donor registries have grown significantly and drugs that prevent rejection and infection have improved. The Food and Drug Administration reviews new drugs used to prepare patients for bone marrow transplants, and drugs that aid in recovery. The FDA also reviews so-called growth factors, genetically engineered substances that stimulate growth of the transplanted cells.

GX005-13-16310498: Relevant

Figure 5.9: An example of two content-equivalent documents that have been inconsistently judged in query 725 of the 2004 TREC terabyte track.

Chapter 6

Applications of fingerprinting in genomics

This chapter contains material that appeared in Cameron et al. (2006a), Bernstein and Cameron (2006), and Cameron et al. (2006b) and is based on research conducted in collaboration with fellow PhD candidate Michael Cameron. I adapted the DECO *package to process genomic sequence data and provided an implementation of the slotted* SPEX *approach. Michael Cameron implemented the clustering algorithm and adapted BLAST to search clusters defined by our method. We contributed equally to the invention of the slotted* SPEX *algorithm and our scheme for managing redundancy using union-sequences and wildcards, and to the experimental evaluations presented in this chapter. We both made contributions to the writing of the original papers, text from which appears in this chapter.*

As with collections of web documents, comprehensive genomic databases such as the GenBank non-redundant protein database (NR) contain — despite the name — extensive internal redundancy. Although exact duplicates are removed from the collection, there remain large numbers of near-identical sequences. This phenomenon has various causes, including the existence of closely-related homologues or partial sequences, sequences with expression tags, fusion proteins, and sequencing errors. These minor sequence variations lead to the over-representation in databases of certain protein domains, particularly those that are under intensive research. For example, the GenBank protein database contains several thousand near-identical protein sequences from the human immunodeficiency virus (HIV).

Inter-sequence redundancy in genomic databases has several negative consequences. As in text search, redundancy can lead to highly repetitive search results for any query that matches closely with an over-represented sequence, increasing the user burden in interpreting the results. Furthermore, sequence homology search algorithms such as BLAST are non-indexed, and hence orders of magnitude slower than indexed search. Larger databases lead to a noticeable increase in search time; as sequencing efforts continue to outpace improvements in computer processing speed, this effect will become increasingly problematic.

A problem unique to algorithms such as BLAST is that large-scale redundancy can have the effect of skewing the statistics used for determining alignment significance, ultimately leading to decreased search effectiveness. Profile-based distant homology search tools such as PSI-BLAST (see Section 2.8.3) can also be misled by redundant matches during iteration, causing them to bias the profile towards over-represented domains. This can result in a less sensitive search or even profile corruption (Li et al., 2002; Park et al., 2000).

Redundancy in genomic databases has in general been managed by the creation of representative-sequence databases (RSDBs), culled collections in which no two sequences share more than a given level of identity. RSDBs are built by grouping into clusters all sequences that exceed a given threshold for mutual redundancy, and then choosing a single representative from each of these clusters. This process requires that the degree of redundancy between every pair of sequences in the database be computed according to some appropriate metric such as alignment score.

Previous studies have investigated a range of approaches to identifying all pairs of highly similar sequences in a collection. Most of the proposed techniques execute significantly faster than a naïve application of query-based algorithms such as BLAST, which takes several days to process a protein database of 100 MB on a modern desktop computer. However, a majority of these algorithms have a fundamental $O(n^2)$ complexity in the size of the collection, rendering them increasingly impractical as genomic databases continue their rapid growth. Malde et al. (2003) have investigated the use of suffix structures (see Section 2.6.1) to efficiently identify high-identity pairs in a single pass of the collection. This approach does not suffer from the quadratic complexity problem; however, the significant resource consumption of these structures renders them unsuitable for large genomic collections.

In this chapter we describe how document fingerprinting can be used to reduce the complexity and execution time of the RSDB construction process. We discuss the challenges faced in adapting fingerprinting to genomic sequences, and propose a new selection heuristic, slotted SPEX, that is better suited to this domain. While slotted SPEX is a lossy algorithm, it offers a performance guarantee that is not generally available from other selection heuristics. We empirically verify that document fingerprinting using slotted SPEX is accurate and discriminative for identification of sequence pairs with a high degree of identity.

We find that, using document fingerprinting as a preprocessing step, we are able to build a RSDB from the GenBank non-redundant database in around 1.5 hours; the fastest existing approach, CD-HIT (Li et al., 2001a), requires over 9 hours for the same task. Furthermore, using fingerprinting we observe a flatter growth in execution times when compared to CD-HIT. Importantly, there is no significant change in the quality of the resultant RSDB.

Having demonstrated that fingerprinting can be used as a fast and accurate method for detecting redundancy in genomic sequence databases, we turn our attention to the way in which redundancy is managed. RSDBs produce compact databases leading to faster searches, and have been shown to significantly improve profile training in iterative search tools such as PSI-BLAST. However, they also have significant deficiencies. In particular, RSDBs do not represent a comprehensive record of all sequences originally present in the database. This leads to search results that are both less accurate, because the representative sequence for a cluster may not be the one that aligns best with a given query, and less authoritative, because the user is only shown one representative sequence from a family of similar sequences. In many cases, the effectiveness of searches conducted on RSDBs is lower than the effectiveness of the same search conducted on the source (redundant) database.

In Section 6.5 we describe a novel way of building and representing clusters of redundant sequences, and accompanying modifications to BLAST for searching databases stored using this representation. Importantly, our representation avoids the deficiencies of RSDBs. Instead of discarding near-duplicate sequences, we identify clusters of redundant sequences and construct a special *union-sequence* that represents all members of the cluster through the use of wildcard characters. When combined with a well-chosen set of wildcards and a system for scoring matches between wildcards and query residues, our approach leads to faster search times without a significant loss in accuracy. Moreover, by recording the differences between the union-sequence and each cluster member using edit information, our approach compresses the collection. Our scheme is generic and can be used to improve search times on most homology search tools.

We have integrated our new redundancy management strategy with our freely available open-source software package, FSA-BLAST. When applied to the GenBank NR database, our method reduces the size of sequence data in the database by 27% and improves search times by 22% with no significant effect on accuracy.

6.1 Existing approaches to redundancy management

In this section we examine existing approaches to the two problems of redundancy management in genomic sequence databases: first, how to identify sequences that are mutually redundant; and second, once redundant sequences have been identified, what to do about them.

6.1.1 Redundancy identification

The first stage of most redundancy management algorithms involves identifying pairs of highly similar sequences. An obvious approach to this task is to align each sequence in the collection with each other sequence using a pairwise alignment scheme such as Smith-Waterman local alignment (Smith and Waterman, 1981). This is the approach taken by several existing clustering algorithms, including d2_cluster (Burke et al., 1999), OWL (Bleasby and Wootton, 1990), KIND (Kallberg and Persson, 1999), and that of Itoh et al. (2004). However, this approach is impractical for collections of significant size. Each pairwise comparison is computationally intensive and the number of pairs grows quadratically in the number of sequences.

Several schemes, including CLEANUP (Grillo et al., 1996), NRDB90 (Holm and Sander, 1998), RSDB (Park et al., 2000), CD-HI (Li et al., 2001b) and CD-HIT (Li et al., 2001a), use a range of BLAST-like heuristics to quickly identify high-scoring pairwise matches. The CLEANUP (Grillo et al., 1996) algorithm builds a rich inverted index of short substrings or *words* in the collection and uses this structure to score similarity between sequence pairs. NRDB90 (Holm and Sander, 1998) and RSDB (Park et al., 2000) use in-memory hashtables of decapeptides and pentapeptides for fast identification of possible high-scoring sequence pairs before proceeding with an alignment. CD-HI (Li et al., 2001b) and CD-HIT (Li et al., 2001a) use lookup arrays of short subsequences to more efficiently identify similar sequences. Such methods can significantly reduce the per-pair comparison time, but do nothing to alter the $O(n^2)$ time complexity of the algorithms.

Many of the above schemes attempt to reduce the number of pairwise sequence comparisons by using a *greedy incremental clustering* approach, which interpolates the redundancy identification and RSDB construction stages. To begin, the collection sequences are sorted by decreasing order of length. Then, each sequence is considered in turn and used as a query to search an initially-empty representative database for high-scoring matches. If a similar sequence is found, the query sequence is discarded. Otherwise, it is added to the database as the representative of a new cluster. When the algorithm terminates, the database consists of the representative (longest) sequence of each cluster. While this technique can significantly reduce the number of pairwise sequence comparisons that need to be made, it still retains the unfavourable $O(n^2)$ time complexity that afflicts all direct pairwise-comparison methods. We show in Section 6.6 that CD-HIT — the fastest of the greedy incremental algorithms mentioned and the most successful existing approach — scales poorly, exhibiting superlinear time complexity in the size of the collection.

ICAass (Parsons, 1995) and the method of Itoh et al. (2004) reduce the number of pairwise comparisons by partitioning the collection according to phylogenetic classifications and clustering only sequences within each partition. This reduces the number of pairwise comparisons. However, the approach assumes that the database has been pre-classified and ignores possible matches between taxonomically distant species. Further, the number of phylogenetic divisions is growing at a far slower rate than database size. Therefore, a quadratic growth rate in computation time remains a limitation.

One way to avoid an all-against-all comparison is to pre-process the collection using an index or suffix structure that can be used to efficiently identify high-scoring candidate pairs. Malde et al. (2003) and Gracy and Argos (1998) investigated the use of suffix structures (see Section 2.6.1) to identify groupings of similar sequences in linear time. However, suffix structures also consume significant resources and are thus not an ideal choice for processing large sequence collections such as GenBank on desktop workstations. Malde et al. (2003) report results for only a few thousand EST sequences; the algorithm described by Gracy and Argos (1998) requires several days to process a collection of around 60,000 sequences. External suffix structures, which record information on disk, are also unsuitable: they use a large amount of disk space and are either extremely slow for searching, or have slow construction times (Cheung et al., 2005).

6.1.2 Redundancy management

The common practice for the management of inter-sequence redundancy in genomic sequence databases to build representative-sequence databases (RSDBs). In this approach, clusters of highly similar sequences are grouped into clusters. Once a set of clusters has been identified, one sequence from each cluster is selected as the *representative* of that cluster to be inserted into the RSDB; the rest are discarded (Holm and Sander, 1998; Park et al., 2000; Li et al., 2001b;a). In the case of greedy incremental clustering, the clusters are built incrementally during the pairwise similarity comparison. In either case, the result is a representative database with fewer sequences and less redundancy.

However, purging near-duplicate sequences can significantly reduce the quality of results returned by search tools such as BLAST. There is no guarantee that the representative sequence from a cluster is the sequence that best aligns with a given query. Therefore, some queries will fail to return matches against a cluster that contains sequences of interest, which reduces sensitivity. Further, results of a search lack authority because they do not show the best alignment from each cluster. Also, the existence of highly-similar alignments, even if strongly mutually redundant, may be of interest to a researcher.

Itoh et al. (2004) describe an alternative redundancy management technique. Their approach builds clusters of sequences and selects a cluster representative in a similar manner as for RSDBs. However, non-representative members of each cluster are retained in the database instead of being deleted. For each cluster, an upper bound is calculated on the difference in score between aligning a query to any sequence in the cluster and aligning it to the chosen representative. During search, the query is compared only to the representative sequence of each cluster. The upper bound is added to the resulting alignment score and if this increased score exceeds the scoring cutoff, all sequences in that cluster are individually aligned to the query.

While the above approach ensures there is no loss in sensitivity, it comes at a substantial cost: unless a high scoring threshold is used during search — Itoh et al. use a nominal score cutoff of 150 in their experiments — there will be numerous false positives, causing search to be slowed. They report experiments using Smith-Waterman (1981) alignment and it is unclear if their approach would work well if applied to a heuristic search tool such as BLAST. In addition, all sequences are retained by their method, so no space is saved.

6.2 Fingerprinting for genomic sequences

In this section we describe how document fingerprinting techniques can be adapted for redundancy identification in the domain of genomic sequences. By so doing, we can eliminate the need for exhaustive pairwise comparison of sequences in the database, and significantly improve the speed of the clustering process.

The SPEX algorithm (and, indeed, any fingerprinting algorithm) can be trivially adapted for use with genomic sequences by simply substituting an appropriate chunk parser. However, the properties of a genomic sequence are quite different from those of a natural language document. The most significant difference is the lack of any higher-order structure in genomic data — in particular, the absence of words. The protein sequences we consider in this paper are represented as an undifferentiated string of amino-acid characters with no natural delimiters such as whitespace, commas, or other punctuation marks.

The absence of any structure analogous to words in genomic sequences has several immediate impacts on the operation and performance of fingerprinting and the SPEX algorithm. First, the granularity of the sliding window must be increased from word-level to character-level. This will result in significantly more chunks being extracted from a genomic sequence than from a natural-language document of the same size. As a result, the SPEX algorithm must work harder to process genomic databases when compared to text collections.

Another domain-specific issue is that the distribution of subsequences within genomic data is less skew than the distribution of words in natural-language text. Given a collection of natural language documents, we expect some words (such as 'and' and 'or') to occur extremely frequently, while other words (such as perhaps 'alphamegamia' and 'nudiustertian') will be *hapax legomena*: words that occur only once. This permits the SPEX algorithm to be effectual from the first iteration by removing word pairs such as 'nudiustertian news'. In contrast, given a short string of characters using the amino acid alphabet of size 20, it is far less likely that the word will occur only once in any collection of nontrivial size. Thus, the first few iterations of SPEX are likely to be entirely ineffectual.

A simple solution to these problems is to introduce 'pseudo-words', effectively segmenting each sequence by moving the sliding window several characters at a time. However, this approach relies on sequences being aligned along segment boundaries, rendering the algorithm highly sensitive to insertions and deletions. Consider, for example, the following sequences with a chunk length of four and a window increment of four:

	Sequence	Chunks
Sequence 1	AB**CDEFGHIJKLM**NOP	ABCD EFGH IJKL MNOP
Sequence 2	AAB**CDEFGHIJKLM**NOP	AABC DEFG HIJK LMNO
Sequence 3	GHAA**CDEFGHIJKLM**Q	GHAA CDEF GHIJ KLMQ

Despite all three of these sequences containing an identical subsequence of length 11 (in bold above), they do not share a single common chunk. This strong correspondence between the three sequences will thus be overlooked by the algorithm.

We propose a hybrid of regular SPEX and the pseudo-word based approach described above, which we call slotted SPEX. Slotted SPEX uses a window increment greater than one

Algorithm 2 The slotted SPEX algorithm

```
chunkLength ⟵ finalLength - Q × (numIterations - 1)
for iteration ⟵ 1 to numIterations
   foreach sequence in the collection
      foreach chunk of length chunkLength in sequence
         if lookup[chunk] $\neq$ 0 then
            increment lookup[chunk]
         else
            count number of subchunks of length chunkLength - Q
                  where lookup[subchunk] = $2^+$
            if count $\geq$ 2 or iteration = 1 and
               number of chunks processed since increment lookup $\geq$ Q then
                  increment lookup[chunk]
   increment chunkLength by Q
```

but is able to 'synchronise' the windows between sequences so that similar sequences are not entirely overlooked as a result of a window-boundary misalignment.

Algorithm 2 describes the slotted SPEX algorithm. As in standard SPEX, we pass a fixed-size window over each sequence with an increment of one. However, unlike SPEX, slotted SPEX does not consider inserting every chunk into the hashtable. In addition to decomposing the chunk into subchunks and checking that the subchunks are non-unique, slotted SPEX also requires that one of two trigger conditions be met: either that at least Q window increments have occurred since the last insertion, or that the current chunk already appears in the hashcounter. The parameter Q is the *quantum*, which can be thought of as the window increment used by the algorithm. Slotted SPEX guarantees that at least every Q^{th} overlapping substring from a sequence is inserted into the hashtable. The second trigger condition — that the chunk already appears in the hashcounter — provides the synchronisation that is required for the algorithm to work reliably.

The operation of slotted SPEX is best illustrated with an example. Using the same set of sequences as above, a quantum $Q = 4$ and a chunk length of four, slotted SPEX produces the following set of chunks:

	Sequence	Chunks
Sequence 1	`AB`**CDEFGHIJKL**`MNOP`	`ABCD` **EFGH IJKL** `MNOP`
Sequence 2	`AAB`**CDEFGHIJKL**`MNOP`	`AABC ABCD` **EFGH IJKL** `MNOP`
Sequence 3	`GHAA`**CDEFGHIJKL**`MQ`	`GHAA CDEF` **EFGH IJKL**

For the first sequence, the set of chunks produced does not differ from the naïve pseudo-word technique. Let us now follow the process for the second sequence. The first chunk — `AABC` — is inserted as before. When processing the second chunk, `ABCD`, the number of chunks processed since the last insertion is one, fewer than the quantum Q. However, the condition *lookup[chunk]* $\neq$ *0* on line 5 of Figure 2 is met: the chunk has been previously inserted. The hashcounter is therefore incremented, effectively synchronising the window of the sequence with that of the earlier, matching sequence. As a result, every Q^{th} identical chunk will be identified across the matching region between the two sequences. In this example, the slotted SPEX algorithm selects two chunks of length four that are common to all sequences. Slotted SPEX also differs from regular SPEX by incrementing the word length by Q rather than by one between iterations.

In comparison to the ordinary SPEX algorithm, slotted SPEX requires fewer iterations, consumes less memory and builds smaller indexes. This makes it suitable for the higher chunk density of genomic data. While slotted SPEX is a lossy algorithm, it does offer the following guarantee: for a window size finalLength and a quantum Q, any pair of sequences with a matching subsequence of length *finalLength + Q - 1* or greater will have at least one identical chunk selected. As the length of the match grows, so will the guaranteed number of common chunks selected. Thus, despite the lossiness of the algorithm, slotted SPEX is still able to offer strong assurance that it will reliably detect highly similar pairs of sequences.

6.3 Fingerprinting for identity estimation

In this section, we analyse the performance of slotted SPEX for distinguishing sequence pairs that have a high level of identity from those that do not.

Following Holm and Sander (1998) and Li et al. (2001a), we calculate the percentage identity between a pair of sequences by performing a banded Smith-Waterman alignment (Chao et al., 1992) using a band width of 20, match score of 1, and no mismatch or gap penalty. The percentage identity I for the sequence pair $\langle s_i, s_j \rangle$ is calculated as $I = S(s_i, s_j)/L(s_i, s_j)$ where $S(s_i, s_j)$ is the alignment score and $L(s_i, s_j)$ is the length of the shorter of the two sequences. This score can be functionally interpreted as being the proportion of characters

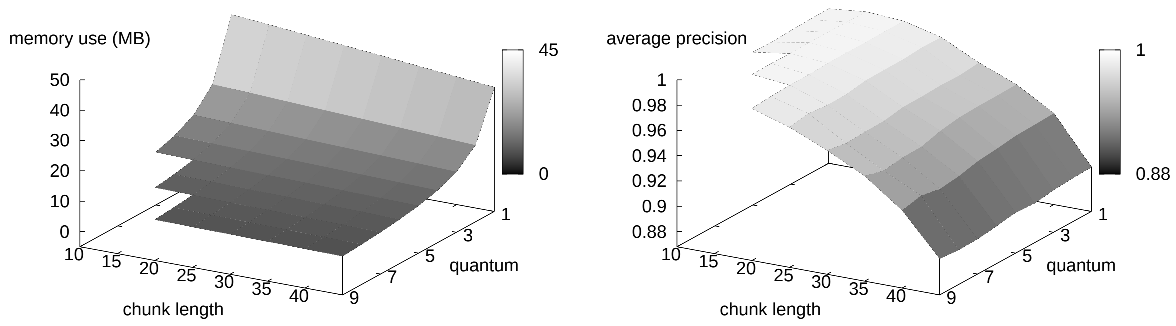


Figure 6.1: Index size as a function of final chunk length and quantum (left) and average precision as a function of final chunk length and quantum (right)

in the shorter sequence that match identical characters in the longer sequence. We define sequence pairs as being redundant if they have at least 90% identity ($I \geq 0.9$). This is the same threshold used in Holm and Sander (1998) and is the default parameter used by CD-HIT (Li et al., 2001a).

For experiments in this section we use version 1.65 of the ASTRAL Compendium (Chandonia et al., 2004), because it is a relatively small yet complete database that allows us to experiment with a wide range of parameterisations. The ASTRAL database contains 24,519 sequences, equating to 300,578,421 unique sequence-pair combinations. Of these, 139,716 — less than 0.05% — have an identity of 90% or higher by the above measure. This is despite the fact that the database is known to have a high degree of internal redundancy. The vast majority of sequence pairs in any database can be assumed to be highly dissimilar.

Although we do not expect fingerprinting to be as sensitive and accurate as a computationally intensive dynamic-programming approach such as Smith-Waterman, we hope that the method will effectively distinguish sequence-pairs with a high level of identity from the large number of pairs that have very low identity. Our aim is to use document fingerprinting to massively reduce the search space within which more sensitive analysis must be pursued.

To find a good compromise between resource consumption and effectiveness, we have experimented with different parameter combinations. We monitored the size of the index generated by SPEX, as well as its average precision, for each of these parameterisations. For the effectiveness measure, we precomputed the identity of all sequence-pairs on the ASTRAL database, and treated all pairs with an identity of 90% or above as being 'relevant'. We then ranked the sequence-pairs detected by the fingerprinting algorithm by their percentage of matching chunks, and computed the average precision (Buckley and Voorhees, 2000) for this list.

Figure 6.1 (left) shows SPEX index size as a function of the chunk length and quantum parameters. The results show that increasing the word length does not result in a large reduction in index size, but increasing the quantum results in a marked and consistent decrease in the size of the index.

Figure 6.1 (right) plots the average precision (Buckley and Voorhees, 2000) as a function of chunk length and quantum. We observe that increasing the chunk length results in a small loss in accuracy, but that increasing the quantum has almost no effect on average precision. This is an encouraging and somewhat surprising result. As a quantum of one is equivalent to the normal SPEX algorithm, our experiments support the conclusion that fingerprinting using slotted SPEX can be used to identify sequence identity nearly as well as the regular SPEX algorithm. Using slotted SPEX with a higher quantim results in reduced memory use, smaller index size, and faster index processing time. Thus, using slotted SPEX, we are able to trade a tolerably small loss in accuracy for a marked in gain in efficiency.

We mention one caveat in interpreting the results of these experiments. The graphs of Figure 6.1 make a strong case for using a shorter word length. However, shorter words place a greater loading on the hashcounter. With larger collections, memory bounds can lead to the hashtable flooding and a consequent blowout in index size. Thus, shorter word lengths are less scalable. Similarly, longer quanta are in general beneficial to performance. However, a larger quantum reduces the number of iterations possible in slotted SPEX. Thus, a very high quantum can result in more collisions in the hashcounter due to fewer iterations, suggesting once again that a compromise is required. Guided by these observations along with the other data, chunk lengths of 25 or 30 with a quantum of 5 to 9 appear to provide a good compromise between the various considerations.

The high average precision results indicate that slotted SPEX provides an accurate and sensitive prediction of whether sequence pairs have a high level of identity. What is particularly surprising is that the average precision stays high even with reasonably long chunk lengths and high quanta. Earlier efforts by Holm and Sander (1998), Li et al. (2001a), and Li et al. (2001b) are extremely rigorous and rely upon short matching chunks, typically less than ten characters in length, between sequence-pairs before proceeding with alignment. Our results indicate that previous approaches were unnecessarily conservative, and that making major efficiency gains by using longer chunk lengths has only a minor impact on result quality.

In our experiments we have focused on identifying sequence-pairs with greater than 90%

identity, and we have shown that fingerprinting is effective at this task. However, it is probable that fingerprinting will prove less useful as the identity threshold is lowered.

6.4 Removing redundant sequences: an application

In this section we demonstrate that fingerprinting can be used to significantly hasten the construction of RSDBs. Fingerprinting is used to determine a list of candidate pairs whose score according to the fingerprinting scheme exceeds a specified threshold. The identity of these candidate pairs is then calculated exactly; all other pairs are assumed to have an identity level below the required threshold. From this point, the standard clustering process is used to group redundant sequences together. However, a vast majority of the pairwise alignments are avoided as a result of the fingerprinting stage.

We measured the performance and scalability of our approach by comparing it to the greedy-incremental algorithm CD-HIT (Li et al., 2001a) — which is freely available for download — using several releases of the comprehensive GenBank NR protein database dating from July 2000 to August 2005, as described in Table 6.1.[1] In all cases we built RSDBs using the standard identity threshold of 90%. For tests with CD-HIT we used default parameters except for `max_memory` which we increased to 1.5 GB. For our approach, we used a target chunk length of 25, a quantum of 9 and 3 iterations. Our threshold for identifying a candidate pair is one matching chunk between the pair. We use this low threshold because our experience showed that it leads to improved accuracy with a negligible increase in execution time. In our experiments with the ASTRAL database and our chosen default parameters, slotted SPEX identifies only 10,143 false positives out of a total of 147,724 sequence pairs identified. This represents a negligible computational burden.

The results in Table 6.1 show no significant difference in RSDB size between our method and CD-HIT, indicating the two approaches are roughly equivalent in terms of accuracy. Figure 6.2 shows the runtime for our approach and CD-HIT for the releases of GenBank tested. A visual inspection reveals that our approach exhibits a near-linear growth rate over the data sets tested, while CD-HIT is clearly superlinear. When processing the August 2005 collection, our approach is more than 6 times faster than CD-HIT.

Note that these results do not allow us to draw any conclusions regarding the asymptotic behaviour of our approach, which may indeed be superlinear. However, as our data sets were

[1]Ideally, we would have had more datapoints for this experiment. However, old releases of the NR database are not officially maintained, and thus we could only find four different releases of the database.

	Original	Size reduction	
Release date	Size (MB)	CD-HIT	Our approach
16 July 2000	157	61.71 MB (39.56%)	61.72 MB (39.57%)
22 May 2003	443	164.38 MB (37.33%)	165.07 MB (37.48%)
30 June 2004	597	217.80 MB (36.71%)	218.76 MB (36.87%)
18 August 2005	900	322.98 MB (36.08%)	324.92 MB (36.30%)

Table 6.1: Reduction in collection size for CD-HIT and our approach for various releases of the GenBank NR database.

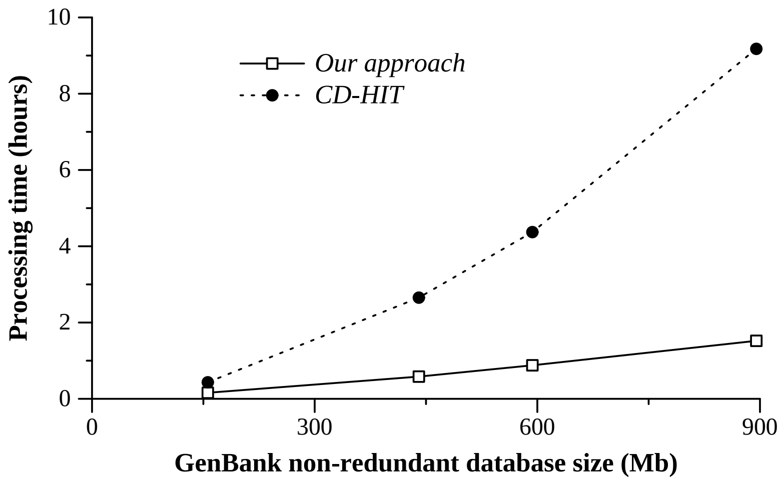


Figure 6.2: Time required to identify and remove redundant sequences from various releases of the GenBank NR database.

representative of the size of collections to which RSDB-construction algorithms are currently applied, it is reasonable to assert that the behaviour of our approach is roughly linear in the size of the input.

6.5 Redundancy management using wildcards

RSDBs are valuable in situations where resources are limited or the user is impatient. They are also valuable for profile-based distant homology search tools like PSI-BLAST. However, they are a blunt instrument: sequences are simply discarded from the database, leaving no trace. This is not ideal if the user wishes to search a comprehensive database, or needs to know the full history of a sequence. Worse still, the sequence in the original database that matches a query best may have been discarded when the RSDB was built. In general homology search, RSDBs tend to be less effective than searches on a comprehensive database.

In this section we describe our novel approach to the management of redundancy in genomic sequence databases that does not possess the deficiencies and drawbacks of RSDBs. Rather than choosing a single representative sequence from each cluster, we use a set of wildcard characters to create a single *union-sequence* that is simultaneously representative of all the sequences in the cluster. A small amount of auxiliary data stored with each cluster allows for the original sequences to be recreated.

During search, the query sequence is aligned with the union-sequence of each cluster. For those union-sequences that produce a statistically significant alignment, the members of the cluster are restored from their compressed representations and aligned to the query. This ensures that precise alignment scores are calculated. Our approach supports two modes of operation: users can choose to see all high-scoring alignments, or only the best alignment from each cluster. The former mode returns results that are indistinguishable from the result of a search against the comprehensive database, while the latter mode reduces redundancy in the results, reducing the user burden in interpreting the results.

The effectiveness of our technique depends upon the careful construction of clusters, the availability of a sensitive but efficient way of scoring the query sequence against the union-sequence, and the selection of a good set of wildcards. We describe our approach to these matters in Sections 6.5.2, 6.5.3, and 6.5.4 respectively.

6.5.1 Cluster representation

We define $E = \{e_1, ..., e_n\}$ as the set of sequences in a collection where each sequence is a string of residues $e_i = r_1...r_n \mid r \in R$. Our approach represents the collection as a set of clusters C, where each cluster contains a union-sequence U and edit information for each member of the cluster. The union-sequence is a string of residues and wildcards $U = u_1...u_n | u_i \in R \cup W$ where $W = \{w_1, ..., w_n \mid w_i \subseteq R\}$ is the set of available wildcards. Each wildcard represents a set of residues and is able to act as a substitute for any of these residues. By convention, w_n is assumed to be the *default wildcard* w_d that can represent any residue; that is, $w_n = R$.

Figure 6.3 shows an example cluster constructed using our approach. The union-sequence is shown at the top and cluster members are aligned below. Columns in which the member sequences differ from each other are shown in bold face; these positions in the cluster are

```
KNQVAMN*QNTVFDAKRLIGRKFDEPTVQADMKHWPFKV*QAEVDV*RFRSNT*ER  (union-seq)
       PQNTVFDAKRLIGRKFDEPTVQADMKHWPFKVIQAEV              (gi 156103)
KNQVAMNPQNTVFDAKRLIGRKFDEPTVQADMKHWPFKVIQAEV              (gi 156105)
        QNTVFDAKRLIGRKFDEPTVQADMKHWPFKVVQAEVDVLRFRSNTKER  (gi 156121)
KNQVAMNPQNTVFDAKRLIGRKFDEPTVQADMKHWPFKVVQAEVDVLRFRSNTK    (gi 552059)
KNQVAMNPQNTVFDAKRLIGRKFDEPTVQADMKHWPFKVIQAEVDVQRFRSNTR    (gi 552055)
KNQVAMNPQNTVFDAKRLIGRKFDEPTVQADMKHWPFKVIQAEVDVQRFRSNTRE   (gi 552057)
       PQNTVFDAKRLIGRKFDEPTVQADMKHWPFKVVQAEVDVLRFRSNTKER  (gi 156098)
        QNTVFDAKRLIGRKFDEPTVQADMKHWPFKVVQAEVDVLRFRS       (gi 156100)
           VFDAKRLIGRKFDEPTVQADMKHWPFKVIQAEVDVQRFRSNTRE   (gi 156111)
       NQNTVFDAKRLIGRKFDEPTVQADMKHWPFKVIQAEVDVQRFRSNTR    (gi 552056)
```

Figure 6.3: Example cluster of heat shock proteins from GenBank NR database. The union-sequence is shown at the top, followed by ten member sequences with GI accession numbers shown in brackets.

represented by a wildcard. In this example, $W = \{w_d\}$ — that is, only the default wildcard is used. It is represented in the figure by an asterisk.

When a cluster is written to disk, the union-sequence — shown at the top of the figure — is stored in its complete form, and each member of the cluster is recorded using edit information. The edit information for each member sequence includes start and end offsets that specify the range within the union-sequence that is occupied by that particular sequence, and a set of residues that replace the wildcards in that range. For example, the first member of the cluster with GI accession 156103 would be represented by the tuple (8,44,PI). The member sequence can be reconstructed by copying the substring between positions 8 and 44 of the union-sequence and replacing the wildcards at union-sequence positions 8 and 40 with characters P and I respectively. Note that our clustering approach does not permit gaps as insertions and deletions are heavily penalised during alignment. Thus, any scheme that includes gaps in clusters will contain clusters in which cluster members may have significantly different alignment scores with respect to a given query. This makes it impossible for alignment against the union-sequence to be used to accurately predict the scores of the sequences it represents.

6.5.2 Clustering algorithm

In this section we describe our approach to efficiently clustering large sequence collections using fingerprinting with the slotted SPEX algorithm. As in the RSDB construction algorithm described in Section 6.4, we use fingerprinting as a preprocessing step to identify candidate pairs. Given the list of candidate pairs, we use a variation on single-linkage hierarchical clustering (Johnson, 1967) to build clusters, as follows. Initially, each sequence is considered to constitute a cluster with one member. Candidate pairs are then processed in increasing order of similarity score, from most- to least- similar, and the pair of clusters that contains the highly-similar candidate sequences are considered for merger.

In general, given two candidate clusters C_X and C_Y with union-sequences X and Y respectively, the following process is used to determine whether the clusters should be merged:

1. X and Y are aligned and the sequence space partitioned into a prefix, an overlap region, and a suffix.

2. The union-sequence candidate U for the new cluster is created by replacing each mismatched residue in the overlap region with a suitable wildcard w.

3. The union-sequence candidate U is accepted if the mean alignment score increase $\bar{Q}$ in the overlap region is below a specified threshold T — this prevents union-sequences from containing too many wildcards and reducing search performance.

If the clusters are merged, a new cluster C_U is created consisting of all members of C_X and C_Y. This process is repeated for all candidate pairs. When inserting wildcards into the union-sequence, if more than one wildcard is suitable then the one with the lowest expected match score $e(w) = \sum_R s(w,r)p(r)$ is selected, where $p(r)$ is the background probability of residue r (Robinson and Robinson, 1991) and $s(w,r)$ is the alignment score for matching wildcard w to residue r. Calculation of wildcard alignment vectors $s(w,\cdot)$ is discussed in Section 6.5.3, and the selection of the pool of wildcards W is discussed in Section 6.5.4.

The alignment score increase Q for a wildcard w is calculated as

$$Q(w) = \sum_R s(w,r)p(r) - \sum_{R\times R} s(r_1,r_2)p(r_1)p(r_2) \tag{6.1}$$

where $s(r_1,r_2)$ is the score for matching a pair of residues as defined by a scoring matrix such as BLOSUM62 (Henikoff and Henikoff, 1992). This value estimates the typical increase

```
                                 VAMNPQNTVMF  (union−sequence)
      Cluster X sequence 1:      VAMNPQNTVMF

                              GRKVIMN*QNTQ    (union−sequence)
      Cluster Y sequence 1:   GRKVIMNCQNTQ
      Cluster Y sequence 2:   GRKVIMNEQNTQ

                              GRKV*MN*QNT*MF  (union−sequence)
    New cluster sequence 1:      VAMNPQNTVMF
    New cluster sequence 2:   GRKVIMNCQNTQ
    New cluster sequence 3:   GRKVIMNEQNTQ
```

Figure 6.4: Merge of two example clusters. Cluster X contains a single sequence and cluster Y contains two sequences. A new cluster is created that contains members of both clusters and has a new union-sequence to represent all three member sequences.

in alignment score one can expect by aligning a random query residue against w instead of against the actual residue at that position.

Figure 6.4 illustrates the process of merging clusters. In this example, cluster X (which contains one sequence) and cluster Y (which contains two sequences) are merged. A new cluster containing the members of both X and Y is created, with a new union-sequence that contains wildcards at residue positions where the three sequences differ.

The approach described above works extremely well for relatively small databases. However, as discussed in Section 3.9, the small number of long postings lists that commonly occur in larger collections constitute a major bottleneck in processing, consuming a disproportionate amount of execution time. We process postings lists with more than M entries — where we use $M = 100$ by default — in a different, top-down manner before proceeding to the standard hierarchical clustering approach discussed above.

The top-down approach identifies clusters from a list of sequences l that contain a frequently-occurring chunk w as follows:

1. All sequences in l are loaded into main memory and aligned with each other.

2. An exemplar sequence is selected. This is the sequence with the highest average percentage identity to the other sequences in l.

3. A new cluster C is created with the exemplar sequence as its first member.

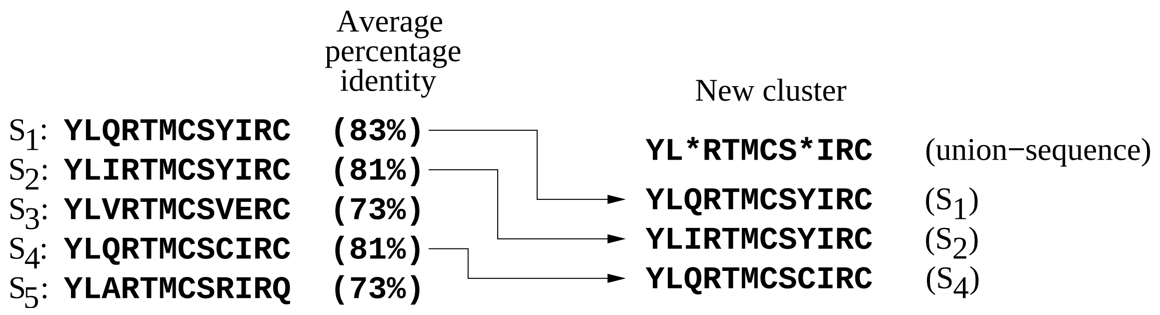


Figure 6.5: Illustration of top-down clustering where sequences $l = \{S_1, S_2, S_3, S_4, S_5\}$ contain the chunk `RTMCS`*. Each sequence is compared to every other sequence in the list and the sequence with the highest average percentage identity (S_1) is selected as the first member of a new cluster. Sequences S_2 and S_4 are highly similar to S_1 and are also included in the new cluster. The remaining sequences $l = \{S_3, S_5\}$ are used to perform another iteration of top-down clustering if $|l| \geq M$.*

4. Each sequence in l is compared to the union-sequence of the new cluster. Sequences where $\bar{Q} < T$ are added to the cluster in order from most- to least- similar using the approach we describe above.

5. All of the members of the new cluster C are removed from l and the process is repeated from step 1 until $|l| < M$.

The top-down clustering is illustrated in Figure 6.5. In this example, a list of five sequences that contain the word `RTMCS` is processed using the top-down method. The sequence S_1 has the highest average percentage identity to the other sequences in l and is selected as the exemplar. A new cluster is created with S_1 as the first member, and sequences S_2 and S_4 are subsequently added. The three members of the new cluster are removed from l, and the process is repeated until $|l| < M$.

Once the postings list has been processed by the top-down method, the shortened list is processed using the hierarchical clustering method described above. While the top-down process is laborious, it need only be used rarely: on the August 2005 release of NR, top-down clustering was necessary for fewer than 0.2% of the postings lists when using default parameters.

Wildcard residues	Scoring vector																							
	L	A	G	S	V	E	T	K	D	P	I	R	N	Q	F	Y	M	H	C	W	B	Z	X	U
L,V,I,F,M	3	-1	-3	-2	3	-2	-1	-2	-3	-2	3	-2	-3	-1	2	0	4	-2	-1	-1	-3	-2	-1	-4
G,E,K,R,Q,H	-2	-1	1	0	-2	3	-1	4	0	-1	-3	3	0	3	-3	-1	-1	3	-3	-2	0	2	-1	-4
A,V,T,I,X	1	2	-1	0	3	-1	3	-1	-2	-1	3	-2	-2	-1	-1	-1	0	-2	-1	-3	-2	-1	-1	-4
S,E,T,K,D,N	-2	0	0	2	-2	2	2	2	4	-1	-2	0	4	1	-3	-2	-1	0	-2	-3	3	1	-1	-4
L,V,T,P,R,F,Y,M,H,C,W	1	-1	-2	-1	1	-1	1	-1	-2	2	0	1	-1	0	2	3	2	2	2	3	-2	-1	-1	-4
A,G,S,D,P,H	-2	3	3	3	-1	0	0	0	2	2	-2	-1	1	0	-2	-2	-1	1	-1	-3	1	0	0	-4
All residues	1	1	1	2	1	2	1	1	1	1	1	1	2	2	1	1	1	2	1	1	1	1	-1	-3

Figure 6.6: Scoring vectors for the wildcards from Table 6.2. The set of residues represented by each wildcard is given in the left-hand column. The scoring vector provides an alignment score between each of the twenty-four amino acid symbols and that wildcard.

6.5.3 Scoring with wildcards

We have modified BLAST to work with our clustering algorithm as follows. Instead of comparing the query sequence to each member of the database, our approach compares the query only to the union-sequence representing each cluster, where the union-sequence may contain wildcard characters. If a high-scoring alignment between the union-sequence and query is identified, the members of the cluster are reconstructed and aligned with the query. In this section we discuss how, given a set of wildcards W, we determine the wildcard scoring vectors $s(w_i, \cdot)$ for each $w_i \in W$.

Ideally, we would like the score between a query sequence Q and a union-sequence U to be precisely the highest score that would result from aligning Q against any of the sequences in the cluster C_U. This would result in no loss in sensitivity as well as no false positives. Unfortunately, such a scoring scheme is not likely to be achievable without aligning against each sequence in every cluster, defeating the purpose of clustering in the first place.

To maintain the speed of our approach, scoring of wildcards against residues must be on the basis of a standard scoring vector $s(w, \cdot)$ and cannot take into consideration any data about the sequences represented by the cluster. Thus, scoring will involve a compromise between sensitivity (few false negatives) and speed (few false positives). We describe two such compromises below, and show how to combine them to achieve a good balance of sensitivity and speed.

During clustering, wildcards are inserted into the union-sequence to denote residue positions at which the cluster members differ. Let us define $S = s_1 ... s_x \mid s_i \in W$ where S is the ordered sequence of x wildcards substituted into union-sequences during clustering of a collection. Each occurrence of a wildcard is used to represent a set of residues that appear in its position in the members of the cluster. We define $o \subseteq R$ as the set of residues represented by an occurrence of a wildcard in the collection and $O = o_1 ... o_x \mid o_i \subseteq R$ as the ordered sequence of substituted residue sets. The k^{th} wildcard s_k that is used to represent the set of residues o_k must be chosen such that $o_k \subseteq s_k$.

Our first scoring scheme, s_{exp}, builds the scoring vector for each wildcard by considering the actual occurrence pattern of residues represented by that wildcard in the collection. Formally, we calculate the expected best score s_{exp} as:

$$s_{exp}(w, r) = \frac{\sum\limits_{k \in P_i} \max\limits_{f \in o_k} s(r, f)}{|P_i|}$$

where P_i is the set of ordinal numbers of all substitutions using the wildcard w_i:

$$P_i = \{j \mid j \ \in \mathbb{N}, j \leq x \ , \ s_j = w_i\}$$

This score can be interpreted as the mean score that would result from aligning residue r against the actual residues represented by the wildcard w. This score has the potential to reduce search accuracy. However, it distributes the scores well, and provides an excellent tradeoff between accuracy and speed.

The second scoring scheme, s_{opt}, calculates the optimistic alignment score of the wildcard w against each residue. The optimistic score is the highest score for aligning residue q to any of the residues represented by wildcard w. This is calculated as follows:

$$s_{opt}(w, r) = \max_{f \in w} s(r, f)$$

The optimistic score guarantees no loss in sensitivity: the score for aligning against a union-sequence U using this scoring scheme is at least as high as the score for any of the sequences represented by U. The problem is that in many cases the score for U is significantly higher, leading to false-positives where the union-sequence is flagged as a match despite none of the cluster members being sufficiently close to the query. This results in substantially slower search.

The s_{exp} and s_{opt} scoring schemes represent two different compromises between sensitivity and speed. We can adjust this balance by combining the two approaches using a mixture model. We define a mixture parameter, λ, such that $0 \leq \lambda \leq 1$. The mixture model score for aligning wildcard w to residue r is defined as:

$$s_\lambda(w, r) = \lambda \cdot s_{opt}(w, r) + (1 - \lambda) \cdot s_{exp}(w, r)$$

The score $s_\lambda(w, r)$ for each $\langle w, r \rangle$ pair is calculated when the collection is being clustered and then recorded on disk in association with the collection. During a BLAST search, the wildcard scoring vectors are loaded from disk and used to perform the search. An example set of scoring vectors $s(w_i, \cdot)$ that were derived using our approach is shown in Figure 6.6. We report experiments with varying values of λ in Section 6.6.

6.5.4 Wildcard selection

Having defined a system for assigning a scoring vector to an arbitrary wildcard, we now describe a method for selecting a set of wildcards to be used during clustering. We assume that one of these wildcards w_n is the default wildcard that can be used to represent any of the 24 residue and ambiguous codes, that is $w_n = R$. The remaining wildcards must be selected carefully: wildcards representing large residue sets can be used more frequently but provide poor discrimination with higher average alignment scores and more false positives. Conversely, small residue sets provide good discrimination but can be used less frequently, thereby increasing the use of larger residue sets such as the default wildcard. Thus, we must balance the tradeoff between sensitivity and applicability.

The first aspect of choosing a set of wildcards to use for substitution is to decide on the size of this set. It would be ideal to use as many wildcards as necessary, so that for each substitution $s_i = o_i$. However, each wildcard must be encoded as a different character, and this approach would lead to a very large alphabet. An enlarged alphabet would in turn lead to inefficiencies in BLAST due to larger lookup and scoring data structures. Thus, a compromise is required. The alphabet used by BLAST for protein representation consists of 25 distinct codes. Each code is represented using 5 bits, permitting a total of 32 codes. Thus, 7 character codes are unused. We have therefore chosen to use $|W| = 7$ wildcards, as this is the maximum number that can be accommodated without increasing the size of the encoding.

We have investigated two different approaches to selecting a good set of wildcards. The first approach to the problem treats it as an optimisation scenario, and works as follows. We first cluster the collection as described in Section 6.5.2 using only the default wildcard,

Minimum alignment score	Physico-chemical classifications	
L,V,I,F,M	L,V,I	(aliphatic)
G,E,K,R,Q,H	F,Y,H,W	(aromatic)
A,V,T,I,X	E,K,D,R,H	(charged)
S,E,T,K,D,N	L,A,G,V,K,I,F,Y,M,H,C,W	(hydrophobic)
L,V,T,P,R,F,Y,M,H,C,W	S,E,T,K,D,R,N,Q,Y,H,C,W	(polar)
A,G,S,D,P,H	A,G,S,V,T,D,P,N,C	(small)
All residues	All residues	(default wildcard)

Table 6.2: Two different sets of wildcards to be used for clustering. Each list is sorted in order from lowest to highest average alignment score A and contains seven entries including the default wildcard. The left-hand list is selected to minimise the average alignment score A using a hill-climbing strategy, and the right-hand list is based on the amino acid classifications described in Taylor (1986).

ie. $W = \{w_d\}$. We use the residue-substitution sequence O from this clustering to create a set W^* of candidate wildcards. Our optimisation goal can then be defined as follows: we wish to select the set of wildcards $W \subseteq W^*$ such that the total average alignment score $A = \sum_{w \in S} \sum_{r \in R} s(w,r)p(r)$ over all substitutions S is minimised. A lower value of A implies a reduction in the number of high-scoring matches between a typical query sequence and union-sequences in the collection, and consequently a reduction in the number of false-positives in which cluster members are fruitlessly recreated and aligned to the query.

In selecting the wildcard set W that minimises A, we use the following greedy approach: first, we initialize W to contain only the default wildcard w_d. We then scan through W^* and select the wildcard that leads to the greatest overall reduction in A. This process is repeated until the set W is filled, at each iteration considering the wildcards already in W in the calculation of A. Once W is full, we employ a hill-climbing strategy where we consider replacing each wildcard with a set of residues from W^* with the aim of further reducing A.

A set of wildcards was chosen by applying this strategy to the GenBank NR database. The left-hand column of Table 6.2 lists the wildcards that were identified using this approach and used by default for experiments reported in this chapter.

Our second approach to wildcard selection involves manually selecting wildcards to represent groups of amino acids with similar physico-chemical properties. We used the amino acid classifications described in Taylor (1986) to define the set of seven wildcards shown in the right-hand column of Table 6.2. In addition to the default wildcard, six wildcards were defined to represent the aliphatic, aromatic, charged, hydrophobic, polar, and small classes of amino acids. We present experimental results for the two wildcard selection strategies in the following section.

6.6 Results

In this section we analyse the effect of our redundancy management strategy on collection size and search times.

A set of 8,759 test queries were extracted from the ASTRAL database such that no two of the queries shared more than 90% identity. To measure search accuracy, each query was searched against the ASTRAL database and the commonly-used Receiver Operating Characteristic (ROC) score was used (Gribskov and Robinson, 1996). A match between two sequences was considered positive if they came from the same superfamily as classified by the SCOP database (Murzin et al., 1995; Andreeva et al., 2004). Otherwise, it was considered negative. The ROC_{50} score provides a measure between 0 and 1, where a higher score represents better sensitivity (detection of true positives) and selectivity (ranking true positives ahead of false positives).

The SCOP database is too small to provide an accurate measure of search time, so we use the GenBank NR protein database to measure search speed. The GenBank collection was downloaded August 18, 2005 and contains 2,739,666 sequences in around 900 megabytes of sequence data. Performance was measured using 50 queries randomly selected from GenBank NR. Each query was searched against the entire collection three times with the best runtime recorded and the results averaged. Experiments were conducted on a Pentium 4 2.8GHz machine with 2 GB of main memory.

We used FSA-BLAST with default parameters as a baseline. To assess the clustering scheme, the GenBank and ASTRAL databases were clustered and FSA-BLAST was configured to report all high-scoring alignments, rather than only the best alignment from each cluster. All reported collection sizes include sequence data and edit information but exclude sequence descriptions. CD-HIT version 2.0.4 beta was used for experiments with 90% clustering threshold and maximum memory set to 1.5 GB. We also report results for NCBI-BLAST version 2.2.11 and our own implementation of Smith-Waterman that uses the same scoring functions and statistics as BLAST (Karlin and Altschul, 1990; Altschul and Gish, 1996).

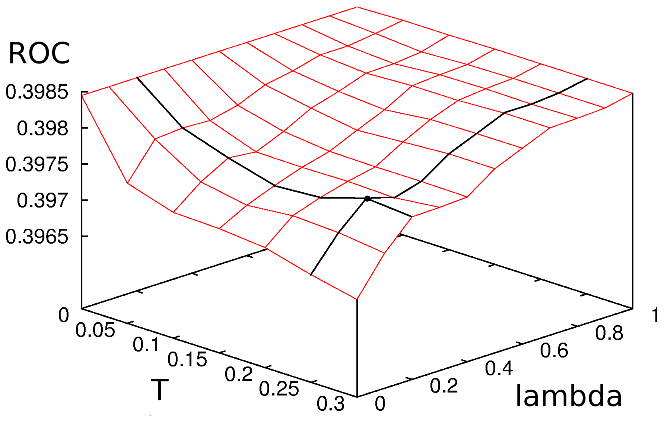


Figure 6.7: Search accuracy for collections clustered with varying values of λ and T. Default values of $\lambda = 0.2, T = 0.25$ are highlighted.

The Smith-Waterman results represent the highest possible degree of sensitivity that could be achieved by BLAST and provides a meaningful reference point. No sequence filtering was performed for our experiments in this chapter.

Figure 6.7 shows the effect on accuracy for varying values of λ and T. We have chosen $\lambda = 0.2$ as a default value because smaller values of λ result in a larger decrease in search accuracy, and larger values reduce search speed. We observe that for $\lambda = 0.2$ there is little variation in search accuracy for values of T between 0.05 and 0.3.

Figure 6.8 shows the effect on search times for varying values of T where $\lambda = 0.2$. As T increases the clustered collection becomes smaller, leading to faster search times. However, if T is too large then union-sequences with a high percentage of wildcards are permitted, leading to an increase in the number of cluster members that are recreated and a corresponding reduction in search speed. We have chosen the value $T = 0.25$ that maximises search speed.

The overall results are shown in Table 6.3. When used with default settings of $\lambda = 0.2$ and $T = 0.25$, and the set of wildcards selected to minimise alignment score in Table 6.2, our clustering approach reduces the overall size of the NR database by 27% and improves search times by 22%. Importantly, the ROC score indicates that there is no significant effect on search accuracy. If users are willing to accept a small loss in accuracy, then the parameters $\lambda = 0$ and $T = 0.3$ improve search times by 27% and reduce the size of the sequence collection by 28% with a decrease of 0.001 in ROC score when compared to our baseline. Since we are interested in improving performance with no loss in accuracy we do not consider these non-default settings further. Overall, our clustering approach with default

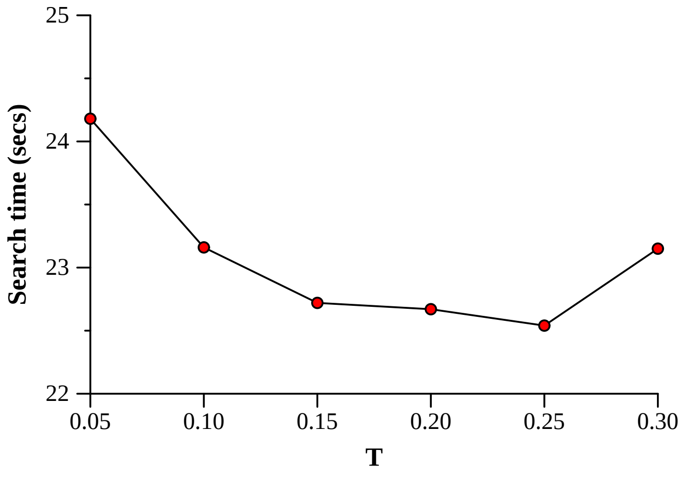


Figure 6.8: Average BLAST search time using $\lambda = 0.2$ and varying values of T.

parameters combined with improvements to the gapped alignment (Cameron et al., 2004) and hit detection (Cameron et al., 2006c) stages of BLAST allow the speed of FSA-BLAST to double that of NCBI-BLAST with no significant effect on accuracy. Both versions of BLAST produce ROC scores 0.017 below the optimal Smith-Waterman algorithm.

The results in Table 6.3 also show that our scheme is an effective means of compressing protein sequences, a task that has been deemed difficult by previous researchers (Nevill-Manning and Witten, 1999; Weiss et al., 2000). Assuming a uniform, independent distribution of amino acids, protein sequence data can be represented with 4.322 bits per symbol (Nevill-Manning and Witten, 1999). Our clustering scheme is able to reduce the space required to store protein sequence data in the GenBank non-redundant database to around 3.15 bits per symbol. To our knowledge, this is significantly less than the current best compression rate of 4.051 bits per symbol (Nevill-Manning and Witten, 1999).

In Table 6.4 we compare search accuracy and performance for our two wildcard selection strategies. In both cases, search performance was superior when compared to collections clustered using only the default wildcard. This supports our approach of using multiple wildcards to construct clusters. The set of wildcards that were selected using the machine-optimisation strategy outperformed the wildcards chosen according to established physico-chemical classifications This is not a surprising result; treating the selection of wildcards as an optimisation problem allows us to choose those that have the greatest direct impact on search performance. Nonetheless, it is interesting to note that the wildcards selected by this process bear little resemblance to any of the traditional physico-chemical amino-acid

Scheme	GenBank NR		ASTRAL
	Time	Sequence data	
	secs (% baseline)	Mb (% baseline)	ROC$_{50}$
FSA-BLAST			
No clustering (baseline)	28.75 (100%)	900 (100%)	0.398
Cluster $\lambda = 0.2, T = 0.25$	22.54 (78%)	655 (73%)	0.398
Cluster $\lambda = 0, T = 0.3$	20.97 (73%)	650 (72%)	0.397
NCBI-BLAST	45.75 (159%)	898 (100%)	0.398
Smith-Waterman	—	—	0.415

Table 6.3: Average runtime for 50 queries searched against the GenBank NR database, and SCOP ROC$_{50}$ scores for the ASTRAL collection.

taxonomies.

Figure 6.9 shows a comparison of clustering times between CD-HIT and our novel clustering approach for four different releases of the GenBank NR database. Inspection suggests that the clustering time of our approach is linear in the collection size and the CD-HIT approach is superlinear (Figure 6.9). On the August 2005 release of the GenBank NR protein database, CD-HIT is around nine times slower than our approach, and we expect this ratio to increase further with collection size.

Table 6.5 shows the amount of redundancy in the GenBank NR database as it has grown over time, measured using our clustering approach. We observe that the proportion of redundancy within the collection is remaining approximately constant, at 27%–29% across versions of the collection tested. This suggests that redundancy will continue to affect genomic data banks as they grow further in size.

Figure 6.10 shows the distribution of cluster sizes on log-log axes for the GenBank NR database. The linear distribution of the data points on these axes suggests that the sizes follow a power-law distribution. Around 55% of clusters contain just two members, and the largest cluster contains 488 members. Of the ten largest clusters identified by our approach, five relate to HIV proteins, three relate to cytochrome b, one relates to elongation factor 1α, and one relates to cytochrome oxidase subunit I. This supports our previous hypothesis that levels of redundancy in the database are proportional to the interest in a research area.

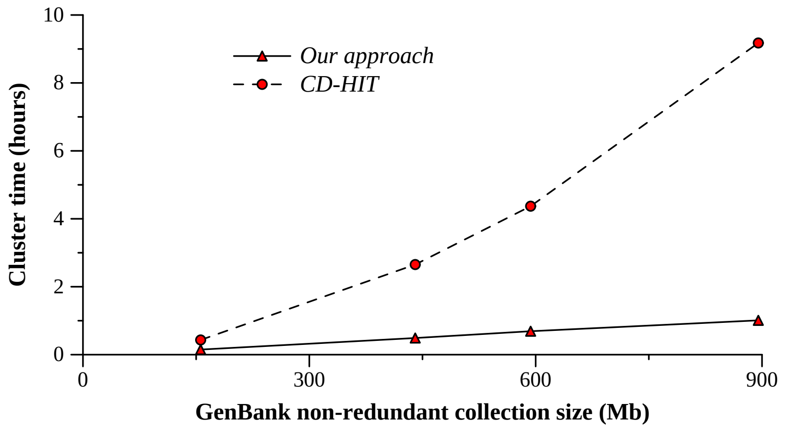


Figure 6.9: Clustering performance for GenBank NR databases of varying sizes.

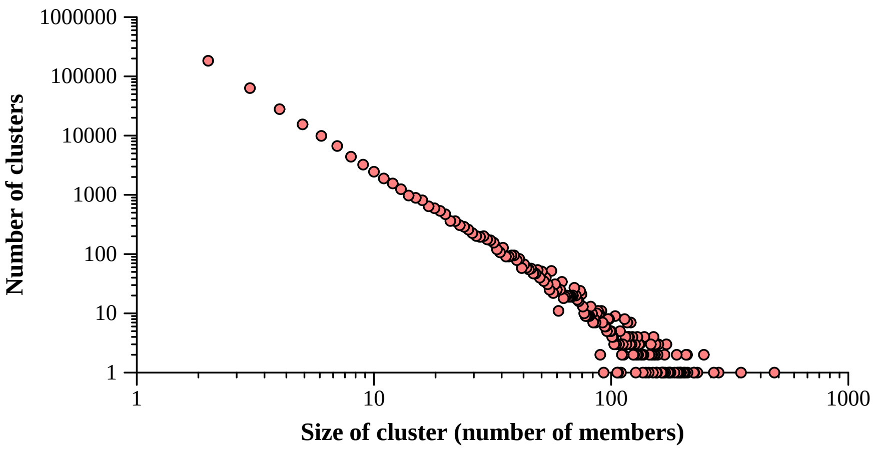


Figure 6.10: Distribution of sizes for clusters identified in the GenBank NR database.

Wildcard set	GenBank NR		ASTRAL
	Time	Sequence data	
	secs (% baseline)	Mb (% baseline)	ROC$_{50}$
Minimum alignment score	22.54 (78%)	655 (73%)	0.398
Physico-chemical classifications	23.25 (81%)	656 (73%)	0.398
Default wildcard only	23.49 (82%)	663 (76%)	0.398

Table 6.4: Average runtime and SCOP ROC$_{50}$ scores for varying sets of wildcards. The first two rows contain results for the wildcard sets defined in Table 6.2. The third row contains results for clustering with only the default wildcard $W = \{w_d\}$

Release date	Number of sequences	Collection Size (MB)	Overall size reduction (MB)	Percentage of collection
16 July 2000	521,662	157	45	28.9%
22 May 2003	1,436,591	443	124	28.1%
30 June 2004	1,873,745	597	165	27.4%
18 August 2005	2,739,666	900	245	27.3%

Table 6.5: Redundancy in GenBank NR database over time.

6.7 Summary

Sequence databanks such as GenBank contain a large number of redundant sequences. Such redundancy has several negative effects including larger collection size, slower search, and difficult-to-interpret results. Redundancy within a collection can lead to over-representation of alignments within particular protein domains, distracting the user from other potentially important results. Furthermore, high levels of redundancy can lead to degraded performance in iterative profile-based search tools such as PSI-BLAST.

We have applied document fingerprinting techniques to genomic data with the aim of more efficiently identifying pairs of redundant sequences in large collections. We have described a new algorithm that we call slotted SPEX. This algorithm requires less main memory and CPU resources compared with regular SPEX when processing genomic collections. We show that slotted SPEX is highly accurate for identifying high-identity sequence pairs, even when using long chunk lengths and large quanta. We have also tested the effectiveness of our slotted

SPEX approach for removing redundant sequences from large collections. When processing the recent GenBank non-redundant protein database our scheme is more than six times faster than the previous fastest approach, CD-HIT, with no significant change in accuracy. Further, our approach scales approximately linearly with collection size.

We have proposed a new scheme for managing redundancy. Instead of discarding near-duplicate sequences, our approach identifies clusters of redundant sequences and constructs a special union-sequence that represents all members of the cluster through the careful use of wildcard characters. We present a new approach for searching clusters that, when combined with a well-chosen set of wildcards and a system for scoring matches between wildcards and query residues, leads to faster search times without a significant loss in accuracy. Moreover, by recording the differences between the union-sequence and each cluster member using edit information, our approach compresses the collection. Our scheme is not dependent on the implementation of a particular tool and thus can be adapted to most homology search tools.

We have integrated our algorithm into FSA-BLAST[2], a new version of BLAST that is substantially faster than NCBI-BLAST. Our results show that our clustering scheme reduces BLAST search times against the GenBank non-redundant database by 22% and compresses sequence data by 27% with no significant effect on accuracy. We have also described a new system for clustering based on fingerprinting with the slotted SPEX algorithm. Our implementation can cluster the entire GenBank NR protein database in one hour on a standard workstation and scales linearly in the size of the collection. We propose that pre-clustered copies of the GenBank collection be made publicly available for download.

We have thus far confined our experimental work to protein sequences and plan to investigate the effect of our clustering scheme on nucleotide data as future work. We also plan to investigate the effect of our approach on iterative search algorithms such as PSI-BLAST, and how our scheme can be used to improve the current measure of the statistical significance of BLAST alignments.

[2]FSA-BLAST is freely available for download at `http://www.fsa-blast.org/`

Chapter 7

Redundancy management for distributed search

This chapter contains material that first appeared in Bernstein et al. (2006). and is based on research conducted in collaboration with fellow PhD candidate Milad Shokouhi. I invented the grainy hash vector data structure, wrote an implementation of the concept, and conducted the experimental evaluations presented in this chapter. Milad contributed to the refinement of the structure and to the design of the experiments. We both contributed equally to the writing of the original paper, text from which appears in this chapter.

Management of redundancy in a distributed information retrieval (DIR) environment is a critical issue. In addition to the standard problems of intra-collection redundancy discussed in Chapter 5, we must contend with the issue of inter-collection redundancy or *collection overlap*. In previous work in DIR, it has generally been assumed that the overall collection is partitioned into disjoint subcollections at the various distributed sites (Callan and Connell, 2001; Nottelmann and Fuhr, 2003; Si and Callan, 2003a; 2004). This is not, in many cases, a reasonable assumption. The decentralised nature of DIR systems means that there is no way of coordinating the composition of individual collections in the system. Overlap (and hence redundancy) between collections is nearly inevitable in any DIR system of significant size and complexity. It has been suggested that tackling this issue is one of the most important challenges in contemporary DIR (Allan et al., 2003).

Management of redundancy in a distributed environment is subject to additional constraints that makes the approaches discussed in Chapter 5 impractical. The lack of centralisation (both physical, and in terms of control) means that preprocessing of collections is not an option; redundancy management must occur at query processing time. Thus, to be usable in a DIR setting, a redundancy management technique must be economical both in terms of its bandwidth consumption and computational effort. Of the current fingerprinting approaches, chunking fails because it is too cumbersome and expensive to compute, while deterministic term extraction fails because it relies on common collection statistics that are not readily available in a distributed environment.

Some work in web-based meta-search engines has included efforts to eliminate exact duplicate documents from the list of final results (Gauch et al., 1996; Meng et al., 2002; Selberg and Etzioni, 1997; Zamir and Etzioni, 1999). However, the proposed techniques are restricted to removing exact matches that refer to the same URL. This technique is quite limited; in particular, it is only applicable to domains in which documents are identified by a URL or some other unique identifier. Many domains have no concept of a unique identifier for a document. Furthermore, the issue of non-identical redundant documents is not addressed at all by this technique. To the best of our knowledge, more sophisticated measures for filtering redundant documents in DIR have not previously been explored.

In this chapter we address for the first time the issue of robust detection of content equivalence between documents in a distributed retrieval environment. We introduce a compact feature, the *grainy hash vector* (GHV), that has properties that meet the requirements for this problem domain. We analyse the properties of GHVs, and show empirically that they can be used for efficient and accurate merge-time detection of content-equivalent documents.

7.1 Duplicates in distributed information retrieval

As discussed in Section 2.5, a DIR environment consists of several autonomous search systems — or servers — and a search broker. A searcher interacts only with the broker; it is the role of the broker to seamlessly interact with the autonomous servers that make up the search environment and return a standard ranked list of documents to the user. In building a final ranked list to return to the user, the broker must perform the resource selection and result merger tasks, both described in Section 2.5.

Poor performance at either of these two tasks can have a detrimental effect on user satisfaction. If promising servers are not identified at the server selection phase, many good documents will never even be returned to the broker. If result lists are not carefully merged, then the ordering of documents in the ranked list can be far from optimal. Inter-document and inter-collection redundancy can have a distorting effect on both of these operations. Just as redundancy between documents affects their marginal worth to the user with respect to each other, redundancy between collections can have a similar effect. There is little value in selecting two collections that are highly mutually redundant, even if each collection is valuable in isolation. Similarly, the presence of duplicate documents in ranked lists affects the result merging process. As discussed in Chapter 5, it is not useful to return many copies of the same relevant document.

The fact that redundancy has a negative effect at both the server selection and result merging stages suggests that it ought to be managed at both. At the server selection stage, the broker can avoid selecting servers that exhibit a high degree of collection overlap with a server that has already been selected. For such an approach to be effective, the rate of overlap between the collections underlying pairs of servers must be accurately estimated in advance. Small estimation errors may lead to the loss of many relevant documents located in the ignored servers.

Hernandez and Kambhampati (2005) introduced the COSCO system, which manages duplicate documents at selection time. The system estimates the overlap between different bibliographic servers and avoids selecting pairs of servers that have high overlap for the same query. The system is trained using 90% of the queries in a log gathered by BibFinder.[1] Using the training query set, the amount of overlap between servers of the same class is calculated. The remaining 10% of the query log is then used to test the system. The CORI system (Callan et al., 1995) is used as a benchmark and it is shown that COSCO finds more relevant documents by selecting the same number of servers as CORI. As in the other previous work, COSCO only considers exact duplicates. It also requires a large set of training queries and is only tested on bibliographic data, which raises questions as to its general applicability.

For redundancy management to take place at the result merger stage, redundant documents must be identified within the merged result list and purged before the list is returned to the user. We believe duplicate management at merge time is important even if the broker makes use of inter-collection redundancy estimates at server selection time. This is because redundant documents can be present at merge time even if their corresponding servers do not have a significant rate of overlap. Furthermore, non-identical redundant document-pairs can be more easily detected at this stage.

[1]`http://kilimanjaro.eas.asu.edu/`

Although distributed search over servers with overlapping collections has been acknowledged as a problem in many papers (Allan et al., 2003; Meng et al., 2002), no serious attempt has been made to manage the problem at result merge time. ProFusion (Gauch et al., 1996), MetaCrawler (Selberg and Etzioni, 1997), and Grouper (Zamir and Etzioni, 1999) all attempt to eliminate exact duplicate documents from the final results. To do this, they simply aggregate those results that point to the same location according to their URL. This method is crude and has a number of deficiencies: it is ineffective for identical documents with different URLs (such as mirrored documents), for near-identical documents, and for domains in which a unique document identifier such as a URL is not available. Clearly, more sophisticated techniques for redundant management are desirable.

7.2 Merge-time redundancy management for DIR

Since collections are not centrally managed, it is not practical to use a preprocessing approach to redundancy management. Rather, it must occur at query time based on additional document information transmitted to the broker. Thus, management of redundant documents is highly sensitive to both computational cost (because it must occur at runtime) and bandwidth (because transmission of additional information in a distributed environment may not be cheap). In this section, we consider the feasibility of using existing document-fingerprinting techniques and features for this task.

The fingerprinting option that seems superficially most suitable to the requirements of redundancy management in a DIR environment is deterministic term extraction, discussed in detail in Section 3.7. The fact that each document only has a single representative hash means that the additional bandwidth requirement in using deterministic term extraction is minimal, and that comparisons between documents at runtime are quick, as all that is required is an equality test between the two representative hashes. The hashes themselves are cheap to compute at index time.

However, as discussed in Section 3.7, deterministic term extraction has its weaknesses. While several empirical tests seem to show that deterministic term extraction techniques are robust in practice (Chowdhury et al., 2002; Cooper et al., 2002; Kolcz et al., 2004), there has been no theoretical analysis that would verify the claim of robustness. It always remains the case that a single difference between a pair of documents, if it occurs in a critical term, can cause the system to fail to identify them as redundant. Pugh and Henzinger (2003) and Kolcz et al. (2004) fortify the robustness of the technique by extracting several hashes based on different term subsets. However, this has the effect of making deterministic term extraction less appealing for the DIR domain, as it increases both the bandwidth cost and the comparison cost at merge-time.

In addition to the reservations above, deterministic term extraction suffers from further complications in the domain of DIR. The deterministic term extraction systems in the literature depend on knowing inverse document frequencies of terms, or some similar collection statistics. In situations where there is a single collection, this presents very little difficulty. However, in a distributed environment where collections may be physically scattered, synchronising collection statistics is a significant challenge, and may be an expensive operation. This is particularly true if individual collections in the system are dynamic. If statistics are not synchronised, however, identical documents on different servers may produce different hashes. Thus, it becomes necessary in a DIR context to use a term extraction heuristic that does not use collection statistics. It is not apparent that this can be done while allowing the technique to remain effective.

Chunking techniques are not particularly suitable for the DIR environment, as they generally require that a large number of chunks be retained and compared in order to achieve reasonable effectiveness. In the DIR context, this translates to significant bandwidth requirements and high computational costs at the broker at merge-time.

However, the class of chunking functions that have a fixed resolution — that is, select a fixed number of chunks no matter the document length — at least offer performance guarantees. Of these, the minimal chunk sampling technique (Fetterly et al., 2003) has two advantages that make it the best choice. The first is that because it is an ordered vector of chunks rather than a simple bag or set, the computational cost for comparison at query-time is reduced. The second is that it has clear statistical properties governing its performance: the distribution of the number of matching hashes between a pair of documents with resemblance r is binomial parameterised by r and by the size of the vector ρ. Using this information we can easily calculate the expected performance of the scheme. For more details refer to Section 3.6.5.

Fetterly et al. (2003) use $\rho = 84$ to generate a vector of 84 hashes of 32 bits each. While such a vector would undoubtedly be a high-quality feature for detecting near-duplicate documents, at 336 bytes per document it imposes a bandwidth premium that may be onerous in many contexts, and requires a substantial computational effort to make a comparison. Even if ρ were reduced substantially, the vector would still need to be several dozen bytes in size to be effective for reliable identification of near-duplicate documents.

7.3 Grainy hash vectors

In this section, we describe our novel document-digest feature for redundant document identification, the *grainy hash vector* (GHV). The GHV is a special implementation of the minimal-chunk sampling technique described in Section 7.2. It includes innovations that combine the benefits of regular minimal-chunk sampling with those of deterministic term extraction for the domain of inter-collection redundancy management. In particular, grainy hash vectors:

- are designed to fit into a single machine word of either 32 or 64 bits;
- have analyzable theoretical properties;
- use bit-parallelism to allow fast comparison between vectors; and
- are robust in the presence of small differences between a pair of documents.

A grainy hash vector parameterised by n and w is a n-bit vector consisting of ρ w-bit hashes, where

$$\rho(n, w) = \left\lfloor \frac{n}{w} \right\rfloor \tag{7.1}$$

In general, w should be a factor of n in order to avoid wasting space in the vector. Each of the hashes in the GHV is produced using the minimal-chunk sampling technique, as described in Section 7.2. For example, a GHV with $n = 32$ and $w = 2$ would consist of 16 2-bit hashes, each produced using the minimal-chunk sampling technique. By using small w, it is possible to pack a large number of hashes into a small space. The tradeoff is, of course, that the probability of collision at each point in the vector becomes non-negligible. In the case of very small w, the collision probability can be quite high. However, we demonstrate that, with an appropriate match threshold, GHVs can provide powerful discrimination of redundant documents, even with $n = 32$.

When w is small, there will be an overwhelming bias in terms of the value of the minimal-chunk hash. For $w = 1$, one would expect the minimal hash to have value 0 in most cases. In order to remove this bias, a much larger w — for example 32 — is used for the initial ordering of hashes from the document. Once the minimal hash value has been identified, the w least-significant bits of this hash are stored in the GHV as the hash for the current field.

For two documents with resemblance r, the probability ϕ of a match between the hashes at a given position in the GHV with vector-width w is given by the function:

$$\phi(r; w) = r + (1 - r)(2^{-w}) \tag{7.2}$$

The first component of this function follows directly from the property of min-wise independence, which results in the probability of a hash match between two documents with resemblance r being equal to r. The second component is the probability that, in the case that the two source chunks are not the same, a collision renders them identical in the hash vector.

Given that these probabilities are independent for each field in the GHV, we note that the number of matching fields between a pair of documents with resemblance r is governed by a binomial distribution, parameterised as follows:

$$Bi(\rho(n, w), \phi(r; w)) \tag{7.3}$$

The following properties are thus directly derivable from the distribution:

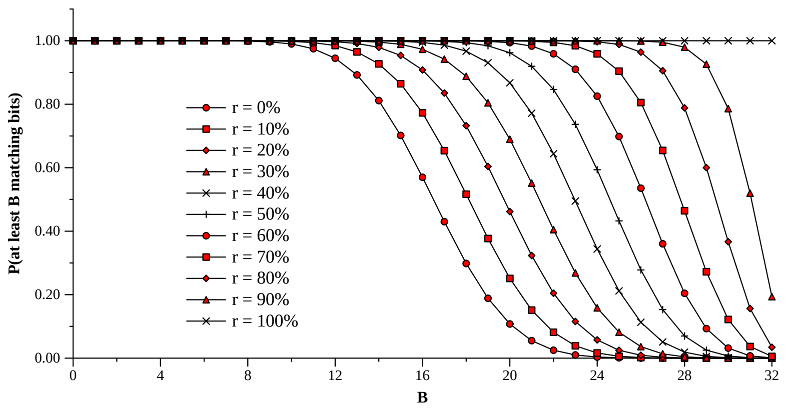
$$P(X = k) = \binom{\rho}{k} \phi^k (1 - \phi)^{\rho - k} \tag{7.4}$$

$$E(X) = \rho\phi \tag{7.5}$$

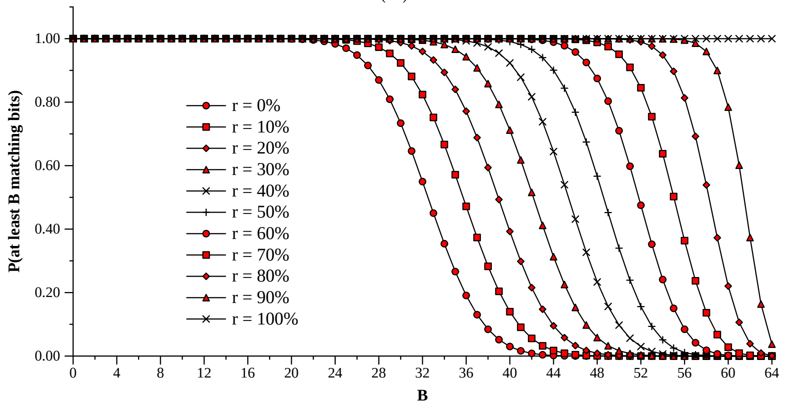
$$\sigma^2 = \rho\phi(1 - \phi) \tag{7.6}$$

GHVs are used in conjunction with some threshold to determine whether a given pair of documents should be considered content-equivalent. For example, we may have $n = 32$, $w = 2$ and a threshold of 14, meaning that at least 14 of the $\rho(n, r) = 16$ hashes in the corresponding vectors of a pair of documents must match for them to be considered near-duplicates. Thus, we are interested in finding a set of parameters for our GHV that minimises the error level.

Figure 7.1 displays a number of curves, derived directly from the binomial distribution, showing the probability that the number of matching fields exceeds a given number for various document resemblance values. While these graphs use $w = 1$, similar curves obtain for different values of w. Introducing a threshold over GHVs effectively draws a vertical line across these curves. Thus, the steeper the curves, the more precise the separation between documents with high levels of resemblance and those with low resemblance. Naturally, the 64-bit GHVs are superior in this respect.



(a)



(b)

Figure 7.1: Probability that the number of matching fields exceeds a value for (a) 32-bit and (b) 64-bit hash vectors, $w = 1$ and various r.

7.3.1 Computing GHV similarity

The single-word nature of grainy hash vectors allows us to rapidly compute the number of matches between a pair of vectors by taking advantage of hardware-level bit parallelism, whereby bitwise operations are performed simultaneously on all bits in a word. We can use the bitwise exclusive-or (XOR) operator to identify the points in a pair of vectors that do not match. The nature of the XOR operator is such that the resultant vector will have a zero value where the vectors match and a one value where there is a mismatch. For a GHV one machine word in size, this operation is completed in a single instruction. An example of how the XOR operator can be used to quickly determine the number of matches between a pair of GHVs is presented in Figure 7.2.

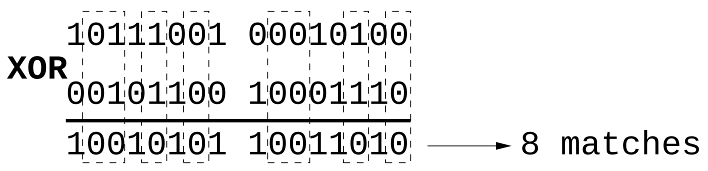


Figure 7.2: A demonstration of how the XOR *operator can be used to quickly compare two 16-bit GHVs with $w = 1$. The number of zero-bits in the resultant vector is equal to the number of matches between the GHVs.*

In the case of $w = 1$ we need only count the number of one-bits in the XOR vector — the *population function* — in order to determine the number of mismatches between the two vectors. While current hardware does not usually implement this function directly, it can still be computed efficiently either with the assistance of lookup tables or using a 'divide-and-conquer' approach such as the one described by Warren (2002).[2]

If w is greater than one, the situation becomes more complicated. We cannot simply count one-bits in the XOR vector, but must count the number of fields in which there is *at least* one mismatched bit. If w is a power of two then we can modify the XOR vector so that the number of 1-bits equals the number of mismatches in $O(\log w)$ time by using shift operators to collapse all bits in each field onto a single bit.

7.4 Experimental evaluation

In this section we report on the results of experiments intended to validate the speed and effectiveness of GHVs in removing duplicates and near-duplicates from result lists in practice. For simplicity, this work uses the results from a centralised IR system. However, the results are applicable to a DIR context.

We used the Associated Press (AP) newswire data, a subset of the TIPSTER collection used for the TREC Ad Hoc retrieval track (Harman, 1993). The AP collection consists of 237,569 documents totalling 729 MB in size.

For each document in the AP collection we created 32 and 64 bit GHVs with w of 1, 2, 4, and 8, using the fast hashing function described by Ramakrishna and Zobel (1997). We then used DECO to calculate the resemblance between all pairs of documents in the AP collection, and the Zettair software[3] with default settings to create a ranked result list of depth 1,000

[2]Many approaches for bit-level problems (including this one) can be found in Sean Eron Anderson's manual 'Bit Twiddling Hacks': `http://graphics.stanford.edu/~seander/bithacks.html`

[3]`http://www.seg.rmit.edu.au/zettair/`

on the AP documents for each of the TREC topics 51–100.

For each topic we loaded the GHVs for the 1,000 documents appearing in the ranked result list. Each pair of GHVs in this list were then compared using the algorithm described in Section 7.3.1, resulting in 499,500 comparisons taking place. Those pairs with a number of matching fields exceeding a threshold t were returned. This process was repeated for all possible values of t.

7.4.1 Comparison efficiency

Performing the 499,500 GHV comparisons required for a full comparison of 1,000 results took approximately 0.1 seconds on a 2.67 GHz Intel Pentium 4 desktop computer with 512 MB RAM, using moderately optimised code. This figure did not vary significantly whether the GHVs were 32 or 64 bits in length, and no matter the value of w. This suggests that memory access costs dominated the very small number of register instructions required to process each GHV pair. In comparison to the significant other costs present in a DIR system — in particular network latency — we argue that this is a very reasonable cost to bear in order to improve the user experience.

Note that the timings presented above are pessimistic, based on an assumption that the GHVs for a full ranked list of 1,000 documents need all be compared to each other. This is not generally necessary if the comparisons are undertaken in a 'lazy' fashion. Many search systems present results incrementally, often ten at a time; we can take advantage of this by only undertaking the GHV comparisons necessary to present the next page of results. Much of the time — depending upon the level of redundancy in the ranked list and the persistence of the user — this can reduce the number of comparisons required from hundreds of thousands to just hundreds. Thus, we assert that the typical cost of merge-time GHV comparison is negligible in comparison to the cost of other stages of the DIR process.

7.4.2 GHV accuracy

In Chapter 5 we established an S_3 score of 0.58 as the threshold at which we consider a document pair to be conditionally equivalent. As the use-case is essentially the same in the current context, it is appropriate to adopt this threshold for the current set of experiments. We thus calibrate the performance of GHVs against the previously evaluated performance of DECO with a fixed threshold of 0.58, rather than making a direct evaluation with a user study.

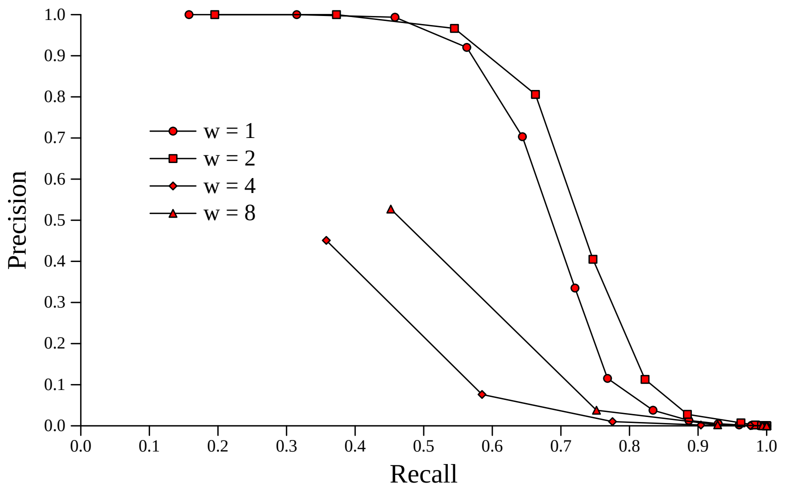


Figure 7.3: Recall-precision curves for various w and GHVs of size 32 bits

We define the recall of a particular parameterisation of GHVs to be the proportion of document pairs identified as conditionally equivalent by DECO that are also identified using GHVs, and the precision as the proportion of all document pairs identified using GHVs that are also identified as conditionally equivalent by DECO. By altering the threshold for the number of matching fields between a pair of GHVs before the corresponding documents are classified as conditionally equivalent, we are able to obtain recall-precision curves over a particular GHV parameterisation.[4] It is important to bear in mind that these recall-precision curves report fidelity with the behaviour of DECO rather than a more direct measurement of the effectiveness of GHVs.

Figure 7.3 shows mean recall-precision curves for the 50 TREC topics using 32-bit GHVs and various values of w. As can be seen, the curve for $w = 2$ clearly dominates, followed by w values of 1, 8, and 4. For $w = 4$ and $w = 8$, the precision level is inadequate for use even at the lowest threshold levels. For $w = 1$ and $w = 2$, precision is adequate up to a recall level of approximately 0.5, meaning that we can use 32-bit GHVs to detect about half the content-equivalent document pairs in the result list without introducing a significant false-positive problem.

Figure 7.4 shows the same curves when 64-bit GHVs are used. As expected, these curves are significantly better than the corresponding curves for 32-bit GHVs. Interestingly, the field widths dominate each other in the same order, with $w = 2$ performing best, followed

[4]Note that the threshold discussed here is different from the S_3 threshold of 0.58, which is held static and is only used for the purpose of comparison.

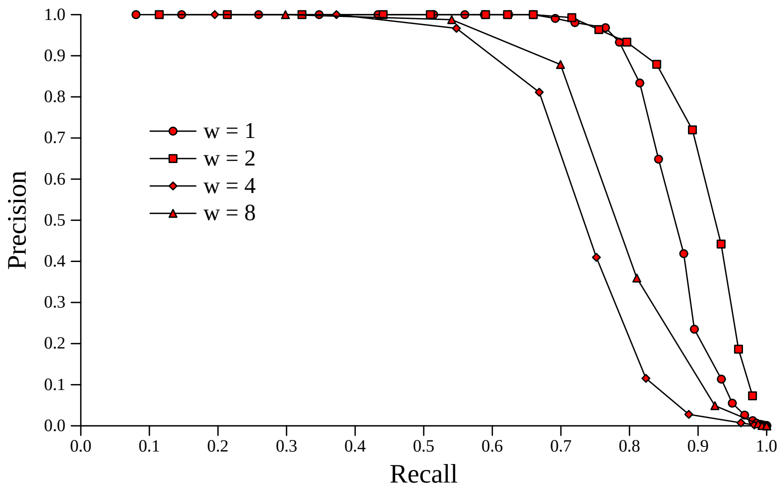


Figure 7.4: Recall-precision curves for various w and GHVs of size 64 bits

by 1, 8 and 4. With $w = 2$ it is possible to achieve recall values of approximately 0.7 before precision moves below 1.

The values represented in Figures 7.3 and 7.4 are probably underestimates of the true effectiveness of GHVs in a distributed environment. The first reason for this is that introducing a hard S_3 score cutoff of 0.58 introduces a distortion: if a pair of documents identified as conditionally equivalent by a GHV happen to have an S_3 score of 0.57, this will count as a false positive. However, such a document pair has a reasonably high likelihood of being perceived as conditionally equivalent if shown to a user. As future work, it would be beneficial to conduct direct user experiments on the effectiveness of GHVs. The outcomes of the user study would allow us to avoid the distortions and inaccuracies introduced by our current use of an indirect metric.

The second reason why the effectiveness of GHVs is probably underestimated in this study is that the perceived effectiveness of GHVs is heavily dependent upon the level of duplication in the underlying collections. For a given parameterisation, one would expect a relatively unvarying quantity of false positives as a result of collisions. However, the higher the level of duplication in the underlying collections, the larger the number of duplicates that will be correctly identified. Thus, for a given level of recall, GHVs will have a higher level of precision on more redundant collections, leading to a recall-precision curve that sits higher above the axis.

In this study we performed our analysis on the AP collection of newswire data, which exhibits a relatively modest amount of duplication compared to many common document domains such as the web. Furthermore, we have not modelled the effect of collection overlap, which would have resulted in a further increase in redundancy in a typical result pool. Thus, one would expect that with the higher level of redundancy present in the envisaged application domain for GHVs, their effectiveness in practice will be superior than suggested by the results presented in this chapter.

Despite these considerations, it is apparent even from these initial experimental results that GHVs are a practical and effective means for screening out all exact duplicate documents and many near-duplicate redundant documents in a cooperative DIR setting. Based on our experiments, we recommend a field width w of two for both 32- and 64-bit GHVs, although it is unclear why this field width consistently outperformed all others that were tested.

7.5 Summary

Management of document redundancy has been cited as one of the major challenges facing the DIR community (Allan et al., 2003). Duplication of documents occurs in DIR due to the same phenomena that cause it to be a problem in centralised IR systems (see Chapter 5), but is compounded by the effect of collection overlap. Despite the seriousness of this problem, it has seen little in-depth investigation, and techniques typically used for redundancy management in centralised IR systems do not translate well to the DIR domain.

In this chapter we have introduced a new data structure, the *grainy hash vector* (GHV), that can be deployed in cooperative DIR systems for efficient and accurate merge-time detection of content-equivalent pairs. GHVs can be used to detect near-duplicates as well as exact duplicates, have well-defined mathematical properties, and are a good fit for a decentralised environment. GHVs can be independently constructed at index time at each site in the DIR system, and only transmitted to the broker at query time. Due to their small size they place little burden on bandwidth. We also present algorithms for efficiently detecting conditionally equivalent document-pairs using GHVs. We demonstrate empirically on the TREC AP collection that GHVs can be used to efficiently and effectively identify conditionally equivalent document pairs at merge time.

Based on the work presented in this chapter, we believe that the GHV is a promising structure for the management of redundancy in cooperative DIR environments. However, they cannot in general be used in an uncooperative situation, as the system relies on the creation and transmission of GHVs by servers on individual sites. Meeting the difficult challenge of effective redundancy management in the more general domain of uncooperative DIR environments remains the subject of future research.

Chapter 8

Conclusions and future work

Redundancy is widespread in current collections of digital data. In this thesis we have studied approaches to defining, detecting, and managing redundancy in the domains of web search, sequence homology search, and cooperative distributed information retrieval. Our research has highlighted the extent of the redundancy problem in these domains, and the inadequacy of current common practice — if any — for its management. We have proposed practical and effective redundancy management techniques in these domains, and demonstrated empirically how they can lead to significant improvements in search-engine behaviour.

8.1 Conclusions

In this section we reiterate the major topics we covered in this thesis, and provide some concluding remarks for each. We discuss our findings and contributions.

8.1.1 Fingerprinting for the discovery problem

There are numerous methods for finding documents that are co-derived with a specified document. However, the management of redundancy in document collections in general requires that we be able to identify all co-derived document pairs without the guidance of a query document. This is the *discovery* problem. At present, only document fingerprinting techniques have been proven feasible for this this problem. Fingerprinting operates by indexing suitable features from documents, and using these features in order to classify document-pairs as co-derivative.

The most popular document fingerprinting algorithms are based on chunking. However, as the volume of features produced by the chunking process is large, only a subset of these features are retained for indexing. The selection algorithm that chooses which chunks to retain has in the past been based on simple heuristics, which leads to a degradation in the overall effectiveness and reliability of the fingerprinting process.

In Chapter 4 we introduced SPEX, a new chunk-selection algorithm that operates on a more principled basis than earlier heuristics. SPEX uses a novel and efficient iterative-hashing process to select only those chunks that contribute to the identification of co-derivation between documents in the collection. This results in an improvement to the tradeoff curve for the fingerprinting process: we can choose equivalent effectiveness for fewer resources, or superior effectiveness for equivalent resources. We also proposed improvements to the overall fingerprinting process and implemented these in DECO, our fingerprinting package.

In experiments, we found that SPEX did significantly reduce chunk-index size on many typical collections. When comparing the effectiveness of the discovery process using SPEX with the commonly-used *modulo* chunking heuristic, we found that SPEX was noticeably more reliable. In particular, with careful parameter selection, SPEX was able to approach 100% accuracy on our experimental testbed.

8.1.2 Redundancy management for web collections

The web offers numerous channels for both *ad hoc* and systematic duplication of content, making it inevitable that a high degree of redundancy exists in web collections. In Chapter 5 we focused on the problem of redundancy management for web search, investigating both the prevalence of redundancy in web collections and suitable ways in which to manage it.

Our user experiments established a correlation between the scores generated by a fingerprinting system and actual user perceptions of redundancy; previously, this correlation had been assumed. The experiments were the catalyst for the formulation of a new class of redundancy — conditional content equivalence — that resulted in increased identification accuracy and a different approach to redundancy management. The user experiments also allowed us to select a sensible threshold to use for classifying document-pairs as redundant.

Having established parameters with the user evaluations, we then analysed two collections crawled from the `.gov` domain on the web: the 18 GB GOV1 collection and the 426 GB GOV2 collection. On the GOV1 collection we found that 17.3% of documents participate in a conditional content equivalence relationship with another document in the collection. We were unable to complete a fingerprinting run of the larger GOV2 collection due to resource limitations, but an analysis using a simpler and less sensitive technique confirmed that the level of redundancy is at least 25% of the entire collection. This analysis uncovered some extraordinarily large clusters of redundant documents, the largest of which consisted of over 500,000 documents. These results highlight the difficulty of crawling the web and confirm the need for effective redundancy management in this domain.

We also analysed the impact of redundancy on ranked result lists returned by search engines. Using the runs submitted to the *ad hoc* search component of the 2004 TREC terabyte track we were able to examine the incidence of conditional content equivalence on search results produced by a wide range of search engines and ranking methodologies. This study confirmed that redundancy was a significant presence in these results.

An important finding in our study was that redundancy within ranked lists was more prevalent amongst those documents that were judged relevant (16.6% redundancy) than it was amongst those ranked non-relevant, for which the rate of redundancy was 14.5%. One consequence of this is that traditional effectiveness measures such as MAP significantly overestimate the actual amenity of these search results. By our estimates, the effectiveness of submissions to the 2004 TREC terabyte track was being overestimated on average by a relative 20%. Another consequence arising from current evaluation methodologies is that any attempt to manage redundancy in search results leads to a decrease in the run's measured evaluation score. The lack of incentive under current evaluation methodologies does much to explain the lack of attention paid to redundancy management in search engine research. This is a significant oversight: we estimate that effective management of redundancy on the ranked lists submitted to the 2004 TREC terabyte track would have resulted in an approximate 16% improvement in true effectiveness.

8.1.3 Redundancy management for distributed information retrieval

Even more than in standard centralised information retrieval, redundancy management in distributed information retrieval systems is a critical issue. Despite this, it has been overlooked. In this context, traditional methods of redundancy management are impractical due to the dispersed nature of the collections. We focus on cooperative distributed information retrieval, in which the servers participating in the search network are able to provide pertinent information to the search broker.

In Chapter 7 we introduced a new data structure, the grainy hash vector, that can be precomputed for each document at its local server, and transmitted along with other document information as part of the server's response to a query. Grainy hash vectors are highly compact, and are designed in such a way that they can be rapidly compared at the broker during the result-merger process. Our experiments show that grainy hash vectors can be used to accurately detect more than half the content-equivalent pairs in a ranked list with a negligible false-positive rate. This makes them the first data structures that are suitable for detection of non-exact duplicates in cooperative distributed information retrieval.

8.1.4 Redundancy management for sequence homology search

In the area of sequence homology search, there is an established orthodoxy in the management of redundancy within collections. Representative-sequence databases (RSDBs) are built by starting with an existing collection, clustering all sequences exceeding a certain identity threshold, and choosing only a single sequence from each cluster to act as its representative in the RSDB. The current approach suffers from two principal problems: the first is that current techniques for RSDB construction suffer from quadratic time complexity; the second is that RSDBs — though they lead to faster search times, eliminate tiresome repetition from ranked lists, and improve performance in certain specialised tools — generally cause a degradation in overall search effectiveness. In Chapter 6 we presented new techniques that largely solve both of these problems.

We investigated the use of document fingerprinting to identify pairs of sequences with high identity in order to obviate the need for direct pairwise comparison. We introduced slotted SPEX, a variant of the SPEX algorithm that is tailored to the data characteristics of this domain. An evaluation of the ranking produced by our modified fingerprinting program using the slotted SPEX algorithm alongside a list of sequence-pairs ranked by actual identity values showed a strong correlation between the two, thus suggesting that fingerprinting is an appropriate choice for building RSDBs. This intuition was confirmed by the creation of a program based on this approach which was able to construct a RSDB from a 900MB sequence collection in less than $1\frac{1}{2}$ hours, more than six times faster than existing approaches. Using document fingerprinting, we were able to flatten the curve for RSDB construction time to one that is approximately linear for the data sizes tested.

In order to address the problem of decreased search effectiveness incurred by use of RSDBs, we introduced a more sophisticated approach to the management of redundancy in genomic collections. In contrast with the RSDB approach, our strategy is non-destructive, as all sequences are retained in the collection. As with the creation of RSDBs, we begin by identifying clusters of sequences with a high degree of identity. However, instead of discarding all but one of the sequences, we make use of wildcards to create a single 'union-sequence' that is able to simultaneously represent all sequences in the cluster.

We described practical measures for clustering, wildcard selection, wildcard scoring, and search using our new representation. Our approach proved effective at managing redundancy and brought significant benefits. The clustered representation resulted in a 22% decrease in search time on the GenBank NR database with no significant reduction in search effectiveness. As an additional benefit, our representation achieved a form of compression: the on-disk footprint of our clustered collection was 27% smaller than the standard representation.

8.2 Future work

In this thesis, we have introduced an improved mechanism for discovering redundancy in document collections. We have analysed the impact of redundancy in several domains, and proposed practical approaches to redundancy management in these domains. We have demonstrated that our redundancy management strategies lead to real improvements in the performance and effectiveness of information retrieval systems. However, there is still scope for many improvements and extensions to the work presented in this thesis. We discuss a range of potential areas for future research below.

8.2.1 Improved scalability of SPEX

The intended purpose of SPEX was to improve the scalability of document fingerprinting by identifying and eliminating extraneous information before it had to be stored and processed. It was successful at this task, and the ease with which the 18 GB GOV1 collection was processed by SPEX demonstrated that it had reasonable scalability characteristics. However, we have not successfully used SPEX to process the 426 GB GOV2 collection, due more to the characteristics of the collection than its sheer size: the high degree of internal redundancy in GOV2 meant that there was less extraneous information to be eliminated in this collection. This highlights the fact that SPEX becomes decreasingly effectual as a collection becomes increasingly redundant. On a collection in which every document is duplicated, SPEX is completely ineffectual and its use represents a waste of resources.

Consequently, it is necessary to investigate ways of making the SPEX algorithm more scalable and more robust with respect to variations in collection characteristics. One possibility is to attempt to make use of lossy variants such as slotted SPEX, although it is unclear how useful it would be in the natural-language domain. If it is not possible to improve the characteristics of SPEX, then it is necessary to look at alternative selection algorithms that take a more robust approach to chunk selection than current alternatives.

8.2.2 Spex for dynamic collections

In its present form, SPEX is a 'one-shot' algorithm: it must process the collection in its entirety. If the composition of the collection changes at any point — particularly if new documents are added — SPEX must be run again on the entire collection. Thus, it is not at present suitable for dynamic collections such as those that may be maintained by web search engines or enterprises. It would be worthwhile to investigate the use of auxiliary data structures and algorithms to allow SPEX to be used with dynamic collections.

8.2.3 Improved algorithms for the discovery problem

Another area in which the issue of scalability must be addressed is in the processing of fingerprint indexes for the discovery problem. In Section 3.9 we discussed the problem of the quadratic cost of postings-list expansion, and described existing solutions in some depth. However, none of these solutions is particularly satisfactory in the general case. One still must choose between a faithful but potentially intractable attempt to reproduce the relationship graph, and a more efficient solution that grossly simplifies the nature of the co-derivation relation. A solution that combines the fidelity of the former approaches with the efficiency of the latter is a goal for future work.

In Chapter 6 we described a process that combined top-down and bottom-up processing of the fingerprint index to build up clusters of sequences with high mutual identity. While our algorithm was tailored for the particular application of building clusters that could be represented by union-sequences with few wildcard substitutions, the approach can plausibly be generalised. As a heuristic for efficient processing of indexes that still captures much of the complexity inherent to a relationship graph, a modification of our clustering approach merits further investigation.

8.2.4 Additional user experiments

In Chapter 5 our series of user experiments correlated scores obtained on DECO to content equivalence and conditional content equivalence. These experiments were valuable in allowing us to make convincing and valid assertions about the degree of redundancy in ranked lists. There are several areas in the thesis that would have benefited from further user evaluations.

We made the claim in Chapter 5 that removing content-equivalent results from ranked lists would have resulted in an estimated 16% improvement in search effectiveness. This estimate is based on a double abstraction: first, that a document-pair exceeding a given score in DECO must necessarily be content-equivalent, and second, that MAP corresponds directly to search effectiveness. The claim of improved effectiveness would have carried more force if a direct user evaluation of this quantity had been performed.

In Chapter 7 we face a similar level of double-indirection when evaluating the effectiveness of the grainy hash vector data structure. Rather than directly capturing the effectiveness of grainy hash vectors in identifying content equivalence, we compared the outcome of using grainy hash vectors with the output produced by DECO. This resulted in a probable underestimate of the true performance of grainy hash vectors; a user study would allow us a more accurate estimate of the true effectiveness of these data structures.

8.2.5 Redundancy management for distributed information retrieval

In Chapter 7 we proposed the grainy hash vector as a data structure that could be used by cooperative distributed information retrieval systems in order to manage inter-collection redundancy. However — save for the unlikely scenario that grainy hash vectors become a standard component of all search engines — the approach is not possible for the more general problem of non-cooperative distributed information retrieval. Thus, an investigation of possible techniques for redundancy management in non-cooperative distributed information retrieval is a logical extension of our existing work. This problem is a difficult one. In the non-cooperative scenario, the broker must intelligently and efficiently manage redundancy without having direct access to collections and with only minimal descriptive information regarding the documents being managed.

8.3 Final Remarks

Advances in technology have put publication of documents into the hands of individuals, and have simultaneously made duplication of data extremely easy. The combination of these two phenomena means that redundancy of information has become pervasive in modern document corpora. In this thesis, we have contributed new techniques for identifying redundancies within large collections. We have used these tools to gain insight into the nature and impact of redundancy on information retrieval in a number of domains, and have used these insights to propose novel ways in which redundancy can be managed in order to improve the information retrieval process.

Bibliography

R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Conf. on Very Large Data Bases*, pages 487–499, Santiago de Chile, Chile, 1994. Morgan Kaufmann Publishers Inc.

J. Allan, C. Wade, and A. Bolivar. Retrieval and novelty detection at the sentence level. In C. Clarke, G. Cormack, J. Callan, D. Hawking, and A. Smeaton, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 314–321, Toronto, Canada, 2003. ACM Press.

J. Allan et al. Challenges in information retrieval and language modeling: report of a workshop held at the Center for Intelligent Information Retrieval, University of Massachusetts Amherst, September 2002. *SIGIR Forum*, 37(1):31–47, 2003.

S. Altschul and W. Gish. Local alignment statistics. *Methods in Enzymology*, 266:460–480, 1996.

S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI–BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

A. Andreeva, D. Howorth, S. Brenner, T. Hubbard, C. Chothia, and A. Murzin. SCOP database in 2004: refinements integrate structure and sequence family data. *Nucleic Acids Research*, 32:D226–D229, 2004.

J. A. Aslam and E. Yilmaz. A geometric interpretation and analysis of r-precision. In A. Chowdhury, N. Fuhr, M. Ronthaler, H. Schek, and W. Teiken, editors, *Proc. CIKM*

Int. Conf. on Information and Knowledge Management, pages 664–671, Bremen, Germany, 2005. ACM Press.

L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. In *Readings in Speech Recognition*, pages 308–319. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

C. Bennett, P. Gacs, M. Li, P. Vitányi, and W. Zurek. Information distance. *IEEE Transactions on Information Theory*, 44, 1998.

C. Bennett, M. Li, and B. Ma. Chain letters and evolutionary histories. *Scientific American*, pages 76–81, June 2003.

D. Benson, I. Karsch-Mizrachi, D. Lipman, J. Ostell, and D. Wheeler. Genbank. *Nucleic Acids Research*, 33:D34–D38, 2005.

A. Berger and J. Lafferty. Information retrieval as statistical translation. In F. Gey, M. Hearst, and R. Tong, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 222–229, Berkeley, California, United States, 1999. ACM Press. doi: http://doi.acm.org/10.1145/312624.312681.

Y. Bernstein and M. Cameron. Fast discovery of similar sequences in large genomic collections. In M. Lalmas, A. MacFarlane, S. M. Rüger, A. Tombros, T. Tsikrika, and A. Yavlinsky, editors, *Proc. ECIR European Conf. on IR Research*, pages 432–443, London, United Kingdom, 2006.

Y. Bernstein and J. Zobel. A scalable system for identifying co-derivative documents. In A. Apostolico and M. Melucci, editors, *Proc. SPIRE String Processing and Information Retrieval Symp.*, pages 55–67, Padova, Italy, Sep 2004. Springer.

Y. Bernstein and J. Zobel. Redundant documents and search effectiveness. In A. Chowdhury, N. Fuhr, M. Ronthaler, H. Schek, and W. Teiken, editors, *Proc. CIKM Int. Conf. on Information and Knowledge Management*, pages 736–743, Bremen, Germany, Oct 2005. ACM Press.

Y. Bernstein and J. Zobel. Accurate discovery of co-derivative documents via duplicate text detection. *Information Systems*, 2006. To appear.

Y. Bernstein, M. Shokouhi, and J. Zobel. Compact features for detection of near-duplicates in distributed retrieval. In *Proc. SPIRE String Processing and Information Retrieval Symp.*, Glasgow, United Kingdom, 2006. To appear.

A. J. Bleasby and J. C. Wootton. Construction of validated, non-redundant composite protein sequence databases. *Protein Engineering*, 3(3):153–159, 1990.

B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970. ISSN 0001-0782.

S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proc. World-Wide Web Conference*, pages 107–117, Brisbane, Australia, 1998.

S. Brin, J. Davis, and H. García-Molina. Copy detection mechanisms for digital documents. In M. J. Carey and D. A. Schneider, editors, *Proc. ACM-SIGMOD Int. Conf. on the Management of Data*, pages 398–409, San Jose, California, 1995.

A. Z. Broder. Some applications of Rabin's fingerprinting method. In *Sequences II: Methods in Communications, Security, and Computer Science*, pages 143–152, 1993.

A. Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences (SEQUENCES'97)*, pages 21–29, Positano, Italy, 1997.

A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166, 1997.

A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations (extended abstract). In *Proc. ACM symposium on Theory of computing (STOC)*, pages 327–336, Dallas, Texas, 1998. ACM Press.

P. F. Brown, J. Cocke, S. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, June 1990.

C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In E. Yannakoudakis, N. J. Belkin, M.-K. Leong, and P. Ingwersen, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 33–40, Athens, Greece, 2000. ACM Press.

C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In M. Sanderson, K. Järvelin, J. Allan, and P. Bruza, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 25–32, Sheffield, United Kingdom, 2004. ACM Press.

J. Burke, D. Davison, and W. Hide. d2_cluster: A validated method for clustering EST and full-length DNA sequences. *Genome Research*, 9(11):1135–1142, 1999.

J. Callan. Distributed information retrieval. In W. B. Croft, editor, *Advances in Information Retrieval*, pages 127–150. Kluwer Academic Publishers, 2000.

J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

J. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington, 1995.

M. Cameron, H. E. Williams, and A. Cannane. Improved gapped alignment in BLAST. *IEEE Transactions on Computational Biology and Bioinformatics*, 1(3):116–129, 2004.

M. Cameron, Y. Bernstein, and H. E. Williams. Clustering near-identical sequences for fast homology search. In A. Apostolico, C. Guerra, S. Istrail, P. A. Pevzner, and M. S. Waterman, editors, *Proc. International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 175–189, Venice, Italy, 2006a.

M. Cameron, Y. Bernstein, and H. E. Williams. Clustered sequence representation for fast homology search. *Journal of Computational Biology*, 2006b. In submission.

M. Cameron, H. E. Williams, and A. Cannane. A deterministic finite automaton for faster protein hit detection in BLAST. *Journal of Computational Biology*, 13(4):965–978, 2006c. To appear.

D. M. Campbell, W. R. Chen, and R. D. Smith. Copy detection systems for digital documents. In *Proc. IEEE Advances in Digital Libraries (ADL)*, pages 78–88, Washington, D.C., 2000. IEEE Computer Society.

J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In W. B. Croft, A. Moffat, C. J. van Rijsbergen,

R. Wilkinson, and J. Zobel, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 335–336, Melbourne, Australia, 1998. ACM Press.

J. L. Carter and M. N. Wegman. Universal classes of hash functions (extended abstract). In *Proc. ACM symposium on Theory of computing (STOC)*, pages 106–112, Boulder, Colorado, 1977. ACM Press. doi: http://doi.acm.org/10.1145/800105.803400.

J. Chandonia, G. Hon, N. Walker, L. L. Conte, P. Koehl, M. Levitt, and S. Brenner. The ASTRAL compendium in 2004. *Nucleic Acids Research*, 32:D189–D192, 2004.

K. Chao, W. Pearson, and W. Miller. Aligning two sequences within a specified diagonal band. *Computer Applications in the Biosciences*, 8(5):481–487, 1992.

X. Chen, M. Li, B. McKinnon, and A. Seker. A theory of uncheatable program plagiarism detection and its practical implementation. Manuscript., May 2002.

X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker. Shared information and program plagiarism detection. *IEEE Transactions on Information Theory*, 50:1545–1551, 2004.

C.-F. Cheung, J. X. Yu, and H. Lu. Constructing suffix tree for gigabyte sequences with megabyte memory. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):90–105, 2005.

J. Cho, N. Shivakumar, and H. Garcia-Molina. Finding replicated web collections. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proc. ACM-SIGMOD Int. Conf. on the Management of Data*, pages 355–366, Dallas, Texas, 2000.

A. Chowdhury, O. Frieder, D. Grossman, and M. C. McCabe. Collection statistics for fast duplicate document detection. *ACM Transactions on Information Systems*, 20(2):171–191, 2002.

C. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2004 Terabyte Track. In E. M. Voorhees and L. P. Buckland, editors, *Proc. Text Retrieval Conf. (TREC)*, 2004.

C. W. Cleverdon. The significance of the Cranfield tests on index languages. In A. Bookstein, Y. Chiaramella, G. Salton, and V. V. Raghavan, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 3–12, Chicago, Illinois, 1991. ACM Press.

P. Clough, R. Gaizauskas, S. Piao, and Y. Wilks. METER: MEasuring TExt Reuse. In *Proc. 40th Anniversary Meeting for the Association for Computational Linguistics*, pages 152–159, Philadelphia, Pennsylvania, 2002. Association for Computational Linguistics.

J. G. Conrad, X. S. Guo, and C. P. Schriber. Online duplicate document detection: Signature reliability in a dynamic retrieval environment. In O. Frieder, J. Hammer, S. Quershi, and L. Seligman, editors, *Proc. CIKM Int. Conf. on Information and Knowledge Management*, pages 443–452, New Orleans, Louisiana, 2003. ACM Press.

J. W. Cooper, A. R. Coden, and E. W. Brown. Detecting similar documents using salient terms. In C. Nicholas, D. Grossman, K. Kalpakis, S. Qureshi, H. van Dissel, and L. Seligman, editors, *Proc. CIKM Int. Conf. on Information and Knowledge Management*, pages 245–251, McLean, VA, 2002. ACM Press.

G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. Efficient construction of large test collections. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 282–289, Melbourne, Australia, 1998. ACM Press.

N. Craswell and D. Hawking. Overview of the TREC-2002 Web Track. In *Proc. Text Retrieval Conf. (TREC)*, Gaithersburg, Maryland, 2002.

N. Craswell, D. Hawking, and P. Thistlewaite. Merging results from isolated search engines. In *Proc. Australasian Database Conf.*, pages 189–200, Auckland, NZ, 1999.

N. Craswell, D. Hawking, and S. Robertson. Effective site finding using link anchor information. In D. H. Kraft, W. B. Croft, D. J. Harper, and J. Zobel, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 250–257, New Orleans, Louisiana, 2001. ACM Press. doi: http://doi.acm.org/10.1145/383952.383999.

W. B. Croft and D. J. Harper. Using probabilistic models of document retrieval without relevance information. In *Readings in information retrieval*, pages 339–344. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN 1-55860-454-5.

F. J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/363958.363994.

M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of protein sequence and structure*, 5:345–358, 1978.

F. Diaz. Regularizing ad hoc retrieval scores. In A. Chowdhury, N. Fuhr, M. Ronthaler, H. Schek, and W. Teiken, editors, *Proc. CIKM Int. Conf. on Information and Knowledge Management*, pages 672–679, Bremen, Germany, 2005. ACM Press.

M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. In A. Gupta, O. Shmueli, and J. Widom, editors, *Proc. VLDB Int. Conf. on Very Large Databases*, pages 299–310, New York, New York, 1998. Morgan Kaufmann Publishers Inc.

P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 390–398, Redondo Beach, California, 2000. IEEE Computer Society.

D. Fetterly, M. Manasse, and M. Najork. On the evolution of clusters of near-duplicate web pages. In R. Baeza-Yates, D. Schwabe, J. Piquer, N. Ziviani, and L. Olsina, editors, *Proc. Latin American Web Congress (LA-WEB)*, pages 37–45, Sanitago, Chile, 2003. IEEE.

R. A. Finkel, A. Zaslavsky, K. Monostori, and H. Schmidt. Signature extraction for overlap detection in documents. In M. J. Oudshoorn, editor, *Proc. Australasian Computer Science Conf.*, Melbourne, Australia, 2002.

T. J. Froehlich. Relevance reconsidered– towards an agenda for the 21st century: introduction to special topic issue on relevance research. *Jour. of the American Society for Information Science*, 45(3):124–134, 1994. ISSN 0002-8231. doi: http://dx.doi.org/10.1002/(SICI)1097-4571(199404)45:3⟨124::AID-ASI2⟩3.0.CO;2-8.

S. Garcia, H. E. Williams, and A. Cannane. Access-ordered indexes. In V. Estivill-Castro, editor, *Proc. Australasian Computer Science Conf.*, pages 7–14, Dunedin, New Zealand, 2004.

S. Gauch, G. Wang, and M. Gomez. ProFusion: Intelligent fusion from multiple, distributed search engines. *Journal of Universal Computer Science*, 2(9):637–649, 1996.

H. T. Glantz. On the recognition of information with a digital computer. *Jour. of the ACM*, 4(2):178–188, 1957. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/320868.320878.

J. Gracy and P. Argos. Automated protein sequence database classification. I. Integration of compositional similarity search, local similarity search, and multiple sequence alignment. *Bioinformatics*, 14(2):164–173, 1998.

M. Gribskov and N. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers & Chemistry*, 20:25–33, 1996.

M. Gribskov, A. D. McLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences (PNAS)*, 84(13):4355–4358, 1987.

G. Grillo, M. Attimonelli, S. Liuni, and G. Pesole. CLEANUP: a fast computer program for removing redundancies from nucleotide sequence databases. *CABIOS*, 12(1):1–8, 1996.

P. Grünwald and P. Vitányi. Shannon information and Kolmogorov complexity. *IEEE Transactions on Information Theory*, 2006. In submission.

D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press, 1997.

D. Harman. Overview of the first TREC conference. In R. Korfhage, E. M. Rasmussen, and P. Willett, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 36–47, Pittsburgh, Pennsylvania, 1993.

D. Harman. Overview of the second text retrieval conference (TREC-2). *Information Processing & Management*, 31(3):271–289, 1995.

D. Harman. Overview of the TREC 2002 Novelty Track. In *Proc. Text Retrieval Conf. (TREC)*, 2002.

T. H. Haveliwala, A. Gionis, and P. Indyk. Scalable techniques for clustering the web. In D. Suciu and G. Vossen, editors, *Proc. WebDB Workshop*, pages 129–134, Dallas, Texas, 2000.

M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 76–84, Zürich, Switzerland, 1996. ACM Press.

P. Heckel. A technique for isolating differences between files. *Communications of the ACM*, 21(4):264–268, 1978. ISSN 0001-0782.

N. Heintze. Scalable document fingerprinting. In *1996 USENIX Workshop on Electronic Commerce*, pages 191–200, Oakland, California, November 1996.

S. Heinz and J. Zobel. Efficient single-pass index construction for text databases. *Jour. of the American Society for Information Science and Technology*, 54(8):713–729, 2003. ISSN 1532-2882. doi: http://dx.doi.org/10.1002/asi.10268.

S. Henikoff and J. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences USA*, 89(22):10915–10919, 1992.

T. Hernandez and S. Kambhampati. Improving text collection selection with coverage and overlap statistics. In A. Ellis and T. Hagino, editors, *Proc. WWW Conference*, pages 1128–1129, Chiba, Japan, 2005.

T. C. Hoad and J. Zobel. Methods for identifying versioned and plagiarised documents. *Jour. of the American Society for Information Science and Technology*, 54(3):203–215, 2003.

L. Holm and C. Sander. Removing near-neighbour redundancy from large protein sequence collections. *Bioinformatics*, 14(5):423–429, 1998.

J. W. Hunt and M. D. McIlroy. An algorithm for differential file comparison. Technical Report CSTR 41, Bell Telephone Laboratories, 1976. URL `http://www.cs.dartmouth.edu/~doug/`.

S. Ilyinski, M. Kuzmin, A. Melkov, and I. Segalovich. An efficient method to detect duplicates of web documents with the use of inverted index. In D. Lassner, D. D. Roure, and A. Iyengar, editors, *Proc. World-Wide Web Conference*, Honolulu, Hawaii, 2002.

P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. ACM symposium on Theory of computing (STOC)*, pages 604–613, Dallas, Texas, 1998. ACM Press.

M. Itoh, T. Akutsu, and M. Kanehisa. Clustering of database sequences for fast homology search using upper bounds on alignment score. *Genome Informatics*, 15(1):93–104, 2004.

N. Jardine and C. J. van Rijsbergen. The use of hierarchical clustering in information retrieval. *Information Storage and Retrieval*, 7:217–240, 1971.

S. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

Y. Kallberg and B. Persson. KIND — a non-redundant protein database. *Bioinformatics*, 15(3):260–261, 1999.

S. Karlin and S. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences USA*, 87(6):2264–2268, 1990.

R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. *IBM J. Res. Dev.*, 31(2):249–260, 1987. ISSN 0018-8646.

J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Jour. of the ACM*, 46 (5):604–632, 1999. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/324133.324140.

A. Kolcz, A. Chowdhury, and J. Alspector. Improved robustness of signature-based near-replica detection via lexicon randomization. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *Proc. ACM KDD conference*, pages 605–610, Seattle, Washington, 2004. ACM Press.

S. Kurtz. Reducing the space requirement of suffix trees. *Softw. Pract. Exper.*, 29(13):1149–1171, 1999. ISSN 0038-0644. doi: http://dx.doi.org/10.1002/(SICI)1097-024X(199911)29:13⟨1149::AID-SPE274⟩3.0.CO;2-O.

N. J. Larsson and A. Moffat. Offline dictionary-based compression. *Proceedings of the IEEE*, 88(11):1722–1732, Nov 2000.

V. Lavrenko. *A generative theory of relevance.* PhD thesis, University of Massachusetts Amherst, 2004.

M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer, 1997.

M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. In *Proc. Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, Maryland, 2003. Society for Industrial and Applied Mathematics.

W. Li, L. Jaroszewski, and A. Godzik. Tolerating some redundancy significantly speeds up clustering of large protein databases. *Bioinformatics*, 18(1):77–82, 2001a.

W. Li, L. Jaroszewski, and A. Godzik. Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, 17(3):282–283, 2001b.

W. Li, L. Jaroszewski, and A. Godzik. Sequence clustering strategies improve remote homology recognitions while reducing search times. *Protein Engineering*, 15(8):643–649, 2002.

X. Liu and W. B. Croft. Cluster-based retrieval using language models. In M. Sanderson, K. Järvelin, J. Allan, and P. Bruza, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 186–193, Sheffield, United Kingdom, 2004. ACM Press. doi: http://doi.acm.org/10.1145/1008992.1009026.

C. Lyon, J. Malcolm, and B. Dickerson. Detecting short passages of similar text in large document collections. In L. Lee, D. Harman, and D. Yarowsky, editors, *Proc. Conference on Empirical Methods in Natural Language Processing*, Philadelphia, Pennsylvania, USA, 2001.

K. Malde, E. Coward, and I. Jonassen. Fast sequence clustering using a suffix array algorithm. *Bioinformatics*, 19(10):1221–1226, 2003.

U. Manber. Finding similar files in a large file system. In *Proceedings of the USENIX Winter 1994 Technical Conference*, pages 1–10, San Fransisco, California, 17–21 1994.

U. Manber and G. Myers. Suffix arrays: a new method for on-line string searches. *SIAM Jour. of Computing*, 22(5):935–948, 1993.

M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *Jour. of the ACM*, 7(3):216–244, 1960. ISSN 0004-5411.

W. Meng, C. Yu, and K. Liu. Building efficient and effective metasearch engines. *ACM Computing Surveys*, 34(1):48–89, 2002.

D. A. Metzler, Y. Bernstein, W. B. Croft, A. Moffat, and J. Zobel. Similarity measures for tracking information flow. In A. Chowdhury, N. Fuhr, M. Ronthaler, H. Schek, and W. Teiken, editors, *Proc. CIKM Int. Conf. on Information and Knowledge Management*, pages 517–524, Bremen, Germany, Oct 2005. ACM Press.

S. Mizzaro. Relevance: the whole history. *J. Am. Soc. Inf. Sci.*, 48(9):810–832, 1997. ISSN 0002-8231. doi: http://dx.doi.org/10.1002/(SICI)1097-4571(199709)48:9⟨810::AID-ASI6⟩3.0.CO;2-U.

S. Mizzaro. How many relevances in information retrieval? *Interacting with Computers*, 10 (3):303–320, 1998.

A. Moffat and R. Wan. Re-Store: A system for compressing, browsing, and searching large documents. In *Proceedings of the International Symposium on String Processing and Information Retrieval*, pages 162–174, Laguna de San Rafael, Chile, 2001. IEEE Computer Society.

A. Murzin, S. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4):536–540, 1995.

E. W. Myers. An overview of sequence comparison algorithms in molecular biology. Technical Report 91–29, University of Arizona, Department of Computer Science, Dec 1991.

G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33 (1):31–88, 2001. ISSN 0360-0300.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

C. G. Nevill-Manning and I. H. Witten. Compression and explanation using hierarchical grammars. *The Computer Journal*, 40(2/3):103–116, 1997.

C. G. Nevill-Manning and I. H. Witten. Protein is incompressible. In *DCC '99: Proceedings of the Conference on Data Compression*, page 257, Snowbird, Utah, 1999. IEEE Computer Society.

C. G. Nevill-Manning, I. H. Witten, and G. W. Paynter. Browsing in digital libraries: a phrase-based approach. In *Proc. ACM Digital Libraries*, pages 230–236. ACM Press, 1997.

H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In C. Clarke, G. Cormack, J. Callan, D. Hawking, and A. Smeaton, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 290–297, Toronto, Canada, 2003.

J. Park, L. Holm, A. Heger, and C. Chothia. RSDB: representative sequence databases have high information content. *Bioinformatics*, 16(5):458–464, 2000.

J. D. Parsons. Improved tools for DNA comparison and clustering. *CABIOS*, 11(6):603–613, 1995.

W. Pearson and D. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences USA*, 85(8):2444–2448, 1988.

Á. R. Pereira Jr. and N. Ziviani. Syntactic similarity of web documents. In R. Baeza-Yates, D. Schwabe, J. Piquer, N. Ziviani, and L. Olsina, editors, *Proc. Latin American Web Congress (LA-WEB)*, pages 194–200, Sanitago, Chile, 2003. IEEE Computer Society.

J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia, 1998. ACM Press.

W. Pugh and M. H. Henzinger. Detecting duplicate and near-duplicate files (United States Patent 6,658,423), 2003.

M. Rabin. Fingerprinting by random polynomials. Technical Report TR-15-81, Center for Research in Computing Technology, Harvard University, 1981.

M. V. Ramakrishna and J. Zobel. Performance in practice of string hashing functions. In *Proc. Databases Systems for Advanced Applications Symposium*, pages 215–223, Melbourne, Australia, Apr 1997.

R. Rivest. The MD5 message-digest algorithm, Apr 1992. RFC 1321.

S. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Jour. of the American Society for Information Science*, 27(3):129–146, 1976.

S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. Text Retrieval Conf. (TREC)*, pages 109–126, 1995.

S. E. Robertson. The probability ranking principle in IR. *Readings in Information Retrieval*, pages 281–286, 1997.

A. Robinson and L. Robinson. Distribution of glutamine and asparagine residues and their near neighbors in peptides and proteins. *Proceedings of the National Academy of Sciences USA*, 88(20):8880–8884, 1991.

J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System – Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, Englewood Cliffs, NJ, 1971.

I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2):95–145, 2003. ISSN 0269-8889.

G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, NY, USA, 1986.

M. Sanderson. Duplicate detection in the Reuters collection. Technical Report TR-1997-5, University of Glasgow, 1997.

M. Sanderson and J. Zobel. Information retrieval system evaluation: Effort, sensitivity, and reliability. In A. Moffat, G. Marchionini, J. Tate, R. Baeza-Yates, and N. Ziviani, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 162–169, Salvador, Brazil, 2005.

T. Saracevic. Relevance: A review of and a framework for the thinking on the notion in information science. *Jour. of the American Society for Information Science*, 26(6):321–343, 1975.

T. Saracevic. Evaluation of evaluation in information retrieval. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 138–146, Seattle, Washington, 1995. ACM Press.

D. V. Sarwate. A note on universal classes of hash functions. *Information Processing Letters*, 10(1), 1980.

A. Schaffer, L. Aravind, T. Madden, S. Shavirin, J. Spouge, Y. Wolf, E. Koonin, and S. Altschul. Improving the accuracy of PSI–BLAST protein database searches with composition-based statistics and other refinements. *Nucleic Acids Research*, 29(14):2994–3005, 2001.

S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: local algorithms for document fingerprinting. In A. Y. Halevy, Z. G. Ives, and A. Doan, editors, *Proc. ACM-SIGMOD Int. Conf. on the Management of Data*, pages 76–85, San Diego, California, 2003. ACM Press.

R. Sedgewick. *Algorithms in C.* Addison Wesley, 1998.

E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert*, 12(1):8–14, 1997.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948.

N. Shivakumar and H. García-Molina. SCAM: A copy detection mechanism for digital documents. In *Proc. Conference on the Theory and Practice of Digital Libraries*, Austin, Texas, 1995.

N. Shivakumar and H. García-Molina. Finding near-replicas of documents on the web. In P. Atzeni, A. O. Mendelzon, and G. Mecca, editors, *WEBDB: International Workshop on the World Wide Web and Databases*, Valencia, Spain, 1998. Springer-Verlag.

L. Si and J. Callan. Unified utility maximization framework for resource selection. In D. Grossman, L. Gravano, C. Zhai, O. Herzog, and D. A. Evans, editors, *Proc. CIKM Int. Conf. on Information and Knowledge Management*, pages 32–41, Washington, D.C., 2004.

L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In C. Clarke, G. Cormack, J. Callan, D. Hawking, and A. Smeaton, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 298–305, Toronto, Canada, 2003a.

L. Si and J. Callan. A semisupervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 21(4):457–491, 2003b.

A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 21–29, Zürich, Switzerland, 1996. ACM Press. ISBN 0-89791-792-8.

T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

I. Soboroff. Overview of the TREC 2004 Novelty Track. In *Proc. Text Retrieval Conf. (TREC)*, 2004.

I. Soboroff and D. Harman. Overview of the TREC 2003 Novelty Track. In *Proc. Text Retrieval Conf. (TREC)*, pages 38–53, 2003.

K. Sparck Jones and C. van Rijsbergen. Report on the need for and provision of an "ideal" information retrieval test collection. Technical report, British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge, 1975.

K. Sparck Jones, S. Walker, and S. Robertson. A probabilistic model of information retrieval: development and comparative experiments, Parts I and II. *Information Processing & Management*, 36(6):779–840, 2000.

S. Tata, R. A. Hankins, and J. M. Patel. Practical suffix tree construction. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *Proc. VLDB Int. Conf. on Very Large Databases*, pages 36–47, Toronto, Canada, 2004.

W. Taylor. The classification of amino-acid conservation. *Journal of Theoretical Biology*, 119:205–218, 1986.

W. F. Tichy. The string-to-string correction problem with block moves. *ACM Trans. Comput. Syst.*, 2(4):309–321, 1984. ISSN 0734-2071.

W. F. Tichy. RCS — a system for version control. *Softw. Pract. Exper.*, 15(7):637–654, 1985. ISSN 0038-0644.

C. J. van Rijsbergen. *Information Retrieval.* Butterworth-Heinemann, 1979.

J.-S. Varre, J.-P. Delahaye, and É. Rivals. The transformation distance: A dissimilarity measure based on movements of segments. In *Proc. German Conference on Bioinformatics*, Cologne, Germany, 1998.

E. M. Voorhees. The cluster hypothesis revisited. In *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 188–196, Montreal, Canada, 1985. ACM Press.

E. M. Voorhees. The TREC robust retrieval track. *SIGIR Forum*, 39(1):11–20, 2005. ISSN 0163-5840. doi: http://doi.acm.org/10.1145/1067268.1067272.

E. M. Voorhees and C. Buckley. The effect of topic set size on retrieval experiment error. In K. Järvelin, M. Beaulieu, R. Baeza-Yates, and S. H. Myaeng, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 316–323, Tampere, Finland, 2002. ACM Press.

H. S. Warren, Jr. *Hacker's Delight.* Addison Wesley, 2002.

O. Weiss, M. Jimenez-Montano, and H. Herzel. Information content of protein sequences. *Journal of Theoretical Biology*, 206(3):379–386, 2000.

H. E. Williams and J. Zobel. Indexing and retrieval for genomic databases. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):63–78, 2002.

M. J. Wise. String similarity via greedy string tiling and running Rabin-Karp matching. Technical report, University of Sydney, 1993.

M. J. Wise. Yap3: improved detection of similarities in computer program and other texts. In J. Impagliazzo, E. S. Adams, and K. J. Klee, editors, *Proc. technical symposium on Computer science education (SIGCSE)*, pages 130–134, Philadelphia, Pennsylvania, 1996. ACM Press.

I. H. Witten and T. C. Bell. Source models for natural language text. *Int. Jour. Man-Machine Studies*, 32:545–579, 1990.

I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Morgan Kauffman, 1999.

S. K. M. Wong and Y. Y. Yao. A probability distribution model for information retrieval. *Inf. Process. Manage.*, 25(1):39–53, 1989. ISSN 0306-4573. doi: http://dx.doi.org/10.1016/0306-4573(89)90090-3.

O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to web search results. In *Proc. World-Wide Web Conference*, pages 1361–1374, Toronto, Canada, 1999.

C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004. ISSN 1046-8188. doi: http://doi.acm.org/10.1145/984321.984322.

C. X. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In C. Clarke, G. Cormack, J. Callan, D. Hawking, and A. Smeaton, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 10–17, Toronto, Canada, 2003. ACM Press.

Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In K. Järvelin, M. Beaulieu, R. Baeza-Yates, and S. H. Myaeng, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 81–88, Tampere, Finland, 2002. ACM Press.

J. Zobel. How reliable are the results of large-scale information retrieval experiments? In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proc. ACM-SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 307–314, Melbourne, Australia, Aug. 1998.

J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 2006. To appear.