THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Zeynep PEHLIVAN

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

ACCESS TO WEB ARCHIVES
Querying, Navigating and Optimizing

ACCES AUX ARCHIVES WEB
Interrogation, Navigation et Optimisation

---

soutenue le 11 Octobre 2013, devant le jury composé de :

Sihem AMER-YAHIA	Rapporteur	CNRS - Laboratoire d'Informatique de Grenoble
Arjen P. DE VRIES	Rapporteur	Université Delft
François BANCILHON	Examinateur	DataPublica
Matthieu CORD	Examinateur	UPMC Paris 6
David GROSS-AMBLARD	Examinateur	Université de Rennes 1
Pierre SENELLART	Examinateur	Télécom ParisTech
Anne DOUCET	Directeur de thèse	UPMC Paris 6
Stéphane GANÇARSKI	Co-directeur de thèse	UPMC Paris 6

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Zeynep PEHLIVAN

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

ACCESS TO WEB ARCHIVES
Querying, Navigating and Optimizing

ACCES AUX ARCHIVES WEB
Interrogation, Navigation et Optimisation

soutenue le 11 Octobre 2013, devant le jury composé de :

Sihem AMER-YAHIA	Rapporteur	CNRS - Laboratoire d'Informatique de Grenoble
Arjen P. DE VRIES	Rapporteur	Université Delft
François BANCILHON	Examinateur	DataPublica
Matthieu CORD	Examinateur	UPMC Paris 6
David GROSS-AMBLARD	Examinateur	Université de Rennes 1
Pierre SENELLART	Examinateur	Télécom ParisTech
Anne DOUCET	Directeur de thèse	UPMC Paris 6
Stéphane GANÇARSKI	Co-directeur de thèse	UPMC Paris 6
Benjamin PIWOWARSKI	Encadrant	CNRS / UPMC Paris 6

You must be ready to burn yourself in your own flame; how could you rise anew if you have not first become ashes?

Nietzsche


define
performance
approaches
related
model
use
Navigation
qi
changes
Experiments
used
windows
example
like
Prune
end
archives
collection
ranking
set
number
account
need
change
information
page
Figure
following
also
Pattern
window
ratio
archiving
2N2P
Pruning
algorithm
context
dimension
web
static
documents
However
part
PI
problem
source
version
Index
al
pruning
index
data
TCP
Chapter
selected
interval
IR
navigation
method
contains
PRP
et
postings
posting
diversification
length
pattern
prune
patterns
best
proposed
probability
users
collections
importance
next
relevance
Section
versions
order
follows
list
pages
fix
temporal
propose
section
query
P10
chapter
Static
terms
user
associated
new
coherence
based
relevant
Ratio
given
term
size
IP-u
according
first
two
pruned
tq
using
one
corresponding
shows
defined
links
different
aspects
time
result
MAP
perform
experiments
results
distribution
etc
NDCG
work
document
retrieval
measure
i.e
score
four
e.g
destination
approach
methods
archive
research
function
better
series
Time
shown
diversity
value
access
search
queries
coherent
standard

ABSTRACT

An important amount of the world's cultural and intellectual knowledge is being created on the web everyday. However, the web has en ephemeral nature *e.g.* new information replaces older information constantly without any notification, leaving a significant gap in our knowledge. That's why archiving the web has become a cultural necessity to preserve the knowledge for the next generations. However, the success of any web archive will be measured by the means of access it provides; as it is the case today on the real web. Our research is placed in the context of access to web archives and studies different research problems related to this issue. These research problems are grouped into two main topics: *Access Methods* and *Optimization of Access*. For access methods, we first propose a conceptual model, as well as operators to manipulate them, as the basis of a query language for web archives to better satisfy user information needs. Next, a new navigation method for web archives that takes the coherence of pages into account is introduced. In the context of access optimization, we propose a change detection algorithm to understand and to quantify what happened (and thus changed) between two versions of a web page. Then, we study the behavior of different static index pruning methods with temporal queries before proposing a new diversification-based static index pruning method and showing its application to temporal collections and a substantial gain in performance.

Keywords: access to web archives, coherence navigation, web page change detection, diversification, static index pruning

RÉSUMÉ

Le Web crée chaque jour une quantité importante de connaissances culturelles et intellectuelles. Ses informations sont de nature éphémère car elles sont constamment remplacées, parfois sans aucune notification. C'est pour cette raison que l'archivage du web est devenue une nécessité culturelle afin de préserver la connaissance pour les prochaines générations. Son succès sera cependant mesuré par ses modes d'accès, comme ceux fournis jusqu'ici par le web. Notre recherche situe dans le contexte de l'accès aux archives web, et étudie les différents problèmes d'accès qui y sont liés. Ces problèmes sont groupés en deux thèmes principaux : *Méthodes d'accès* et *Optimisation des accès*. Pour les méthodes d'accès, nous proposons la base d'un langage de requête ayant par objectif de de mieux satisfaire les besoins d'information des utilisateurs. Une nouvelle méthode de navigation est ensuite introduite, qui prend en compte la cohérence des pages. Pour l'optimisation de l'accès, nous proposons un algorithme de détection de changement pour comprendre et quantifier ce qui s'est passé (et a donc changé) entre deux versions d'une même page Web. Nous étudions aussi le comportement des différentes méthodes d'élagage d'index statiques avec des requêtes temporelles. En outre, nous proposons une nouvelle méthode d'élagage index statiques basée sur la diversification et nous montrons son application aux collections temporelles et un gain supstanciel de performance par rapport aux autres approaches.

Mots clés : accès aux archives web , navigation coherence, détéction des changements sur les pages web, diversification, élagage d'index statiques

ACKNOWLEDGEMENTS

And once the storm is over you won't remember how you made it through, how you managed to survive. You won't even be sure, in fact, whether the storm is really over. But one thing is certain. When you come out of the storm you won't be the same person who walked in. That's what this storm's all about.

Murakami

My years as a Ph.D. candidate are the most beautiful storm of my life. What a storm!, full of passion, happiness, science, friendship, knowledge, tears, love... All these years have been shared with many people that I would like to thank.

I would like to express my thanks and appreciations to my supervisors. They complemented each other wonderfully well and they were always there for me. Stéphane Gançarski, as my daily supervisor, is the person who opened the doors of research to me. He always encouraged me, supported me, kept a sense of humour when I had lost mine and was always available for any kind of discussion any time. Anne Doucet has been a role model for me not only as a scientist but also as a directrice. Anne and Stéphane conducted an administrative combat for grants funding my work that I am really grateful for. Benjamin Piwowarski walked through my PhD life in its last year and brought me freshness, extra force to overcome last obstacles and opened new horizons. They all together showed me how to do research. *Many thanks for being that nice with me, working with you was a great privilege.*

Many thanks to Sihem Amer-Yahia and Arjen P. de Varies for accepting to review this manuscript and for their feedback. I am also grateful to François Banchillon, Matthieu Cord, David Gross-Amblard and Pierre Senellart for accepting to be a part of my jury. Special thoughts to Stéphane Gançarski and Matthieu Cord, also for being such cool bosses in EU project.

It has been a great privilege to spend several years in the Database group at LIP6, and its members will always remain dear to me. The group has been a source of friendships as well as good advice and collaboration. Therefore my special thanks go to Bernd Amann, Hubert Naacke, Camelia Constatin and also past and present Ph.D. students Idrissa, Nelly, Andres, Jordi, Clément, Kun and specially to my long term officemates Roxana and Myriam.

My friends in Turkey, in France and in other parts of the World are sources of joy and support. Special thanks to Gözde Üçur for listening to me, calming me down, being a source of support for me. Having her by my side is a great privilege. I thank Ilknur Ayyildiz for encouraging my journey to Paris, helping my adaptation to new Parisian life, supporting me while I was whimpering. Heartfelt thanks to all my friends who found themselves in the storm with me and supported me: Gülçin & Simon, Ilkay, Nergis, Ilke, Rengin, Murat and many others...

I wish to thank all my family. Special thanks go to my dad. *Thanks for always being there for me, for picking me up when I fail, for being such a wonderful dad but also my mom, my best friend, my brother....* (*her zaman yanımda olduğun, her düştüğümde elini uzattığın, bu kadar super bir baba ama ayni zamanda anne, arkadaş ve kardeş oldugun için teşekkür ederim.)* My soulmate, Sébastien Bourbonnais, *thanks for making each day of my life better than the other one and for making me a happier person.*

Finally, I would like to dedicate this work to my dad and also to my mom who left us too soon. I hope that I can make them proud.

Zeynep PEHLIVAN

Anneme ve Babama...

CONTENTS

INTRODUCTION 1

The main focus of this PhD thesis is to study *how to access web archives and optimize their access*. In this chapter, we describe our motivations and the research issues studied in this thesis. At the end of the chapter, we present its overall organization.

1.1 Motivations

The web plays a crucial role in our information society by providing information for all types of events, opinions, and developments within society. Today the web is a mirror of the population that uses it. However, the web has a dynamic nature which means that pages, and often whole sites, are continually evolving, *i.e.* pages are frequently changed or deleted. For instance, [NCO04] measured that after one year 80% of the web pages are no more available.

As the web is the main source of information, archiving the web has become a cultural necessity to preserve the knowledge for the next generations. Web archives which contain up to billions of pages versions obtained by crawling, represent thus a huge information source, inherently greater than the web itself.

This makes web archiving an active research area with numerous open issues like crawler optimization, storage models, etc. An overview of main issues is presented in different works [Mas06, HY11, WNS+11]. Figure 1.1 provides a high level overview of the key issues in the web archiving.

Selection issue is related to determine exactly what to collect. There are basically a few distinct

approaches to web archiving: bulk archiving, domain archiving and thematic archiving. An example of bulk archiving is the approach of the Internet Archive, trying to archive as much of the public web as possible. Some libraries like British National Library, French National Library, focus on domain archiving partially a domain (in deed, the purpose is to archive the entire domain) such as that of a particular country (*e.g. .fr, .uk*). Thematic or event-based archives focus on specific topics as the United States National Elections, the Iraq War, and the events of September 11.

Harvesting, also known as crawling, refers to automating the process of collecting web pages by using crawlers. Crawlers use a seed list to start downloading web content, and follow the hyperlinks to discover and download additional web content. In order to maintain the archive up-to-date, crawler must revisit periodically the pages and update the archive with fresh version snapshots. However, the crawler can not revisit a site and download a new version of a page too frequently because it usually has limited resources (such as bandwidth, space storage, etc.) with respect to the huge amount of pages to archive. In fact, it is impossible to maintain a complete archive of the whole Web, or even a part of it, containing all the versions of all the pages. Another problem in this field is that the web becomes more and more an application and transaction platform, thus increasing the percentage of not publicly available documents (so called deep web). Hence, it is also important that the harvesting technology follows the new web technologies. Crawling is the most studied key issue in the literature.

Storage refers to the process of retaining archived websites on a storage medium securely and reliably. Most common formats are ARC and WARC[1]. **Preservation** consists of the processes that ensure the continuous accessibility of web archives over time. Preservation systems should be designed by taking the threats like media, hardware and software failures, component obsolescence etc. into account [Gla07].

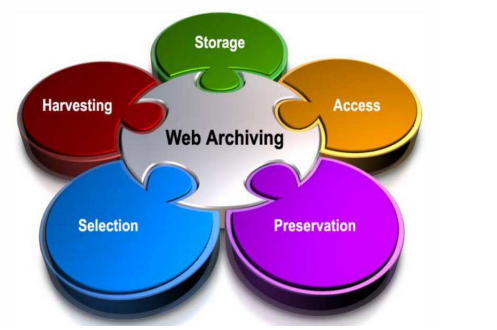


Figure 1.1: Key issues in web archiving [HY11]

The success of the web however is based largely on easy access to all kinds of resources. Thus, we can assume that the success of a web archive will be measured by the access tools the archive provides. In this thesis, we address the major challenges related to the **Access** key issue. How to access to web archives takes on different research questions in different domains (*e.g.* information retrieval, question answering systems, data visualization, web mining etc.). We now present the overall

[1] ARC is a lossless data compression and archival format by System Enhancement Associates (SEA). The WARC file format is a revision and generalization of the ARC format used by the Internet Archive `http://archive.org` to store information blocks harvested by web crawlers.

problems related to accessing web archives.

New access methods: The best known way of accessing web archives is the "Wayback Machine". It is a collaborative project between Internet Archive (IA), the pioneer web archive founded in 1996 as a non-profit company, and Alexa Internet. It was launched in the fall of 2001 and allows users to see captured versions of web pages over time. However, it requires the users to know beforehand which URL to search. For a given URL, the results are displayed in a timeline according to their crawled dates that the user can browse. The other methods currently provided are well-known web access methods: full-text search and navigation within a requested time range.

One of the current projects of the International Internet Preservation Consortium [IIP03], called WERA, is an archive access solution for searching and navigating web archives. It allows a full-text search besides wayback machine style search. These methods are powerful enough for casual users, who search the web for general information and represent the largest proportion of web users. However, according to the "Web Archiving User Survey" [RvB07a], the web archive users profiles consist of historians, journalists, lawyers, students etc. more than casual users. Furthermore, the main reason for using web archives is research activity: web archive users need to analyze, compare and evaluate the information. In order to achieve this, web archive systems must provide tools to execute complex queries.

User Interfaces (Visualization): Web search engines have been extremely successful in enabling users to easily formulate their search goals through an arbitrary list of words, and to quickly receive ranked lists of links to relevant web pages. Unlike in the web, an URL is not unique anymore in web archives. Different versions of the same page can be relevant for a query and this can lead the first page of results containing redundant information. One of the solution to this problem is to group the web objects with the same URL as proposed in [SJ09]. In web archives, not only the pages content is evolving over time but also the linking structure. As the pages are not crawled exactly at the same time on the real web, that leads the users to browse from one page to another where both pages have never appeared at the same time on the real web.

Different projects and national libraries investigate more sophisticated visualization of archived data since a few years. UKWAC (British National Library Web Archives) has new features, such as showing results as a thumbnail of snapshots, like 3D Wall or n-gram visualization which visually enrich the users experiences. Recently, WebART (web archive retrieval tools)[2] proposed new interfaces that displays word frequency, co-word analysis.

Indexing: Web archives search systems need a good index design to support the temporal dimension of the data and queries. It entails issues related to the choice of data structures, as well as building, maintaining and organization of these indexes. There is clearly a need for fast and efficient access regarding to the growing size of archives: the web archives grow fast, *e.g.* the

[2] `http://www.webarchiving.nl/`

web archives of Library of Congress grow at a rate of about 5 terabytes per month[3] , the Finnish web Archive for 148 millions of content objects (e.g. HTML pages, pdfs, images etc.), Canada WA for 170 millions of content objects, Digital Heritage of Catalonia for 200 millions of content objects [GMC11]. This problem is studied in two different directions: updating indexing schemes or index compression techniques. To update indexing scheme, some approaches change the structure of inverted indexes by adding a temporal dimension, besides the one that propose different partitioning methods. There are two basic category for index compression: lossy compression that involves the removal of data (*e.g.* static index pruning) and loseless compression that does not remove the data (*e.g.* 2-DIFF, n-s coding etc.)

Ranking: Given a query and a time constraint, the search systems for web archives should produce the ranked list within the specified time range. Ranking the results by time-aware relevance for temporal queries is a well studied challenge in research community. Time constraints can be provided by users or determined by the system. Which time to use for ranking is another issue as pages can contain different time-stamp (HTTP-header, crawl date, last update, temporal expressions in text etc.) or not at all. Determining the "exact" time-stamp of the page is another challenge. Ranking methods should also take the effect of languages changing over time into account.

Duplicate Detection: Duplicate versions of the same document are created when documents are repeatedly harvested from selected web sites. This induces a waste of storage and it becomes very tedious for those using a web archive if many copies of the same web document are presented in the user interface. Therefore the ability to define when a document has changed and detect (near)duplicates are other challenges for web archiving.

In this section, we reviewed the main challenges related to accessing Web archives. There are other challenges like temporal clustering, temporal summarization, recommender systems for web archives. Accessing web archives is a vast and expanding discipline. Different challenges and future research directions are underlined in recent works [Kan09, WNS+11, ASBYG11]. It is clear that there is a need for highly efficient and practical approaches to access to web archives. The purpose of this thesis is to identify and study some of the related problems and propose approaches as solutions to these problems. In the next section, the research problems addressed in this thesis are presented in detail.

1.2 Research Questions

As mentioned in the previous section, the main question is *how to access web archives and how to optimize this access?* The motivations show that the subject is vast and large, and our research focused on the following topics: *Querying*, *Navigation*, *Change Detection*, *Reducing*

[3] http://www.loc.gov/webarchiving/faq.html

access time.

Querying

The profile of web archives users differ from casual web users: they need to analyze, compare and evaluate the information. Thus, for effective analysis and mining, a declarative query interface that supports complex expressive queries is needed. These queries view a web archives simultaneously as a temporal collection of text documents, as a navigable directed graph, and as a set of metadata related to web pages (length, URL, title, etc.). Ranking and ordering relationships should also be defined over pages and links to filter out and retrieve only the "best" query results by taking the temporal dimension into account. Thus, the first research question we address is:

> *RQ1: What are the requirements, the data model and the operations to define a query language for web archives?*

Navigation

Navigation is one of the search methods used on the web. It consists of following hyperlinks and browsing web pages starting from a source web page. On web archives, it requires that every page be reconstructed. Each hyperlink in archived web pages are reconstructed to let users to navigate like *in real web*. Consider a web page crawled at August 19^{th}, 2009 containing a hyperlink to "www.lip6.fr". When a user starts to navigate from this version in web archive and follows this hyperlink, she should not be redirected to the "www.lip6.fr" of today. That's why all web pages hyperlinks are modified in order to point to the corresponding versions in the archive. The challenge starts here because web archives are not an exact copy of the web as it was at any time - they are inherently incomplete. It means that it is not possible to "*freeze*" the web while crawling to prevent the pages from changing, and at the same time it is not possible to crawl all the web all the time. If the hyperlinks of the source page are not crawled at the same time of the source page, to which version the user should be directed? Thus, the second research question we address is:

> *RQ2: How to to optimize browsing to enable users to navigate through the most coherent page versions at a given query time?*

Change Detection

Capturing the changes between page versions is an important challenge in the web archiving context. This topic relates several key issues presented in Section 1.1. An efficient change

detection approach helps to optimize crawling by deciding if the page should be crawled or not on the fly. For the "Access" issue, the evaluation of web pages can be presented visually by showing the changes through versions. The complex queries can be run over delta files (files that keep changes between versions) whenever possible instead of their original files. This optimizes the execution time of the queries. Hence, the next research question we address is:

> *RQ3: How to know, understand and quantify what happened (and thus changed) between two versions of web page?*

In addition, detecting changes is also important for other issues. Different patterns can be defined based on changes of pages and the crawler can be scheduled according to these patterns. For example, if a page is usually updated at a given day of the week, then it is worth crawling it just after this time point [SPG11]. However, crawling is an highly error-prone process. It may happen that the essential data is missed by the crawler and thus not captured and preserved. To create a high quality web archive, doing quality assurance is essential, for instance, by comparing the live version of the page with the just crawled one. For the "Storage" key issue, a change detection approach helps eliminating (near) duplicates offline to increase the search efficiency by decreasing size of the archive. From the "Preservation" point of view, change detection is necessary to ensure the quality of archives (*e.g.* after format migration like ARC to WARC), to detect format obsolescence following to evolving technologies.

Reducing access time

A common way to access web archives is the use of temporal queries that combine standard keyword queries with temporal constraints. The growing size of web archives require big disk spaces, which in turn leads to big index files that do not fit into the memory. Consequently, the search engine needs lots of disk-accesses, increasing query response time. *Static Index pruning* methods reduce significantly index sizes without significantly changing the retrieval performance by removing the postings that are less likely to alter the ranking of the top-k documents retrieved by query execution. It does not require any knowledge on queries as it is applied offline. Different static index pruning methods are shown to be very successful (*e.g.* [CCF+01, BC06, BB07]). However, none of the existing methods take the temporal dimension into account and they are never tested against temporal queries. Thus, one research question that we address is:

> *RQ4: How the existing static index pruning methods work in presence of temporal queries?*

Consider the query "France Presidential election". It can be used to generate different temporal queries by adding different temporal constraints to it (*e.g.* @2012, @2007 or @[2007-2012]).

Pruned index should be able to provide results that satisfy different temporal information needs. Ideally, the pruning method should keep the documents from different time periods, *i.e.* to preserve the temporal diversity of postings while pruning. Here, we address our last research question:

RQ5: How to prune indexes by taking the temporal diversification into account?

1.3 Contributions

Our contributions consist of both improvements on existing approaches and new approaches. In this section, we give a summary of those contributions by indicating the corresponding research question and the chapters where the details can be found.

Contribution 1 - Querying:

We propose a conceptual model, as well as operators to manipulate them, as the basis of a query language for web archives. In our model, we take into account different topics in web pages by using visual blocks as an unit of retrieval with corresponding importance. A block-based approach is used for information retrieval on the web, however, as far as we know, it is never used with temporal dimension. Navigation operators with temporal dimension let users execute queries over web archives temporal hyperlink structure. The model and operators enriched with the temporal dimension allow querying web archives powerfully. This contribution is a solution to RQ1 and is studied in detail in Chapter 2.

Contribution 2 - Navigation:

By exploiting regular patterns, we propose a novel coherence-oriented browsing that improves the quality of the navigation in web archives. It enables users to navigate through the most coherent page versions at a given query time. This contribution is a solution to RQ2 and is studied in detail in Chapter 3.

Contribution 3 - Change Detection:

We propose a new change detection algorithm that detects differences between two web pages (or two versions of the same page) based on their visual representation. Preserving the visual structure of web pages while detecting changes gives relevant information to understand modifications which is useful for many applications dealing with web pages. It detects two types of changes; structural and content changes. This contribution is a solution to RQ3 and is studied in detail in Chapter 4.

Contribution 4 - Static Index Pruning

We perform the first study on an empirical comparison of static index pruning methods for temporal queries. In addition, we discuss the relation of results to temporal diversification by doing statistical tests. This contribution is a solution to RQ4 and is studied in detail in Chapter 5.

Contribution 5 - Static Index Pruning We propose the first method for static index pruning that takes the diversification of results into account. Then, we show how to use diversification based static index pruning to ensure temporal diversity in temporal collections. This contribution is a solution to RQ5 and is studied in details in Chapter 6.

1.4 Outline

This thesis contains two different parts. The first one presents access methods for web archives, while the second presents optimization methods for accessing web archives. Each part contains different chapters that correspond to different research questions.

In the first part, Chapter 2 presents the requirements for a query language to access to web archives. It presents a data model, existing operators and new operators for this language. In Chapter 3, the new navigation method is proposed for web archives by taking the coherence into account.

In the second part, Chapter 4 presents a change detection algorithm and its different applications in the context of web archiving. Chapter 5 presents the first study that compares behaviour of four static index pruning methods with temporal queries. Chapter 6 introduces a new diversification based static index methods and it shows its application to the temporal collections. We conclude by describing future research directions.

Part I

Access Methods to Web Archives

2 Query Language for Web Archives

Web archive users differ from casual web users. Archive users need to analyze, evaluate and compare the information, which in turn requires complex queries including temporal dimension. These queries can not be performed with current access methods such as wayback machine, full-text search and navigation. In this chapter, we address this requirement by proposing a data model and a temporal query language for web archives, called *WACQL*.

2.1 Motivation

As web archives get larger and larger, sophisticated search functions become more important for archive users. Unfortunately, this area has received relatively little attention until now within the archiving community. For instance, the "Wayback Machine" [Tof07] which allows users to see captured versions of web pages over time, is the best known access method to web archives. The others methods currently provided are well-known web access methods: full-text search and navigation within the requested time range. These methods are powerful enough for casual web users, who search the web for general information without precise complex needs in mind. However, according to the "Web Archiving User Survey" [RvB07b], the web archive users consist of historians, journalists, lawyers, students etc. who have specific needs beyond these of casual users: the main reason for using web archives is research activity. Web archive users need to analyze, compare and evaluate the information. In order to achieve this, web archive systems must provide tools to execute complex queries.

To illustrate this issue, consider a researcher, Derin Deniz, has been assigned to carry out a research on how international media covered the event "Gezi resistance in Istanbul" over last three months. She has a list of web sites that she is interested in (*e.g.* www.cnn.com etc.). For instance, at the very beginning of her research she would like to know the number per week of *different regions* in web pages referring to the event. This kind of queries can not be performed by wayback machine, full-text search or navigation methods but with an efficient query language.

A number of possible user scenarios over web archives are listed and illustrated with technical requirements in "Use cases for access to Internet Archives" [IIP06]. Like the above example, a significant number of them requires complex query capability to be handled.

New approaches to explore web archives with complex queries are needed. An appropriate query language, that enables temporal search besides content queries and structural queries (not only for the hypertext structure, also for web page's internal structure) is especially required for formulating complex queries.

There are a number of query languages proposed for querying web data (e.g WebSQL, W3QL, WebLog, WebOQL, etc.) but most of them are not suitable for web archives due to a lack of temporal dimension and a lack of handling challenges related to web archives like temporal coherence, incompleteness etc..

In our proposed language model, we use visual blocks, which are extracted from a page, as unit of retrieval rather than a whole page. Each element of the model is timestamped with a period called *validity*. Operators for data manipulation, navigation and ranking are proposed. We want to underline the fact that in this chapter we clarify formally the requirements of a query language for web archives (WACQL). We use existing operators (*e.g.* select) and propose new ones specially adapted for searching web archives.

Contributions

The query language that we propose for web archives has the following features:

- It allows block-based temporal search
- It defines different types of operators (e.g. navigation, temporal etc.) that allows to build complex queries adapted to Web archives.

Organization

The remainder of this chapter is organized as follows. In Section 2.2, we present the related

works. In Section 2.3, we present the features of the query languages for web archives. In the following section, the conceptual model is presented with related operators. Before the conclusion in Section 2.5, different use cases are explored.

2.2 Related Works

Prior related works can be categorized into following topics: *Existing methods to access web archives*, *Web query languages* and *Block-based search*. We new present each topic in detail.

2.2.1 Existing methods to access web archives

To explore web archives, traditional access methods, *i.e.* navigation and full-text search are proposed by web archiving initiatives [IA96, Pan99, UKW96]. In the fall of 2001, the Internet Archive (IA) [IA96] launched a collaborative project with Alexa Internet called the "Wayback Machine" [Tof07]. It allows users to go back in time and view earlier versions of a web page for a given URL and is used by most of web archive initiatives. The limitation of this method is the necessity of knowing the exact URLs.

The increasing number of national web archives and diversity of existing works fostered the creation of the International Internet Preservation Consortium (IIPC) in Paris at 2003 with 12 participating institutions. Its aim is to develop common standards, tools and techniques for web archiving. One of the characteristics of IIPC is to develop open source applications like the Heritrix crawler developed through a cooperation of NWA and IA.

WERA, one of the current projects of IIPC, is an archive access solution for searching and navigating web archives. It allows a full-text search besides the wayback machine style search. IA, NWA and IIPC collaborated in this project. WERA is based on the NWA toolset that includes another project called NutchWAX full-text indexer, an extension of the open source search engine Nutch[1] for indexing and searching web archive collections. The Portuguese Web archive, Minerva, BNF, INA adapted NutchWAX to enable full-text indexing and search [IA96, Pan99, UKW96].

In this chapter, we propose a query language that goes beyond what has been proposed so far, and enables experienced users to issue complex queries that one needs while doing research on web archives. A side of the efforts, different projects and national libraries investigate different ways to access web archives since a few years. For instance, the UK Web Archives [UKW96]

[1] Nutch is itself based on the Lucene project.

proposes new features such as tools to navigate between different versions of a page easily, showing results as thumbnails of snapshots like 3D Wall or n-gram visualization which visually enriches the user experiences. Recently, WebART (Web Archive Retrieval Tools) [2] proposes new interfaces that enables word frequency, co-word analysis. Even if they are of interest, we did not focus on those visualization techniques in this thesis.

2.2.2 Block-Based Search

Multi-topics and noisy information on a web page affects the search performance. In recent years, there has been an increasing interest in web based searching by taking these different topics as units of retrieval and by eliminating noisy information like copyrights, navigation bars [BFMS09, CYWM04, ZTY+05]. To achieve this, different page segmentation algorithms are proposed [KDG+02, CYWM03a, CNDZ08]. They partition web pages into non-overlapping hierarchical blocks where each block deals with a different kind of information or only contains noisy information. As web archives are temporal collections of web pages, this block-based approach needs to be extended with temporal dimension in order to be used in web archive search.

Web page segmentation aims to detect automatically these different blocks. Using blocks in web pages as an unit of information retrieval is an active search area. The vector space model customized with importance and permeability (the indexing of neighbors blocks) is proposed in [BFG+09] without temporal dimension. In [CHWM04], after segmenting each page into non overlapping blocks, an importance value is assigned to each block which is used to weight the links in the ranking computation. A block-based language model is proposed in [ZTY+05].

In this chapter, we use this idea of blocks to bring more expressivity to our query language. Our approach is based on visual page segmentation of web pages [BSGP09] and uses an importance model proposed in [Son04].

2.2.3 Web Based Query Languages

A number of SQL-like query languages for Web data have been developed in the past like WebView[WO05],Whoweda[BMN03] WebSQL[MMM97], WebOQL[AM99], etc. They are query languages that model the Web as a kind of relational table, allowing more sophisticated queries and operations than the keyword-based query over the text of the document as search engines provide. As mentioned above, all those languages are intended for online queries on the web except Whoweda that stores extracted web information as web tables and provides web

[2] http://www.webarchiving.nl/

operators to manipulate those tables. From the perspective of web archiving, the most important drawback of those query languages is their lack of handling temporal dimension and the lack of handling challenges related to web archives.

The most related work to our research is the WebBase project [RGM03] at Stanford University. It is a web repository project that aims to manage large collections of web pages and to enable web based search. A web warehouse is interpreted simultaneously as a document collection, as a directed graph and as a set of relations. For web based search, a query language with the notions of ranking, ordering and navigation is proposed. However, one copy of each page at a time is archived, thus, no temporal dimension is provided in that project.

The key differences between WACQL and these query languages are: i) they are intended either for on-line queries on the Web or for Web site management; and ii) they are not properly designed to handle temporal dimension, iii) they do not consider different segments on a web page.

2.2.4 Use Cases

To illustrate the need for complex queries, we present different use cases.

Use Case 1: A social researcher who studies how French media covered the event "earthquake at Haiti in 2010" over last year wants to know the number per month of *different regions* in web pages in domain .fr referring to earthquake.

Use Case 2: Government web sites outgoing links give an idea about the position of governments through social events. Consider a political scientist who want to examine these changes between Bush federal government and Obama's one. She would like to have two lists of outgoing links, one for removed links, one for inserted links. She wants to limit her research six months before and after the election date November 4, 2008.

Use Case 3: Consider a country that has a law that restricts the web site for children to have an access to harmful content in 5 clicks away. They have a black list of URLs that children should not browse. For a web page designed for children, they would like to check if the web sites owner has followed the law in the past six months.

Use Case 4: Consider a journalist who wants to determine the popularity of PhDComics amongst people at University Pierre and Marie Curie from 2000 to 2010.

In conclusion, full-text search and navigation are the only applied solutions to access to web archives which are not appropriate for the cases presented in Section 2.2.4. Most of the web query languages do not contain temporal dimension for querying historical data. Different semantics (one for each block content) of a web page are not processed in search except recent works like [BFMS09, CYWM04, CHWM04] which suffer from lack of temporal dimension to be used for web archives. In our approach, we propose a data model which takes into account different semantic regions of a web page with associated importance and temporal dimension.

2.3 Features of WACQL

The overarching design goal of our approach is to offer a query language for web archive users. In this section, we describe the features of this query language and the rationale behind them.

Block-Based Search: Today, web pages may contain various blocks on it. A typical example is the web pages of newspapers/TV channels like www.bbc.co.uk/news where multiple blocks with unrelated topics are presented with different colors Figure 2.1. A web page with matched terms in the same region is more relevant than a web page with matched terms distributed over the entire page.

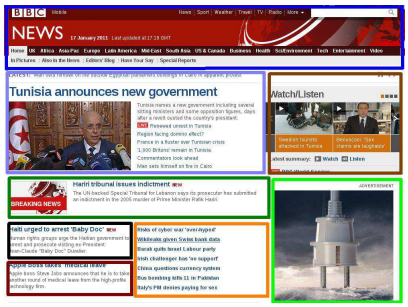


Figure 2.1: Various semantics in a web page

The blocks in a page have different importance as discussed in previous works [KDG+02, CYWM03a]. Different importance weights are assigned to different regions inside a web page according to their location, area size, content, etc. Using a web page as an unit of retrieval does not take into account these different semantics and their importance. In our model, we add this notion to enrich our data model and propose an operator called InBlock explained in detail in Section 2.4.3.

Incompleteness and Temporal Coherence: Due to the limited sources, all web archives are incomplete (*i.e.* they do not contain all possible versions of all the pages on the web) and we should query them as they are. If a user asks for a version at t, and if the archive does not have

versions at *t* but it has the versions at *t-2* and at *t+2*, the access model should decide or support different choices. This is supported with Nearest and Recent operators described in Section 2.4.3.

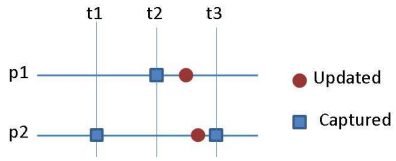


Figure 2.2: Temporal Coherence

Temporal coherence is another issue in web archiving. The main reason is the dynamic behaviour of the web. It changes continuously in an unpredicted and unorganized manner. The web sites politeness constraints and the limited allocated resources do not allow archiving a whole web site at once, at the same moment. For example, in Figure 2.2, for a temporal navigation starting from the version of p1 at *t2*, p2 at *t1* is coherent, while p2 at *t3* is incoherent. The access model should be able to find the most coherent version. This is supported with Coherent operator described in Section 2.4.3. We study this issue in detail in Chapter 3.

Temporal Ranking and Grouping: Ranking and grouping are the most common ways to deliver query results for large-scale data. For web archives, the temporal dimension must be included in ranking and grouping process. Ranking has also a dynamic nature in WACs. For example, a page archived mentioned "Obama" in 1999 should not have the same ranking result when querying in 2000 and in 2010. Rank, Order and GroupBy operators are used for these purposes.

Temporal Predicates: Support for temporal predicates enriches the language. For example, a researcher who wants to analyze the effects of the arrest of Julian Assange can execute a query: "Find pages linked to wikileaks.org *after* Julian Assange's arrest in London". Allen's interval based temporal logic operators [AF94] are appropriate for these purposes and are included in our query language.

User-Friendliness: This query language will be used by researchers and casual web users. So it should enable users to find information without long-term training. Simple queries (like keyword search) should be expressed straightforwardly. Complex syntax should be used only to express more complex queries. We believe that with an advanced GUI, the users can avoid writing "codes". However, this chapter does not discuss this aspect of the language and focus on the data model, operators and predicates.

2.4 Data Model

In this section, we present how time and web archives are modeled in our approach.

2.4.1 Modeling Time

Before presenting the time data model that we use, we first discuss how time information can be associated to a web page and to the queries.

Time on the Web

Integrating time into data models, query languages and database management system implementations continues to attract the attention of researchers. In the context of web archiving, there are different types of temporal information: time in content, HTTP headers, crawled time.

Temporal expressions can be found embedded in the content of a web page. Those expressions are divided into three categories in [AGB07]: Explicit, Implicit and Relative. Explicit temporal expressions refer to a specific point in time like "December 24, 2010". Names of holidays or events which can be anchored in timeline are considered as implicit expressions like "Christmas Day 2010". Relative expressions can not be anchored in timeline directly like "today", "on Wednesday" etc. Extraction of temporal expressions from documents is an active research field.

For web archives, we are more interested by the information contained in the temporal fields in HTTP headers contain `Date` , `Last-Modified` and `Expires` as they are linked to web page changes time.

Servers send Last-Modified header with date time value when the content was changed last time. `Last-Modified` header is mostly used as a weak validator. According to the definition of HTTP specifications, a validator that does not always change when the resource changes is a weak validator. The meaning of `Last-Modified` depends on the implementation of the origin server and the nature of the original source (files, database gateways, virtual objects, etc.) [W3C99]. The header `Expires` indicates when a document stops being fresh. If it is used correctly, this header indicates a validity period for the document and an exact date for recrawling. But it is an unreliable tool: many servers do not provide the header or provide with zero or with low expiration delay.

In HTTP headers `Date` represents the date and time at which a web page is returned from a HTTP server upon request by the HTTP client. Crawled time is the time that crawlers capture

the snapshot of a page. Web crawlers set the value of the Date field in crawled documents to the date that a document is crawled, unless they are configured otherwise. It is the most trustworthy time because it does not depend on host servers configurations. In our model, crawled time is used but we should underline the fact that our model supply also other choices.

As we saw above, there are various sources of information to determine the last modification date of a document; as they all are uncertain, we do not want a time model where only a timestamp is associated with one web page. Hence, our model uses Allen's interval-based representation of temporal data [All83a] where intervals are used as primitive time elements. Each element is temporally stamped by a period, called *validity*, defined as a time interval $[t_s, t_e)$ where t_s represents a starting time point and t_e is an ending time point.

Figure 2.3: Example of a web page crawled at three different time point

Time in queries

In our view, there are two kinds of temporal queries: time-point and time-interval. In time-point queries, a time point t is given in the query and the results should contain data whose validity contains t. In time-interval queries, an interval [t1,t2) is given in the query and data whose validity contains / overlaps with [t1,t2) are returned as result. A time-interval query is also called a time slice query.

2.4.2 Modeling Web archives

In this section, we define the data model of WACQL. We will use Figure 2.3 to illustrate the main features of the model. In this example, we have a web page crawled at t1, t2 and t3. The page is segmented into three blocks that we will refer by colors (green, blue and pink). An overall view of our model is given in Figure 2.4. Site is modeled as a set of pages that share a common domain. A page is a set of blocks which is a set of contents, links and importances. Each of those has a validity time-interval $[t_b, t_e)$, notated as *val* to denote the duration during which the the element was valid.

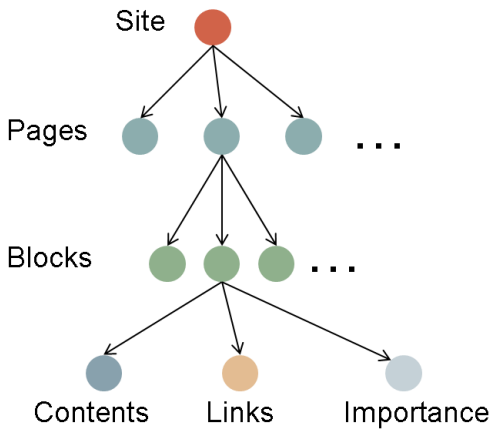


Figure 2.4: Overall Model

We now define more generally all the objects we rely upon for WAC.

Site: A website (s) is a set of web pages that are addressed relative to a common URL.

$$s = \{p_{url1}, p_{url2}, p_{url3} \cdots p_{urln}\}$$

A snapshot of a website at time t is a set of pages valid at t.

$$s^t = \{p^t_{url1}, p^t_{url2}, p^t_{url3} \cdots, p^t_{urln}\}$$

Page: A page (p_{url}) is a set of blocks sharing a common URL. The validity of a page is the union of all blocks' validity.

$$p_{url} = \{cb_1, cb_2, cb_3 ...\}$$

A snapshot of a page (p^t_{url}) is a set of block snapshot for a given URL valid for a requested time.

$$p^t_{url} = \{cb^t_1, cb^t_2, cb^t_3 ..., cb^t_n\}$$

Block: The web page segmentation (see Chapter 3 for more detail) returns a set of non overlapping blocks. It contains: the URL to which it belongs, a Dewey identifier that indicates its structural position in the page, its validity interval *val*, $\{c\}$ is a set of content and $\{i\}$ is a set of importances and $\{l\}$ is a set of links from a block. If a block disappears and reappears later, it is considered as a new block. A block is defined as follows:

$$b = (Url, DeweyID, val, \{c\}, \{i\}, \{l\})$$

We next explain in detail content, importance and links:

- **Content:** Contents are objects like images, videos or texts in a web page. A *validity* attribute allows to trace the content changes on its block. Its validity should be included in the validity of its containing block. A content of a block is defined as:

  $$c = (object, val)$$

- **Importance:** The blocks in a page have different importances (Section 2). We define the importance as:

  $$i = (\alpha, val)$$

  The importance of a block, denoted α, can depend on its location, area size, content, etc. following the chosen importance model *e.g.* [Son04]. The validity of importance is not equal to the validity of the content in the same block. Although its content stays unchanged, the importance of a block can change by added / deleted link or image, or by updating blocks size in the page.

- **Links:** Links represents the hyperlinks in the page. They point to a page from a block and are defined as follows:

  $$l = (label, type, to, val)$$

  label is a link label as shown here: $< a\ href = "URL" > linklabel < /a >$. The attribute *type* is used to distinguish global, internal and local hyperlinks.

  An hypertext link in an HTML document is said to be:

  - interior if the destination document coincides with the source document
  - local if the destination and the source documents are different but in the same domain (ex:href="/news/politic.html")
  - global if the destination and the source documents are located on different servers.

Example: We illustrate the above by showing how the the block colored in green of Figure 3.5 is described using our data model.

$$b = (www.bbc.co.uk/news, 2.2, [t1, now), C, I, L)$$

where

$$C = \left\{ \begin{array}{c} (French, hostage..., [t1, t2)) \\ (image, [t1, t2)) \\ (Woody, ageing..., [t2, now)) \\ (image, [t3, now)) \end{array} \right\},$$

$$I = \left\{ \begin{array}{c} (0.1, [t1, t2)) \\ (0.3, [t2, t3)) \\ (0.4, [t3, now)) \end{array} \right\},$$

$$L = \left\{ \begin{array}{c} (\text{“}TwoFrenchhostagesinNiger\text{”}, local, green, \text{“}/news/world-4598422\text{”}, [t1, t2)) \\ (\text{“}WoodyAllenonageinganddeath\text{”}, local, green, \text{“}/news/cinema-2659874\text{”}, [t2, now)) \\ \end{array} \right\},$$

2.4.3 Operators

The proposed language consists of classical set operators (SQL) with their temporal extensions: relational operators, navigational operators, temporal and text predicates, aggregation operators, ranking and grouping operators. In this section, we present the new operators that we propose. The list of existing operators can be found in Table 2.1 and their formal definitions can be found in Appendix B. All the operators presented in this section takes a bag of tuples and returns a bag of tuples.

Content Related Operations

InBlock: It finds matches that satisfy a full-text selection in the same block. For example, assume that we are looking for information about Woody Allen's new movie in which Carla Bruni Sarkozy participates (Figure 3.5). A query like “Sarkozy and Allen” will return version at t2 and version at t3. In fact, information in version at t2 is not relevant. But a query like “Sarkozy InBlock-AND Allen” will return only version at t3 which has more relevant information.

OPERATOR	DESCRIPTION
	Unary Operators
Select	corresponds to well-known select operator in the relational algebra
Project	corresponds to well-known project operator in the relational algebra
GroupBy	creates groups of tuples sharing some attribute value
	Binary Operators
Union	returns the union of two bags
Temporal Union	returns the union of temporal elements where their validities overlap
Intersection	returns the intersection of two bags
Temporal Intersection	returns the intersection of temporal elements where their validity overlap
Difference	returns all the elements of the first bag which are not in the second bag
Temporal Difference	in addition to non-temporal Difference, it checks and removes overlapping temporal parts from bags
Cartesian Product	computes the Cartesian product of two bags
Temporal Cartesian Product	returns a temporal Cartesian product of two bags
Join	performs an inner-join on two non-temporal bags based on predicates
Temporal Join	performs an inner-join on two bags based on predicates
Aggregate Operators	COUNT, MIN, MAX
	Predicates
Temporal Predicates	Allen's temporal operators [All83b]
Logical Text Predicates	OR, AND, MILD-NOT, NOT, CONTAINS, LIKE
Temporal Operators	Max,Min,T-Intersect, T-Union, Minus
	Various Operators
Rank	applies a ranking function over a bag
Distinct	removes duplicates for non temporal bags
Temporal Distinct	in addition to non-temporal Distinct, it removes duplicates by taking into account overlapping temporal parts
OrderBy	sorts a set into a specific order

Table 2.1: List of Operators

Wayback: It returns a bag of blocks of a page identified by its URL for a given period.

$$WAYBACK(URL, [ts, te))$$

Nearest/Recent/Coherent: These operators are used to deal with incompleteness and incoherence as explained in Section 2.3. For example, in Figure 2.5, if the version at t is requested where $t1 < t < t2$, we need to make an assumption over the version at t because the page p1 is not captured at this time point. Three different operators are proposed:

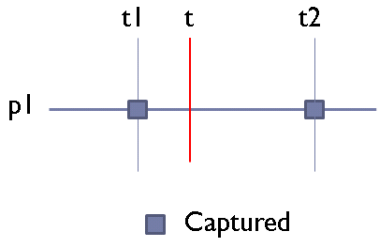


Figure 2.5: Example for Nearest/Recent operators

- Nearest: it returns the nearest time point by minimizing $|t - tx|$
- Recent: it returns the closest time point before t. It is the default operator, if the user does not specify another one.
- Coherent: it returns the most coherent one like described in Chapter 3.

T-Flat: This operator takes a set as input and eliminates the nesting occurred as a result of temporal dimension (validity period). It creates a new validity period by using the time operators (e.g T-Intersect, Minus) over each element's validity attribute. It eliminates the temporal dimension of each element and then it creates a new set with non temporal elements and with the new validity it computed. We underline that the output set can keep its nested structure depending on elements non-temporal type.

For instance, if we want to eliminate temporal nesting in data (*e.g.* to find different versions), T-FLAT operator can be used. For the example of block colored with green b_{green}, T-FLAT returns temporally flattened blocks as follows:

$$T - FLAT(b_{green}) = \left(green, \left\{ \begin{matrix} (French, hostage...) \\ (image) \end{matrix} \right\}, 0.1, l6, [t1, t2)\right)$$
$$(green, (Woody, ageing...), 0.3, l7, [t2, t3))$$

$$\left(green, \left\{\begin{matrix}(Woody, ageing...)\\(image)\end{matrix}\right\}, 0.4, l7, [t3, now)\right)$$

Navigational Operators

The operators described in this section add navigation capabilities into the query language.

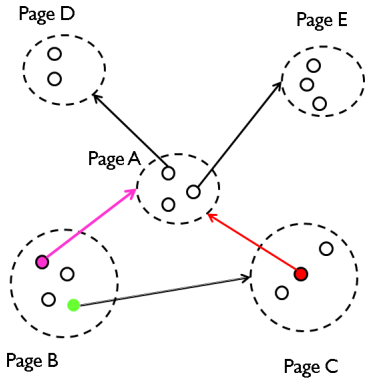


Figure 2.6: Example for navigational operators

Out/out_b: Operator out_b finds the bag of pages pointed in one step (by following one link, one edge) from the bag of blocks (CB) by following any of the links valid at a given period. Operator *out* uses out_b to find the bag of pages reachable in one step from the bag of pages (P) during a given period. For that, it finds the bag of blocks for each page in P and calls out_b for each CB. Each element is checked for the validity constraint.

$$out : \mathbb{CB} \times period \rightarrow \mathbb{CB}$$

$$out_b : \mathbb{CB} \times period \rightarrow \mathbb{CB}$$

In Figure 2.6, pages are presented as nodes in dashed circles, blocks in pages are represented as circles in pages and the hyperlinks are represented as directed arcs. For the green block in Page B out_b returns Page C which is a set of blocks. Operator *out* over Page B returns the bag of pages including Page A and Page C.

In:

$$in : \mathbb{B} \times period \rightarrow \mathbb{B}$$

Operator *in* finds the bag of blocks that points to a bag of pages by links valid for a requested period. In Figure 2.6, *in* operator for Page A returns two blocks colored in pink and red.

Jump:

$$jump^{+} : \mathbb{B} \times int \times period \rightarrow \mathbb{B}$$

$$jump^{-} : \mathbb{B} \times int \times period \rightarrow \mathbb{B}$$

Operators $jump^{+}$ (resp. $jump^{-}$) returns a bag of pages reachable through incoming (resp. outgoing) links in one to n steps by following links valid at a given period. It is defined as a combination of *in* and *out* operators with iteration.
For example, in Figure 2.6, $jump^{-}$ from block red in 2 steps return Page A, Page D and Page E.

2.5 Queries for use cases

In this section, to illustrate how the different operators work in our approach, we use the examples in Section 2.2 and construct the corresponding queries.

- ***Example 1:*** A social researcher who studies how French media covered the event "earthquake at Haiti in 2010" over last three years wants to know the number per month of *different regions* in web pages in domain .fr referring to the earthquake.

  First, we need to find all blocks in domain .fr which are valid between 2010 and 2013. LIKE is used to compare string patterns. Then, with CONTAINS operator, we find contents which mention given keywords (Haiti, earthquake). EXPAND operator is used to group by month over validity of content. COUNT operator is used to find out the number of different regions. By using GROUP BY operator with url, we can limit the count to web pages.

- ***Example 2:*** Government web sites outgoing links give an idea about the position of governments through social events. Consider a political scientist who want to examine these changes between Bush federal government and Obama's one. She would like to have two lists of outgoing links, one for removed links, one for inserted links. She wants to limit her research six months before and after the election date November 4, 2008. This analysis will take a huge time using a wayback machine.

  By using our proposed query language, we first find all blocks in domain .gov for two different periods: 6 months before election and 6 months after election. Then, for each, we select a set of links, L1 and L2 respectively. By using classical "difference" set operator (-), L1-L2 gives the list of inserted links and L2-L1 gives the list of removed links.

- ***Example 3:*** Consider a country that has a law that restricts the web site for children to have an access to harmful content in 5 clicks away. They have a black list of URLs that children should not browse. For a web page designed for children, they would like to check if the web sites owner follows the law in the past six months.

  For each web site, we first need to call $jump^{-}$ with n = 5 in time range of six months. Then, we select all links and call a join operators on the URL attribute of links on blacklist.

- ***Example 4:*** Web archives are important sources for trend analysis. Consider a journalist who wants to determinate the popularity of PhDComics at University Pierre and Marie Curie from 2000 to 2010. at UPMC from 2000 to 2010. For each url like "*upmc.fr/~*", we can count the outgoing links to PhDComics web pages.

2.6 Discussions and Outlook

In this chapter, we start to address the problem of accessing information in web archives. We presented a conceptual model as the basis of a query language for web archives. The operators to support queries are also described. In our model, we take into account different topics in web pages by using visual blocks as an unit of retrieval with assigned importance. Block-based approach was used for information retrieval on the web, however, as far as we know, it is never used with temporal dimension. Navigation operators with temporal dimension let users to execute queries over web archives temporal hyperlink structure. The model and operators enriched with the temporal dimension allow querying web archives powerfully, as illustrated by the use cases we described in this chapter. We want to underline the fact that in this chapter we clarify the requirements of WAC query language formally. It can be implemented as a new query language or as an extension of an existing query language (*e.g.* [RGM03]).

3 Coherence Oriented Navigation using Patterns

Maintaining archives of good quality is not an easy task because the web is evolving over time and allocated resources are usually limited (*e.g.* bandwidth, storage space, etc.). Hence, during crawling one may never be sure if the contents collected so far are still consistent with those contents to be crawled next. Approaches that try to maximize the quality can be based on different measures like coherence, completeness etc. and can be applied *a priori* (*e.g.* crawling optimization) or *a posteriori* (off-line). In this chapter, we are interested in coherence as a quality measure. Coherence measures how much a collection of archived pages versions reflects the real state (or the snapshot) of a set of related web pages at different points in time. Coherence in web archiving context is studied in earlier works as a part of *a priori* solutions, like crawling optimization. In this chapter, we propose an *a posteriori* solution based on web changes patterns. This solution introduces a novel navigation approach that enables users to browse between the most coherent page versions at a given query time.

3.1 Motivation

Navigation on the Web (also known as surfing) is the activity of following hyperlinks and browsing Web pages to seek an information. Navigation can be classified into three different categories according to [Lev11]:

- *Direct Navigation*, when the URL is given directly to the web browser,
- *Navigation within a directory*, when a web portal (*e.g.* www.yahoo.com) is used as an entry point,
- *Navigation using a search engine*, when a search engine is used to find an entry point. It consists of the following steps: query formulation, selection, navigation, query modification.

The Web can be represented as a directed graph called web graph. Figure 3.1 shows an example of a simple web graph where web pages are nodes, links from one page to another are directed edges and the labels on the edges can be attribute names.

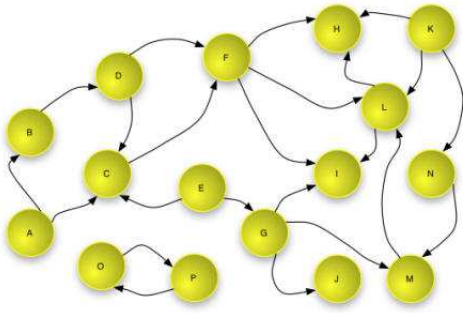


Figure 3.1: Web graph example

However, navigation in web archives is different from the web because of the temporal dimension and the challenges related to web archiving (*e.g.* coherence, incompleteness etc.). To ensure coherence, a user should ideally browse between the versions of pages from the same point in time.

In the navigation process, the user starts to navigate from an entry point (*e.g.* an url provided by search engines) by following hyperlinks until they find the information they are seeking for, or until they restart the process. In some cases, users simply give up the process when they loose the context in which they are browsing. This phenomenon called *navigation problem* or *getting lost in hyperspace* [LL01].

This problem remains in the archival context but, in addition, we define a new phenomenon called *temporal navigation problem* or *getting lost in time*. This can happen with any kind of temporal navigation model. User who seeks information between $[t_x, t_y]$ can find himself/herself, staying or not staying in the same context but in a different time period.

The issue that we study in this chapter is that of "coherence". During navigation in archives, users may browse from one page to another where both pages have never appeared at the same time on the real web. This may lead to conflicts or inconsistencies between page versions, and

such pages versions are considered temporally incoherent. For example, a user reading an article related to funerals of Steve Jobs can follow an hyperlink from the current page and reach a page discussing the participation of Steve Jobs to the new launch of iPhone 5. The reason is that these two pages were not crawled at the same time.

Figure 3.2 depicts an example of such an incoherence. We consider two pages URL_1 and URL_2 of a site. The pages were updated (red circles) at time t_5 for URL1 and t_0, t_4 for URL2. The page URL_1 was crawled (blue squares) at time t_1 (A) and t_6 (A') while URL_2 was crawled at time t_3. A' and B are not coherent because they have never appeared at the same time on the Web (caused by pages update). However, the two versions A and B are coherent because they appear "as of" on time interval $[t_1, t_2, t_3]$.

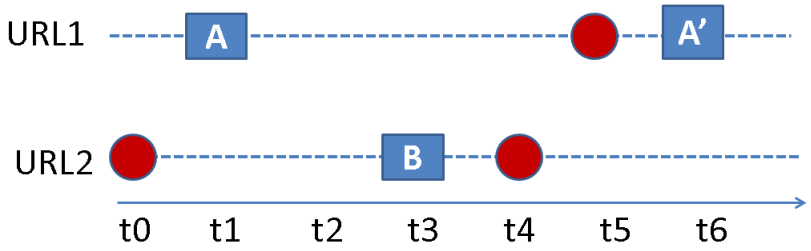


Figure 3.2: Coherence example in web archives

Formally, observable coherence is defined as follows in [SDM+09]:

Definition 1. A set of n Web contents are observable coherent if there exists a single time point $t_{coherence}$ such that there is a non-empty intersection of the intervals spanning the respective download time $t(p_i)$ and the corresponding last updated stamp μ_i retrieved at time of download ($\mu_i \leq t_i$):

$$\forall p_i, \exists t_{coherence} \in \cap_{i=1}^{n} [\mu_i, t(p_i)] \neq \emptyset$$

Temporal incoherence occurs when pages change their content during the crawl of an entire collection. This is a potential issue even for web sites of moderate size. For larger collections, *e.g.* domain level, it is more evident to observe incoherence as a full crawl can take days or weeks to complete. This problem can be handled by two solutions: *a priori* and *a posteriori*. The *a priori* solutions adjust crawlers strategy to maximize the coherence of collected pages versions independently of other quality measures (*e.g.* completeness, freshness, etc.). It is impossible to obtain 100% of coherence in the archive due to the limited resources, so an *a posteriori* solution is also needed that deals with incoherence at access level (*e.g.* while navigation). Thus, our challenge is to optimize browsing to enable users to navigate through the most coherent page versions at a given query time. By exploiting regular patterns of web page changes, we propose a novel coherence-oriented browsing that improves the quality of the navigation in web archives.

Contributions

The work presented in this chapter makes the following contributions:

- We propose the first study on coherence-oriented navigation for web archives.
- Experiments conducted as a simulation based on real patterns extracted from France TV web sites.

Organization

The remainder of this chapter in organized as follows. In Section 3.2, we present the related works. Then, in Section 3.3, we give a background information on patterns and their applications on web archiving. In the following section, our approach *coherence oriented navigation* is presented in detail. We compare our proposed method to existing ones in Section 3.5 and conclude in Section 3.6.

3.2 Related Works

In web archives, navigation, also known as surfing, is enriched with the temporal dimension. In [JKN+06], two different categories are defined: *horizontal navigation* and *vertical navigation*. Horizontal navigation lets users browse chronologically among different versions of a page, while vertical navigation lets users browse by following hyperlinks between pages like in the web. In this chapter, we are interested in vertical navigation. Although it looks like navigation in the real web, the issues induced by web archives (temporal coherence and incompleteness) lead to broken or incoherent links which hamper the correct navigation. There are two well-known policies used for navigation in web archives: *Nearest* and *Recent*. The *Nearest* policy links to page versions that are the closest to the query time t_q. The *Recent* policy links to page versions that are the captured before the query time t_q. Even when finding the nearest version from different web archives, like in the Memento framework[1] [dSNS+09], there is no guarantee that version reflects the navigation like it was in real web. None of this method ensures the maximum coherence between two pages.

[1]Rather than expecting people to know about the growing number of Web archives, and to guess which archive might hold an older version of the resource they're looking for, Memento proposes to make archived content discoverable via the original URL that the searcher already knew about.

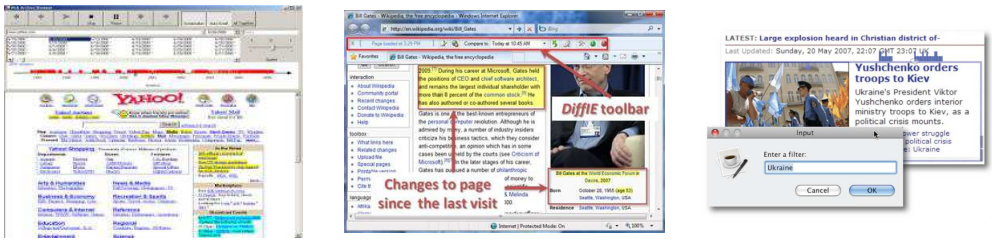


Figure 3.3: Example of temporal browsers resp. [JKN+06, TDLH09, ADFW08]

There are several browsers proposed to navigate over historical web data [JKN+06, TDLH09, ADFW08] that are interested in navigation between versions of the same pages by showing the changes over versions, as seen in Figure 3.3. As far as we know, no approach proposes to improve the navigation in web archives by taking the temporal coherence into account. The reason, as explained in [BCF+08], can be that temporal coherence only impacts the very regular users who spend lots of time navigating in the web archives. Even though, today the archive initiatives do not have many users, we believe that, popular web archives (*e.g.* Internet Archive, Google News Archive) will get the attention of more and more regular users over web archives. For instance, there were 9434 unique visitors with 48318 pages viewed on the UK Web Archive between 1st and 30th of July 2012[2].

3.3 Background on application of Patterns in Web Archives

In this section, we introduce the background necessary for the remainder of the chapter. We give the definition of (time) patterns of page changes in the web archiving context. As these patterns contain information about the updates on the pages, they will be used to estimate the coherence between versions and to choose the "most" coherent version to browse. A pattern is a model or a template used to summarize and describe the behavior (or the trend) of a data having some recurrent events (*i.e.* cycles). A pattern for a web page models the behavior of page's changes over periods of time, *i.e.* during a day. We assume that it is periodic and may depend on the day of the week and on the hour within a day. Pages with similar changes behavior can be grouped to share a common pattern. For example, France2.culture.fr and France3.culture.fr have the same template and share quite similar information. As they also follow similar updates, they have a common pattern.

In order to discover patterns, page changes are observed and monitored for a long period of time. The framework as proposed by [BSG11] consists of five different phases to discover patterns

[2]ISO report: Information and Documentation: Statistics and Quality Indicators for Web Archiving http://netpreserve.org/sites/default/files/resources/SO_TR_14873__E__2012-10-02_DRAFT.pdf

from web pages:

- Crawling: The crawler harvests the web by iteratively downloading new versions of pages. The frequency of the crawler is chosen to be not too long (*e.g.* hourly crawl) to do not miss relevant modifications to generate the patterns. Each retrieved version is then stored and indexed in the archive.
- Change Detection: Based on the change detection algorithm, called Vi-DIFF, detailed in Chapter 4, each crawled page (v_n) is compared with its previous version (v_{n-1}).
- Change importance Estimation: In this phase, the successive delta files generated during a day are evaluated [BSG11] to estimate the importance of changes. Then, for each page, a time series that represents the evolution of the importance changes during a day is built.
- Pattern discovery: The set of time series are analyzed to extract periodic patterns for each page based on changes average over time. Patterns are discovered by averaging the importance of changes of time series collected during a long period.

A pattern obtained by following these steps is shown in Figure 3.4 and formally defined as follows:

Definition 2. *Pattern*

A pattern of a page P_i with an interval length l is a nonempty sequence Pattern(P_i) = $\{(\omega_1,[t_1,t_2[);(\omega_2,[t_2,t_3[);...\}$, where N_T is the total number of periods in the pattern and ω_k is the estimated importance of changes in the time range $[t_k,t_{k+1}[$.

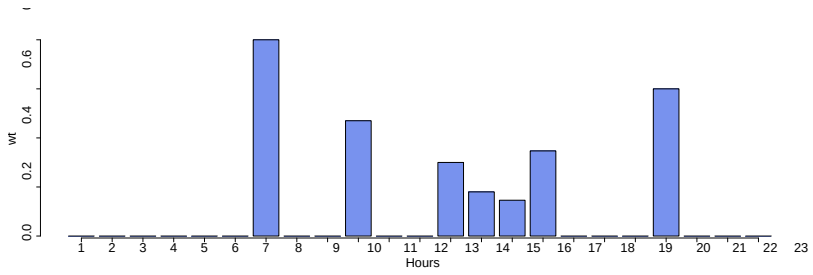


Figure 3.4: Page Changes Pattern

3.4 Coherence Oriented Navigation

While navigating in the archive, we focus on the coherence of two linked pages: the source page and the destination page pointed by an hyperlink from the source page. Coherence-oriented

browsing can be exploited between different pages belonging to different web sites. In the following, a page is denoted by P_j and a version of the page crawled at instant t is denoted by $P_j[t]$.

3.4.1 Informal Overview

A simple example is given in Figure 3.5 to better explain our coherence-oriented navigation approach. Consider a user who starts to navigate in the archive from the version of the page P_1 captured at t_q ($P_1[t_q]$). This user wants to follow the hyperlink to browse the page P_2. The closest version of P_2 before t_q is $P_2[t_1]$ and the closest version of P_2 after t_q is $P_2[t_2]$. They are the candidate destination versions. We assume that the patterns of P_1 and P_2 which describe the behavior of changes are known. These patterns are used to decide which version is the most coherent. As shown in Figure 3.5, the sub-patterns, defined according to the periods $[t_1, t_q]$ (in red) and $[t_q, t_2]$ (in green), are extracted from the patterns of P_1 and P_2. To find the most coherent version to $P_1[t_q]$, we estimate the importance of changes for each sub-pattern. The smaller the importance of changes predicted by sub-patterns is, the smaller the risk of incoherence. Thus, the group of sub-patterns (a) and (b) is compared to other group of sub-patterns (c) and (d) by using the importance of changes. The group of sub-patterns which has the smallest total importance of changes is selected. This means that the navigation through the corresponding page versions in the selected group is more coherent. In the example, the group of sub-patterns (a) and (b) has smaller importance of changes than the group of (c) and (d). Thus, the most coherent version $P_2[t_1]$ (corresponding to sub-pattern (b)) is returned to the user.

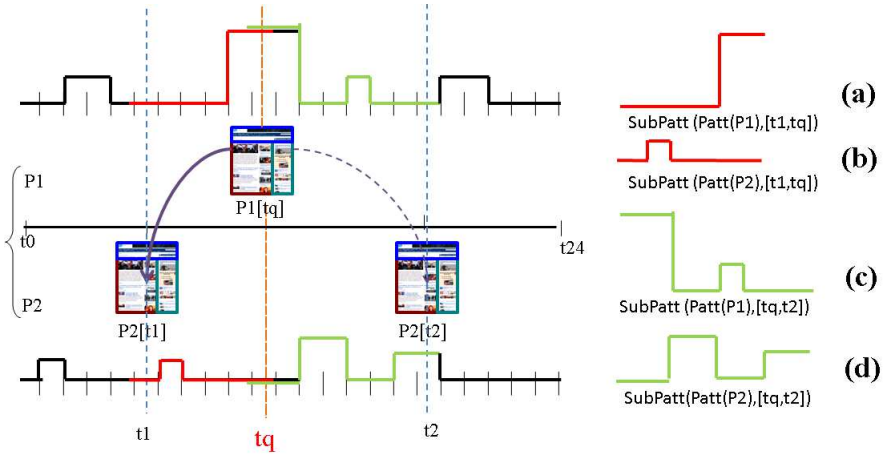


Figure 3.5: Coherence-oriented Navigation

3.4.2 Formal Definitions

In this section, we give the formal definitions of our approach explained in the previous section.

The definition of a pattern given in Section 3.3 is appropriate for *a priori* solution, *e.g.* for a crawler as to decide when it is appropriate to crawl, it is enough to check only one daily pattern for each page. However, when navigating in Web archives, two pages versions can ?? on arbitrary amount of time, so we define a function that extends patterns for our needs. For the rest of the chapter, pattern will refer to extended pattern.

Definition 3. *Extended Pattern*

Given a pattern Pattern(P) with length T for the page P, the extended pattern (Patt) is defined as follows:

$$Patt(P) = \bigcup_{k} \bigcup_{(w,[t,t'[)\in Pattern(P)} \{(w,[t+kT,t'+kT[)\}$$

We now define how to extract sub-pattern to measure later the coherence between the source page and the candidate destination page.

Definition 4. *SubPattern*

Given a pattern Patt(P), the sub-pattern $SubPatt(Patt(P),[t_x,t_y])$ is a part of the pattern valid for a given period $[t_x,t_y]$.

$$SubPatt(Patt(P)) = \bigcup_{(w,[t,t'[)\in Patt(P)} (w,[t,t'[) \text{ where } [t,t'[\cap[t_x,t_y]) \neq \emptyset$$

Following two functions are defined to measure the coherence between two page versions. The first one (Ψ) returns overall importance of the change on one pattern, while the second one returns the sum of the pattern importance for sub-patterns of each page for a period of lag of crawling.

Definition 5. *Pattern Changes Importance*
The function $\Psi(Patt(P))$ estimates the total importance of changes defined in the given pattern $Patt(P)$. It is the sum of all changes importance ω_i of $Patt(P)$.

$$\Psi(Patt(P)) = \sum_{i=k}^{i\leq l} \omega_i$$

Definition 6. *Navigational Incoherence*
Let $P_s[t_q]$ be the source version where the navigation starts. Let $P_d[t_x]$ be the destination version ($P_d[t_x]$) pointed by an hyperlink. The navigational incoherence (Υ) between the two versions

$P_s[t_q]$ and $P_d[t_x]$ is the sum of changes importance predicted by their corresponding sub-patterns during the period $[t_q, t_x]$. t_q and t_x are respectively the instants of capturing the source and the destination versions.

$$\Upsilon(P_s[t_q], P_d[t_x]) = \Psi(SubPatt(Patt(P_s), [t_q, t_x])) + \Psi(SubPatt(Patt(P_d), [t_q, t_x]))$$

We now define a function to find out the most coherent destination version, it compares the navigational incoherence (Υ) between the source version and the set of the candidate destination versions and returns the one smallest incoherence.

Definition 7. *Most Coherent Version*

The probability of being incoherent depends on the importance of changes of sub-patterns. In other words, if there are less changes (smaller Υ), the source and the destination version are expected to be more coherent. Let δ be the set of possible destination versions, the most coherent version is described as follows:

$$P_d = argmin_{p \in \delta} MCoherent(P_s[t_q], p)$$

Example 1. We take the example in Figure 3.5 to explain the process. We assume that the importance of four changes occurred on the page P_1 are 0.3, 0.6, 0.1, 0.3. For the page P_2, the importance of four changes are respectively 0.1, 0.1, 0.4, 0.3. The pattern change importance of four sub-pattern (a,b,c,d) are respectively 0.6, 0.1, 0.7 (0.6 + 0.1), 0.7 (0.4 +0.3).

$$MCoherent(P_s[t_q], \{P_d[t_x], P_d[t_y]\}) = P_2[t_1] \text{ where}$$
$$\Upsilon(P_1[t_q], P_2[t_1]) = 0.6 + 0.1 = 0.8 \text{ and } \Upsilon(P_1[t_q], P_2[t_2]) = 0.7 + 0.6 = 1.3$$

3.5 Experiments

We evaluate here the effectiveness of the coherence-oriented navigation. The coherence of our navigation policy *Pattern* is compared to two related navigation strategies as discussed in Section 3.2: *Nearest* and *Recent*. The *Nearest* policy uses the closest version to the query time t_q. The *Recent* policy uses the closest version before the query time t_q.

However, finding a suitable dataset is not so evident. It should contain crawled web pages with their patterns but also for each link on each page version, we need to know which version of

the destination page is coherent. Thus, we have conducted simulations experiments on 60 real patterns from [BSG11]. This collection contains France TV channels web pages hourly crawls. [BSG11] collected versions of each page, over a day, and analyzed them to extract daily patterns as described in Section 3.3. With this collection we have daily patterns but also 1000 real page crawl with page changes used to estimate patterns. Although this collection contains daily patterns, it is still not applicable to our experiments because we do not have any information over links and their coherent destinations.

The experiment consists in simulating the navigation from a page source P_s to different destination pages P_d by following all outgoing links. However, the pages in the collection are not crawled by following hyperlinks but by using a predefined list of urls. Thus, there is no hyperlink information that we can use. Hence, we choose a random page from this dataset and we add to it 10 links to other randomly chosen versions of pages. In addition, all results are averaged over 1000 trials over 30 different size of time ranges (1 day to 30 days). We know which version of the destination page is coherent as we have a real web pages with their changes. To measure the effectiveness of the method, we count how many times the coherent version is chosen.

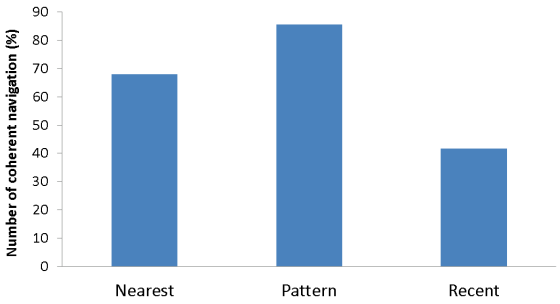


Figure 3.6: Results for coherence oriented navigation

This results presented in Figure 3.6 shows the percentage of coherent versions obtained by different strategies (*Pattern*, *Nearest*, *Recent*). Our navigation policy based on patterns outperforms its competitors *Nearest* and *Recent*. It improves the coherence by around 15 % compared to *Nearest* and by around 40 % compared to *Recent*.

3.6 Discussions and Outlook

This work addresses an important issue of improving coherence of archives under limited resources while navigating. Results of experiments have shown that our approach can improve the coherence during the navigation compared to related policies *Nearest* and *Recent*. To the best of our knowledge, this work is the first to exploit patterns to improve coherence of crawling and navigation.

This study is preliminary and new experimental works are needed. The proposed solution should be tested also over real data to have a better estimate. Our challenge is to enable users to navigate through the most coherent versions. Further study needs to be done to evaluate how far users can perceive coherence improvements when they navigate in the archive and to connect the metrics with user experience.

Part II

Optimization of Access to Web Archives

4 VI-DIFF: UNDERSTANDING WEB PAGES CHANGES

One of the major problems encountered by archiving systems is to understand what happened between two different versions of the web page. Furthermore, many applications (*e.g.* e-commerce or on-line trading sites) need to detect and discover changes on the web to help users understanding page updates and more generally, the web dynamics. In this chapter, we address this problem by proposing a new change detection approach that computes the differences between two versions of HTML web pages. Our Vi-DIFF solution can serve for various applications such as crawl optimization, archive maintenance, web changes browsing, etc. Experiments on Vi-DIFF were conducted and the results show that our approach is computationally reasonable.

4.1 Motivation

The Web is constantly evolving over time. Web content like texts, images, etc. are updated frequently. As those updates reflect the evolution of sites, many applications, nowadays, aim at discovering and detecting changes on the web. There are also many services [Bla10] that track web pages to identify what and how a page of interest has been changed and notify the concerned users. For instance, a browser plug-in [TDLH09] uses the page cache to explicitly highlight changes on the current page since the last visit. The goal of such approaches is to help users to understand what has changed on a web page and more generally, the web dynamics.

Detecting changes between page versions is also important for web archiving. Thus, many national archiving institutes [ACMS02, CLV04, Cat03, GSS06] around the world are collecting and preserving different web sites versions. Most often, web archiving is automatically per-

formed using web crawlers. A crawler revisits periodically web pages and updates the archive with a fresh snapshot (or version). However, it is impossible to maintain a complete archive of the web, or even a part of it, containing all the versions of the pages because of web sites politeness constraints and limited allocated resources (bandwidth, space storage, etc.). Thus, archiving systems must identify accurately how the page has been updated at least for two reasons:

- avoid archiving several times the same versions which is a specific issue for web archive crawlers
- model the dynamics of the web site in order to archive "the most important versions" by crawling the site at a "right moment" which is a general issue for all crawlers

Our research problem can be stated as follows: *How to know, to understand and to quantify what happened (and thus changed) between two versions of a web page ?*

Most of the pages on the web are HTML documents. As HTML is semantically poor, comparing two versions of HTML pages, based on the DOM[1], does not give a relevant information to understand the changes. Thus, our idea is to detect changes on the visual representation of web pages. Indeed, the visual aspect gives a good idea of the semantic structure used in the document and the relationship between them, *e.g.* web designers would put most important information in the center and put the navigation bar on the header or on the sides. Preserving the visual aspect of web pages while detecting changes gives relevant information to understand the modifications. It simulates how a user understands the changes based on his visual perception. The concept of analyzing the visual aspect of web pages is not new. However, as far as we know, it has never been exploited to detect changes on web pages.

Our proposed approach, called Vi-DIFF, compares two web pages in three steps: (i) segmentation, (ii) change detection and (iii) delta file generation. The segmentation step consists in partitioning web pages into visual blocks. Then, in the change detection step, the two restructured versions of web pages are compared and, finally, a delta file describing the visual changes is produced.

Our solution for detecting changes between two web pages versions can serve in various applications:

- **Visualization and browsing:** Our changes detection algorithm can be used to visualize the changes of web pages over time and navigate between the different archived versions. The visual changes would be explicitly displayed to help the user understand the relationship between the previous version and the current one. For instance, updated blocks can be framed with a different color to focus the user's attention on the updated content. Given

[1]The DOM presents an HTML document as a tree-structure. `http://www.w3schools.com/htmldom/`

two versions of web pages as shown in the Figure 4.1, the new inserted block in version 2 is framed in red. The corresponding blocks in the two versions that have an updated image are framed in blue. Several applications [JKN+06, LPT00, TDLH09] have been designed

Figure 4.1: An example of visual changes detection

to visualize changes in the browser by highlighting the content changes on pages such as a text update, delete, etc. However, to the best of our knowledge, no one explicitly view both the structural and content changes occurred on page such as a block insertion, move, update, etc. Our change detection approach can also be exploited to extend web browsers [TDLH09] that use the page cache to help users in understanding the changes on the current page since the last visit.

- **Web crawling:** The detected changes between web pages versions can be exploited to optimize web crawling by downloading the most "important" versions [BSGP09]. A relative importance of blocks can be assigned automatically by using for instance the algorithm of [Son04], or through others supervised machine learning methods. The evaluation of changes importance can be exploited to improve the web crawling by estimating how urgent it is to revisit a given page. This allows to keep the archive as complete as possible with the most important snapshots although allocated resources are limited.

- **Indexing and storage:** Web archive systems can avoid wasting time and space for indexing and/or storing some versions with unimportant changes. An appropriate change importance threshold can be fixed (*e.g.* through a supervised machine learning) to decide when the pages should be indexed or stored. Also, querying temporally the archive will take less time because indexing and storage can be efficiently performed by reducing the number of useless archived versions as explained before.

- **Archive maintenance:** Vi-DIFF can also be used after some maintenance operations to verify the quality of these operations in the web archive. The migration from the Internet Archive's file format ARC to WARC [2] is an example of maintenance operation. After the migration from ARC to WARC, our change detection Vi-DIFF can be exploited to ensure

[2]The WARC format is a revision of ARC file format that has traditionally been used to store Web crawls url. The WARC better support the harvesting, access, and exchange needs of archiving organizations.

that the page versions are kept "as they are", *i.e.* it will be possible to render them as when they were stored in ARC format. It is an important issue for digital preservation.

Contributions

The work presented in this chapter makes the following contributions:

- We propose a novel change detection approach (Vi-DIFF) that compares the two restructured versions of web pages by taking the page structure into account with a reasonable time complexity.
- We show through experiments the feasibility of the approach.

Vi-DIFF has been used in two projects: CARTEC (The French National Research Agency ANR Project ANR-07-MDCO-016) and SCAPE (Scalable Preservation Environments) project that is co-funded by the European Union under FP7 ICT-2009.4.1 (Grant Agreement number 270137)).

Organization

The remainder of this chapter is structured as follows. Section 4.2 discusses related works. Section 4.3 presents the VI-DIFF and the different change operations we consider. Section 4.4 explains, in detail, the second step of Vi-DIFF. Section 4.5, presents the implementation of VI-DIFF and discusses experimental results. Section 4.6 concludes.

4.2 Related Works

We convert our HTML pages into the XML pages by using a page segmentation algorithm to detect the changes. Hence, in this section, we describe the different change detection algorithms for XML files. Since XML documents can be represented as trees, we focus on tree oriented change detection algorithms. Various algorithms have been proposed with different complexities, memory usages, operations and delta formats. Some consider the trees as ordered, others consider as unordered[3]. We review the different change detection algorithms for ordered and unordered trees in the following.

[3] An ordered tree or plane tree is a rooted tree for which an ordering is specified for the children of each vertex. An unordered tree is one in which parent-child relationships are important, but there is no sibling order.

Algorithms for ordered trees :

The algorithm proposed in [ZS89] finds the tree edit distance [4] between two ordered labeled trees with basic operations (insert, delete and update). This algorithm has the best performance on correctness (detecting all the changes) in terms of quadratic time complexity. This algorithm is also used by Logilab XML Diff[5] and Microsoft XML Diff [6]. It does not support the move operation.

Chawathe's approach, LaDiff [CRGMW96], supports the move operation as well as the basic operations. Their proposed algorithm made the assumption that given two labeled trees T1 and T2, there is a "good" matching function compare, so that given any leaf s in T1, there is at most one leaf in T2 that is "close" enough to match s. However, this assumption does not hold for the documents that have duplicated or/and similar objects. Its cost is in $O(ne + e^2)$ where n is the total number of leaf nodes, and e is the weighted edit distance[7] between the two trees.

Cobéna et al. proposed the XyDiff algorithm [CAM02] which supports moves in addition to basic operations. The XyDiff algorithm computes hash values and weights for every node in the XML trees of both XML documents to compare. Then it compares the signatures of two nodes and if they are equal, the nodes are matched. It achieves a time complexity of$O(nlogn)$. However, because of the greedy rules used, it can not guarantee an optimal result and in some cases it can even mismatch subtrees.

Algorithms for unordered trees :

X-Diff [WDC03] is proposed for unordered trees. X-Diff integrates key XML structure characteristics with standard tree- to-tree correction techniques by using the notion of node signature together with a new matching between XML trees to find the minimum-cost matching and generate a minimum-cost delta script. The algorithm only specifies the three basic operations. The extension of this algorithm is used in *BioDiff* to detect changes in genomic and proteomic data specified as XML.

Chawathe also proposed an algorithm MH-Diff[CGM97] which handles moves and even copies besides updates, inserts and deletes. The copy and move operations result in an higher quality edit scripts, especially copied or moved sub-tree is large. It has, according to its authors, a heuristic solution with a worst-case $O(n^3)$ time, where n is the number of nodes, and an average

[4]The tree edit distance is defined as the minimum-cost sequence of node edit operations (insertion, deletion, substitution) that transform one tree into another.

[5]http://www.logilab.org/859 visited at 26 June 2013

[6]http://msdn.microsoft.com/en-us/library/aa302295.aspx visited at 26 June 2013

[7]Weighted edit distance is an edit distance where each edit operation has its own weight.

of $O(n^2)$ time. Besides all these algorithms, DeltaXML [LF01], which is a commercial tool, is the market leader. It can compare, merge and synchronize XML documents for ordered and unordered trees by supporting basic operations. Although it runs extremely fast and its results are close to minimum according to [CAM02], it has a limitation at level of tree size. It does not handle a move operation.

A detailed review on major approaches to detect changes on Web pages is proposed in [OS11]. A comparative study of different diff algorithms for XML documents is presented in [Pet05, CAH02]. Despite the variety of methods of comparison, there are no algorithms to detect changes on both ordered and unordered trees and in addition support all operations. Being able to detect changes both on ordered and unordered trees allows better flexibility when comparing different parts of web pages where the order can sometimes be considered relevant or irrelevant. Currently, there are no method that provides a semantic interpretation of changes in a Web page that is independent from HTML tags and scripts. Our aim is to detect changes in a way that the user observes changes visually. Thus, we propose our *ad-hoc* diff algorithm. As it is designed for one given specific type of document, it allows for a better trade-off between complexity and completeness of the detected operations set. Our proposed change detection algorithm detects structural and content changes. *Structural changes* alter the visual appearance of the page and the structure of its blocks. In contrast, *content changes* modify text, hyperlinks and images inside blocks of the page. Then, it constructs a delta file describing all these changes.

4.3 Vi-DIFF

In this section, we explain in detail the Vi-DIFF algorithm which detects structural and content changes between two web page versions. First, we explain how to convert HTML files into XML files, called Vi-XML, by using segmentation. Then, we describe the operations for content and structure on Vi-XML files and how to store them for further use.

4.3.1 Segmentation

The first step of our approach is the segmentation of the web page. Several methods have been proposed to construct the visual representation of web pages. Most approaches discover the logical structure of a page by analyzing the rendered document or the document code. Gu et al. [GCMC02] propose a top-down algorithm which detects the web content structure based on the layout information. Kovacevic et al. [EDG+02] define heuristics to recognize common page areas (header, footer, center of the page, etc.) based on visual information. Cai et al. [CYWM03b] propose the algorithm VIPS which segments the web page into multiple semantic blocks based on visual information retrieved from browser's rendering. Cosulshi et al. [CMM04] propose an approach that calculates the block correspondence between web pages by using positional

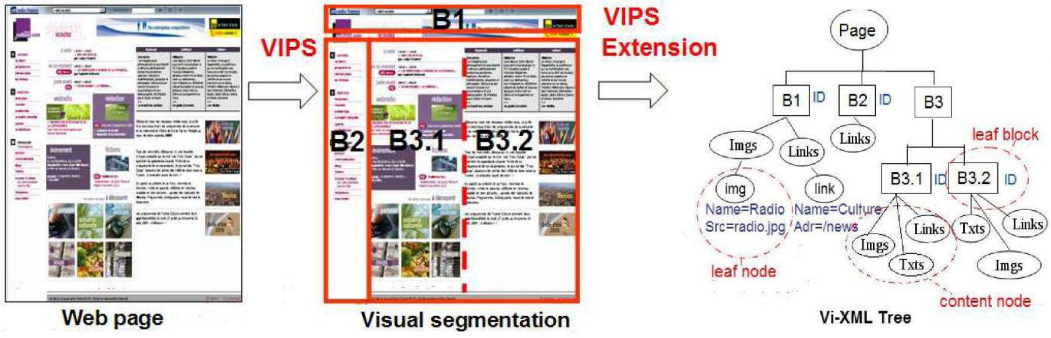


Figure 4.2: VIPS Extension

information of DOM tree elements. Kukulenz et al.'s automatic page segmentation [KRH08] is based on the estimation of the grammatical structure of pages automatically.

Among these visual analysis methods, the VIPS algorithm [CYWM03b] seems to be the most appropriate for our approach because it allows an adequate granularity of page partitioning. It splits the Web page into smaller parts based on the DOM and visual cues such as split lines, decorative images, colors, and fonts. The algorithm uses the content source and tries to simulate how people understand web page layout. It has three main steps: block extraction, separator detection and content structure construction. First, the visual block extraction is performed from the root node of the DOM tree based on visual clues. The authors propose the vision-based content structure, where every node, called a block, is a basic object or a set of basic objects (*e.g.* the leaf node in the DOM tree that can not be decomposed anymore). The web page is firstly segmented into several big blocks. For each of these blocks, the same segmentation process is carried out recursively until all appropriate nodes are found to represent the visual blocks. Second, visual separators (here, separators denote the horizontal or vertical lines in a web page that visually cross with no blocks) among adjacent blocks are identified, and the tree structure of the blocks is constructed. Then, based on detected separators the blocks are merged and represented as a node in the result tree. However, VIPS returns only the XML tree describing the structure of the page without any content information on it. To complete the visual structure of the page, we use the extended VIPS [BSGP09] that also extracts links, images and texts for each block as shown in Figure 4.2. The complete hierarchical structure of the web page is described in an XML document called *Vi-XML*.

We define, three types of nodes for Vi-XML as shown in Figure 4.2:

- a *leaf block* is a block which has no child blocks
- a *content node* is a child of a *leaf block* that contains set of *leaf node*s
- a *leaf node* is the nodes without a child such as image, link.

These three types of nodes are uniquely identified by an ID attribute. This ID is a hash value computed using the node's content and its children node content. Content nodes have also an attribute called IDList which has list of all IDs of its leaf nodes. Each block of the page is referenced by the attribute Ref which contains the *Dewey Identifier* of the absolute location of the block generated by VIPS. The Dewey ID of nodes are assigned in the following way: the relative position of each node among its siblings are recorded and the concatenation of these relative positions using a separator *e.g.* dot ".", "-" starting from the root composes the Dewey ID of nodes. For example, the block with Dewey ID, 1-2-1 is the first child of its parent 1-2. Leaf nodes have other attributes such as the name and the URLs for links. An example of Vi-XML document is shown on Figure 4.3.

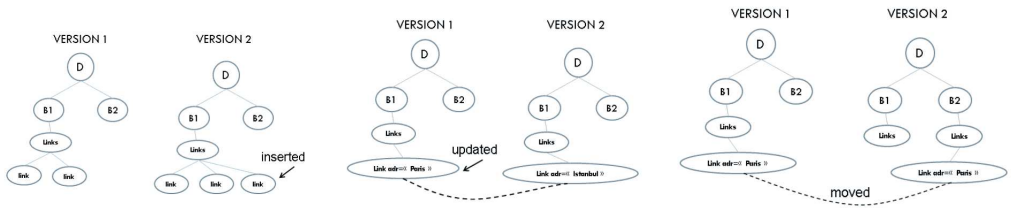
```
<Block Ref="Block1-2-2" ID="f0dd5cbfbace186c1d784a9e6b9fc783" Pos=" H:10 W:928 T:938 L:46">
    <Links ID="fc2175af65bcaa1f4b69b2ee3bff6e35" IDList="044abfe8a320fd384bba768ccc78884d,...
      <link ID="044abfe8a320fd384bba768ccc78884d" Name="Suivez tout sur la crise financière" ...
      <link ID="f18b2cd9982ba5abccb83bede6db9ca4" Name="Aide" Adr="http://help.yahoo.com/l/fr" />
    </Links>
    <Txts ID="778cb870ed54bf8cfd98af0ebf5a27e5" Txt="  Copyright © 2009 Yahoo! Tous droits réservés. " />
</Block>
```

Figure 4.3: Vi-XML file example

4.3.2 Change Operations

In this section, we give the description of operations which are used by Vi-DIFF to describe the changes between two Vi-XML files. As mentioned in Section 4.2, we consider two types of changes: content changes and structural changes. Detecting structural changes allows us to have a faster algorithm and more compact delta files.

• **Content change operations** Content changes are modification of the content nodes in blocks. They are modeled at the leaf node level for links, images and texts in Vi-XML files through the following operations:



(a) Insert (b) Update (c) Move

Figure 4.4: Content changes operations

- *Insert(x(name, value),y):* Inserts a leaf node *x*, with node name and node value, as a child

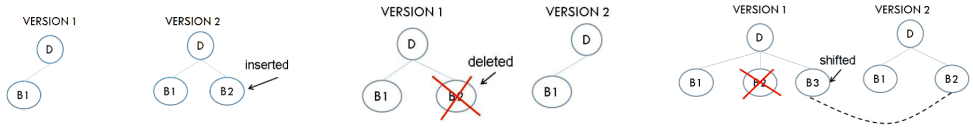
node of content node *y* as shown in Figure 4.4a

- *Delete(x):* Deletes a leaf node *x*.
- *Update(x,attrname,attrvalue):* Updates the value of *attrname* with *attrvalue* for a given leaf node *x* as shown in Figure 4.4b.
- *Move(x,y,z):* Moves a leaf node *x* from content node *y* to content node *z* as shown in as shown in Figure 4.4c. Most of the existing algorithms treat the move operation as a sequence of delete and insert operations. The use of a move operation can have a great impact on the size of the delta script and on its readability. It reduces the file size but increases the complexity of the diff algorithm.

Delete, Update and Insert operations are considered as basic operations and existing algorithms support those operations. Move is only supported by few approaches [CRGMW96, CGM97] as discussed in Section 4.2.

• Structural changes operations

The structural changes are modifications of the leaf block nodes in Vi-XML files. Detecting structural changes decreases the size of the delta file and makes it easier to query, store and manage web archives. Most of all, it provides a delta much more relevant with respect to what visually happened on the web page.



(a) Insert (b) Delete (c) Move

Figure 4.5: Structural changes operations

The operations are listed as follows:

- *Insert(Bx,y,Bz):* Inserts a leaf block subtree Bx, which is rooted at *x*, to the (leaf) block node *y*, after (leaf) block node Bz as shown in Figure 4.5a.
- *Delete(Bx):* Deletes a leaf block rooted at *x* as shown in Figure 4.5b.
- *Move(Bx,By):* Moves sub-tree rooted at *x* (Bx) to the node *y*. After insert and/or delete operations, the nodes are moved (shifted) as shown in Figure 4.5c.

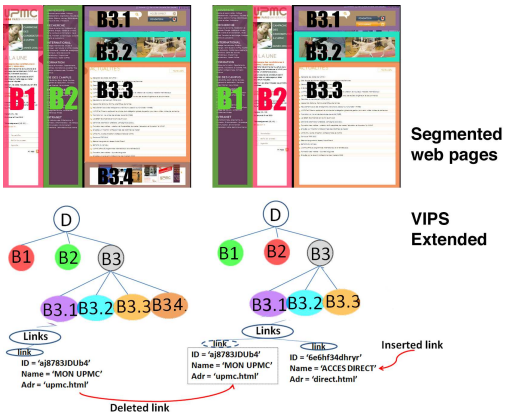
4.3.3 Delta files

Detected changes are stored in a delta file. We use XML to represent delta files because we need to exchange, store and query deltas through existing XML tools. The delta files contain all operations (delete, insert, update, move, etc.) that transform one version into another one. The final result after applying delta file to the old version should be exactly the new version.

As far as we know there is no standard for delta files, hence we decided to build our own format.

The content changes are represented by operation tags (delete, insert, update, move) as children of their parent block's tag. The structural change operations such as delete and insert are added directly to the root of Vi-Delta. For move operation in structural changes, a newRef attribute is added to the related block, so that we can keep track of a block even if it moves inside the structure.

Figure 4.6 shows an example of Vi-Delta. It contains three move and one insert as structural changes operations, one delete and one update as content changes operations. We can see, in this example, how detecting structural changes makes delta file easier to query, to manage and to store. For instance, in our example, instead of deleting and inserting all the content of moved blocks (B1, B2, B3), the structural move operation is used.




```
<xml>
<Delta FROM="V1" TO="V2">
  <Delete>
      <Block Ref="B3.4" />
  </Delete>
  <Block Ref="B2" newRef="B1" />
  <Block Ref="B1" newRef="B2" />
  <Block Ref="B3.1">
      <Delete>
          <link Name="MON UPMC"
          Adr="upmc.html"/>
      </Delete>
      <Insert>
          <link Name="ACCES DIRECT"
          Adr="direct.html"/>
      </Insert>
  </Block>
</Delta>
</xml>
```

Figure 4.6: Vi-Delta

4.4 Change Detection

The Vi-DIFF algorithm has three main steps:

- *Segmentation*: Web pages are converted to XML files (Vi-XML) as discussed in 4.3.1.
- *Change Detection*: The structural and content changes described in 4.3.2 between Vi-XML files are detected.
- *Delta file*: The changes detected are saved in Vi-Delta files as described in 4.3.3.

In this section, we give the details of Vi-DIFF second step. The algorithm contains a main part and two sub-algorithms: one for detecting structural changes and one for detecting content changes.

4.4.1 Main Algorithm

The main algorithm detects basic structural changes. It uses the ID of each block, that represents its content, to compare the source and destination files. This can be done efficiently by sorting blocks by their ID value first. When a block is present in both the source and destination file, we check its Dewey ID to see whether it has been moved. Blocks whose IDs are not matched are then passed to the content change detection algorithm for further processing.

Algorithm 1 Vi-DIFF

Input: Vi-XML Tree1, Vi-XML Tree2
Output: Vi-Delta
1: Traverse Tree1, get list of blocks B1
2: Traverse Tree2, get list of blocks B2
3: Create Delta Tree DT
4: DetectStructuralChanges (B1, B2, DT)
5: Save DT

4.4.2 Detecting Structural changes

For blocks whose IDs are not matched, we introduce the notion of the block distance. This allows to further restrict the list of blocks for which we will explore the content changes. To do this, we compare the list of IDs (hash values) of contained contents (*i.e.* images,links and texts).

The distance measure of two blocks is defined as a function of the distance between their content nodes. We define the distance between two leaf block nodes B1 and B2 as:

$$Distance(B1,B2) = 1 - \frac{\sum_{x \in Elements} Jaccard(B1(x),B2(x))}{|Elements|}$$

where $Elements = \{Links, Images, Text\}$, is a set of IDs of its leaf nodes for element x in block B. The Jaccard similarity coefficient measures similarity between sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. If Distance(B1,B2) is greater than a threshold γ, the two blocks are considered as distinct, otherwise, they are considered as similar. This threshold allows to consider that no change has occurred if two blocks are similar enough. The blocks that are considered as distinct are kept separately for the source and the destination file and are passed to DetectingContentChanges algorithm detailed in the next section.

4.4.3 Detecting Content Changes

The content changes are at the level of a leaf node, for the children nodes of Links, Images and Texts. As Links and Images have nearly the same structure, they are treated in the same way. For Texts, we compute text similarities, and treat them separately.

Links and Images

A "link" element (resp. a "image" element) in Vi-XML file has four attributes:

- *Name* holds the anchor text of the link (resp. the caption for image)
- *ID* holds the hash value of attributes values
- *Source* holds the hyperlinks
- *BlockRef* holds the Dewey ID of its block

Two leaf nodes are equal if their IDs are equal. There is two kind of updates for leaf nodes: update on Name attribute or update on Source attribute. Two nodes with the same Name but different Source or the same Source but different Names are considered as updated. Leaf nodes with different BlockRefs are moved nodes.

By traversing two list of blocks in one iteration by comparing their IDs, we obtain all leaf nodes for "Links" (resp. for "Images") content node where IDs are different. We have two lists of all leaf nodes: one for source, one for destination. An efficient way of comparing these lists is to use merge sort algorithm based on Name attribute and traverse both lists at the same time. Each time we detect an operation, we remove the appropriate nodes from their corresponding lists. At the end of our iteration, all the nodes left on the list for v1 are considered deleted and all the nodes left on the list of v2 are considered as inserted.

Algorithm 2 CompareLeafNodes

Input: LeafNode1, LeafNode2, Type
Output: DeltaTreeNode

```
1:  if Type != Text then
2:     if LeafNode1.ID == LeafNode2.ID AND
       LeafNode1.BlockRef != LeafNode2.BlockRef then
3:        MOVE detected
4:     else if (LeafNode1.BlockRef == LeafNode2.BlockRef) AND
       ((LeafNode1.Adr/Src != LeafNode2.Adr/Src) OR
       (LeafNode1.Name != LeafNode2.Name)) then
5:        UPDATE detected
6:     end if
7:  else
8:     if DistanceText > ϕ then
9:        INSERT detected
10:       UPDATE detected
11:    end if
12: end if
13: Create DeltaTreeNode based on detected change
14: Return DeltaTreeNode
```

Texts

We compute the distance between two texts to decide if it is an update operation or a delete+insert operation. For this, we compute the proportion of distinct words between two versions. If this value is more than a threshold, they are considered as different texts and we have two operations (delete then insert). Otherwise, the text is considered as updated. Each text in each block is compared separately.

4.4.4 Complexity analysis

We now briefly analyze the complexity of our algorithm. This issue is important because we do not want that page processing becoming a bottleneck for our system.

We give the worst case complexity in which all blocks of a page changed from one version to another without any structural changes. In this case, the Vi-DIFF algorithm has an $O(nlog(n))$ time complexity, where n is the total number of leaf nodes. The complexity of the traversing each Vi-XML tree to get list of leaf nodes is $O(n)$. To compare two list of leaf nodes, it is $O(nlog(n))$ due to the merge sort algorithm. Thus, our Vi-DIFF algorithm without detecting structural changes is log-linear and achieves a time complexity of $O(nlog(n))$ which is rather

Algorithm 3 DetectContentChanges

Input: Source,Version:list
Output: DT:Delta Tree

```
1: Sort both lists with merge sort by attribute "Name"
2: counterS,counterV
3: while true do
4:    objSource = Source[counterS]
5:    objVersion = Version[counterV]
6:    DeltaTreeNode = CompareLeafNodes(objSource,objVersion,type)
7:    remove objSource, objVersion from their arrays if any operation detected
8:    if End of Source then
9:       INSERT detected for all remaining elements in Version
10:   end if
11:   if End of Version then
12:      DELETE detected for all remaining elements in Source
13:   end if
14:   Add detected operations to DT
15:   if |Source| = 0 and |Version| = 0 then
16:      break
17:   end if
18: end while
19: return DT
```

promising. If there are structural changes, the worst case complexity, *i.e.* if the whole structure is changed, is $O(mlogm)$ where m is the total number of blocks in source and destination file. The web pages are usually rather small and the cases where a page completely changes from one version to another one are very rare. In order to verify this, we measured the actual complexity through experiments described next.

4.5 Experiments

Experiments are conducted to analyze the performance (execution time) and the quality (correctness) of the Vi-DIFF algorithm.

4.5.1 Segmentation

Visual segmentation experiments have been conducted over HTML web pages by using the extended VIPS method. We measured the time spent by the extended VIPS to segment the page

and to generate the Vi-XML document. We present here results obtained over various HTML documents sized from about 20 KB up to 600 KB. These represent only sizes of container objects (CO) of web pages and thus do not include external objects like images, video, etc. According to [Kin08], the average size of web pages was about 312 KB (50 % of it is the size of CO) in 2008. Thus, the CO's size of the average web page is around 156 KB that we can use as a reference to analyze our results.

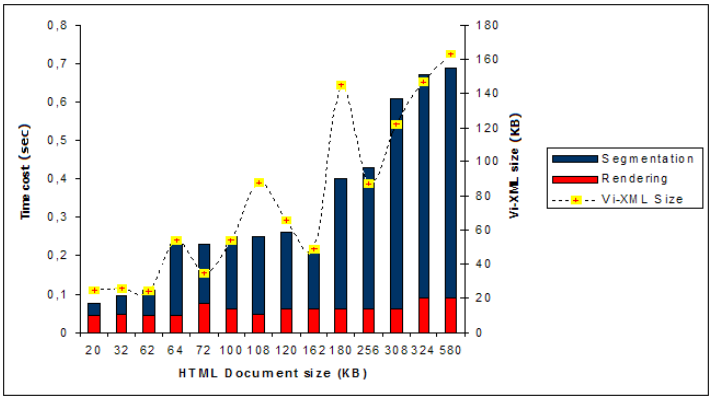


Figure 4.7: Segmentation Time

We measured the performance of the segmentation in terms of execution time and output size. Experiments were conducted on a PC running Microsoft Windows Server 2003 over a 3.19 GHz Intel Pentium 4 processor with 6.0 GB of RAM. The execution time for the browser rendering and the visual segmentation is shown in Figure 4.7. The horizontal axis represents the size of HTML documents in KBytes (KB). The left vertical axis shows the execution time in seconds for each document. The time for rendering is almost constant, about 0.1 seconds. The execution time of the visual segmentation increases according to the size of HTML documents. The time of the segmentation is about 0.2 seconds for documents sized around 150 KB which represents the average size of CO in the web. This execution time seems to be a little bit costly but is counterbalanced by the expressiveness of the Vi-XML file that really correlates with the visual aspect of web pages. Nevertheless, this time cost should be optimized. The main idea for that purpose is to avoid rebuilding the blocks structure for a page version if no structural change has occurred since the former version. We are currently trying to find a method that detects directly changes inside blocks based on the visual structure of previous versions of the document.

The right vertical axis in Figure 4.7 shows the the size of the output Vi-XML file with respect to the size of the original HTML document. From this experiment, we can observe that the Vi-XML document size is usually about 30 to 50 percent less than the size of the original HTML document (for those sized more than 100 KB). This is interesting for the comparison of two Vi-XML documents since it can help to reduce the time cost of changes detection algorithm.

4.5.2 Change Detection

We conducted two types of preliminary experiments in this section: one without structural changes and another with structural changes. For this section, tests are realized on a PC running Linux over a 3.33GHz Intel(R) Core(TM)2 Duo CPU E8600 with 8 GB of RAM.

Without structural changes

To analyze the performance and the quality of the second step of Vi-DIFF algorithm, we built a simulator that generates changes on a Vi-XML document. The simulator takes a Vi-XML file and generates a new one according to parameters given as input, as shown in Figure 4.8. It also generates a reference delta file, called Vi-Sim-Delta. The two Vi-XML files are compared with the Vi-DIFF algorithm and a Vi-Delta is generated. This simulator takes as parameters the percentage of changes for each type of content (Links, Images,Texts) and for each operation (insert/update/delete/move).

For example, for content Links and operation Insert (Links-Insert), 100% is given as input parameter. The simulator takes the number of links (n) in each block, and doubles by adding n new links, and Images-Update 50% takes the number of images (m) in each block and updates m/2 images. For the move operation, the content is always moved to next block. If it is the last block, it is moved to first block.

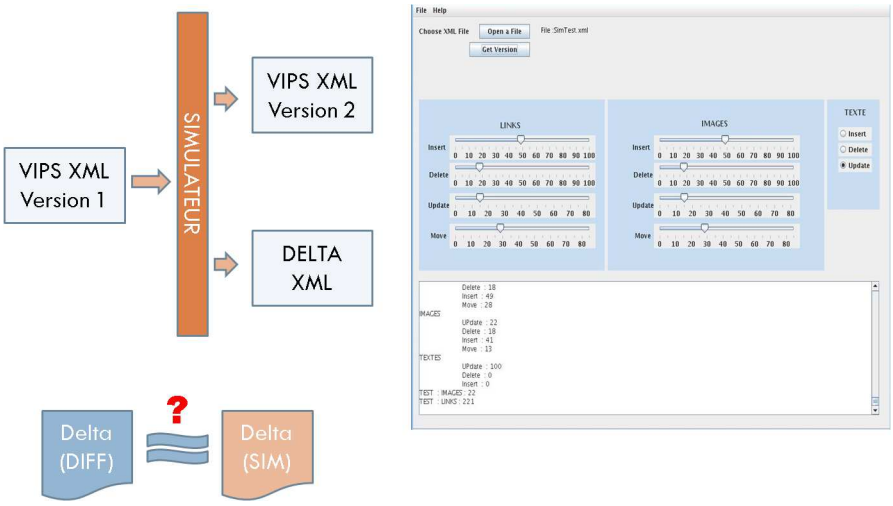


Figure 4.8: Simulator

For the test without structural changes, different versions are obtained from a crawled web page http://www.france24.com/fr/monde by using our simulator. For each type of operation (delete, insert, update, move) the change rate is increased by 10% until reaching 100%. It means that

the last version with 100% change rate is completely different from the original one. As links and images are treated the same way, we observe that the execution time is the same on average. To simplify the figure's readability, we only give the results for links. As the change detection part is measured in milliseconds, processor events may cause some variability on the processing time. Thus, all the process is executed 10000 times and the results are averaged. The results are presented in Figure 4.9.

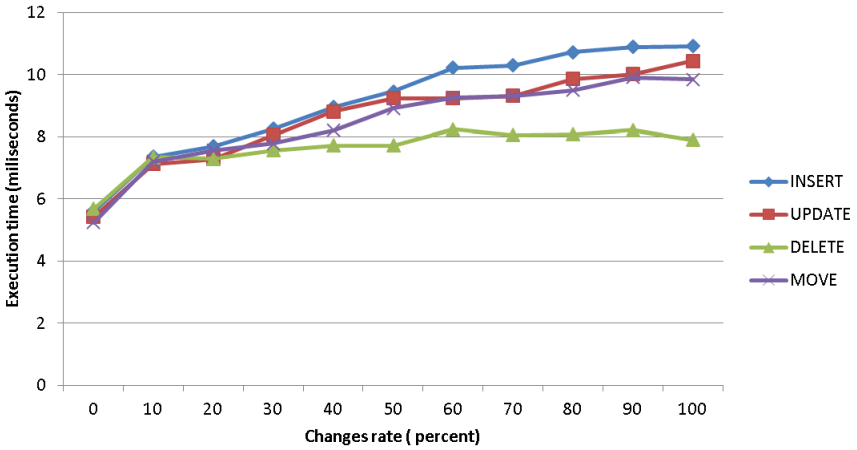


Figure 4.9: Change Detection Execution Time

As we can see, move and update operations have nearly the same execution time because they do not change the size of Vi-XML files, but this is not the case for insert and delete operations. The execution time increases with the insert operation rate (more objects to compare), and decreases with the delete operation rate (less objects to compare). Also a profiler analyzer is used to see the details of execution time. In average, 70% of the time is used to parse Vi-XMLs and to save Vi-Delta, 10% is used to detect if there is a structural changes, 20% is used to detect changes.

With structural changes

In order to do experiments with the processing time related to the structural changes, as the simulator does not support the structural changes for instance, the versions with the structural changes are obtained by modifying the segmentation rate (degree of coherence) in the VIPS algorithm. It can be seen as "merge and split"of blocks which change the Vi-XML's structure. We do our test by using hourly crawled web pages from http://www.cnn.com. This page is selected because it is updated frequently and contains several blocks. To give an idea about the baseline environment, we first compare two identical Vi-XML files. The execution time is 7.3 ms. Then, two versions hourly crawled are segmented with different segmentation rate into two Vi-XML files. These files are compared. The execution time is 9.5 ms and the generated Vi-Delta size is 1.8 Kb. To show the benefit of detecting structural changes in terms of the execution time and delta file size, the Vi-DIFF algorithm was modified so it does not detect structural changes.

Then,the same Vi-XML files are compared. The execution time is 48 ms and the generated Vi-Delta size is 40 Kb. Because with structural change detection, we try to detect changes in content only for similar blocks so the change detection is faster. Also, a smaller delta file is easier to store, to read and to query.

According to our preliminary experiments, Vi-DIFF is able to detect basic operations and move (without structural changes) correctly between two web page versions. For the CNN web page, the total execution is between 0.1 seconds and 0.8 seconds depending on the page versions' size. The execution time for change detection step is satisfying as it allows to process about one hundred pages per second and per processor.

Correctness of Vi-Diff

Checking the correctness of the algorithm is merely a matter of verifying that it is able to detect all changes between two versions (when $\gamma = 0$). To check the correctness of change detection step, we compared all Vi-Delta et Vi-Sim-Delta files with the DeltaXML trial version [LF01]. Vi-Delta and Vi-Sim-Delta files are always identical which shows the correctness of this step of Vi-DIFF. As the simulator is not able to make structural changes for the moment, the correctness of delta file for this section was checked manually and we observed that Vi-DIFF is able to detect all structural changes.

4.6 Discussions and Outlook

In this chapter, we proposed Vi-DIFF, a change detection method for web pages versions. Vi-DIFF helps to understand what happened and changed between two page versions. It detects semantic differences between two web pages based on their visual representation. Preserving the visual structure of web pages while detecting changes gives relevant information to understand modifications which is useful for many applications dealing with web pages. Vi-DIFF consists of three steps: (i) segmentation, (ii) change detection and (iii) generation of delta file. It detects two types of changes: structural and content changes. Structural changes (insert, delete, move) alter the block structure of the page and its visual appearance. Content changes (insert, delete, update) modify links, images, texts. In addition to basic operations, it supports a move operation, if there is no structural changes. However, it does not support yet the move of content changes when there are structural changes. This should be handled by future versions.

The proposed Vi-DIFF algorithm generates, as output, a Vi-Delta file which describes change operations that occurred between the two versions. The structure of the produced Vi-Delta helps to visually analyze the changes between versions. It can serve for various applications: web

crawl optimization, archive maintenance, historical changes browsing, etc.

A simulator was developed to test the performance of the algorithm. Experiments in terms of execution time were conducted for each step. The execution time for change detection step is very satisfying, but the execution time cost for the segmentation step must be optimized.

5 A Comparison of Static Index Pruning Methods with Temporal Queries

Temporal queries combine textual and temporal constraints and are used for searching temporal collections like web archives. Indexing these collections might result in huge index files which can reduce the performance of query processing. Static index pruning can be used to increase efficiency. However, it remains unclear whether these methods are adapted for temporal queries. More specifically, we hypothesize that pruning should preserve the temporal diversity of the collection. In this chapter, we compare the effectiveness of four static index pruning methods for temporal collections and show the relation between temporal diversification and retrieval effectiveness. Evaluations are conducted using the TREC and Wikipedia collections.

5.1 Motivation

Time evolving content from the web (*e.g.* web archives, news archives and wikis), is attracting more and more attention as acknowledged by national libraries and non-profit organizations who have been crawling the web since the late 90s. Querying these collections requires the use of temporal queries that combine temporal constraints with standard keyword queries. However, the growing size of temporal collections (for example, Internet Archive crawled over 150 billion web pages since 1996) implies a growing size of the corresponding index which can become a bottleneck for query processing.

To solve this problem, different compression techniques were proposed. *Index pruning* is one of them which reduces the index size by removing information (postings) from the index. Index pruning can be *dynamic* or *static*. Dynamic pruning depends on the query and prunes during query processing by deciding which entries should be involved in the ranking process and if the ranking process should stop or continue and does not reduce the index size. Static pruning reduces the index size by removing the entries from the index off-line, in advance, independently from any query. It is a lossy index compression technique because it is not possible to bring the compressed data back to its exact uncompressed form. The challenge is to decide which information to prune without reducing significantly the retrieval effectiveness. The two approaches are somehow complementary and in this chapter we focus on static index pruning.

None of the existing pruning techniques deals with the problem of temporal collections and using existing pruning techniques may lead to loose postings from different time periods. Consider the example of the temporal query "Iraq War in 1991". If the documents mentioning "Iraq War in 2003" are more relevant than the documents mentioning "Iraq War in 1991", the index after pruning will have less or no relevant documents related to the first query. Ideally, the pruning methods should keep the relevant documents from different time periods, *i.e.* they should preserve the *temporal diversity* of postings. As the existing index pruning techniques were not designed with time in mind, it is not clear if they are adapted for such a task. Investigating whether they are adapted and, if not, how they fail, is a necessary step before working on new temporal index pruning methods.

In this chapter, we provide a comparison of four state-of-the-art static index pruning methods [CCF+01, CLTH12, TC11, BB10a] in terms of retrieval effectiveness.

Contributions

Our main contributions in this chapter are as follows:

- We perform the first study on an empirical comparison of static index pruning methods for temporal queries by conducting extensive experiments, we compute the performance of different static index pruning methods using the same datasets, TREC-LATimes and Wikipedia collections.
- We show the relation between temporal diversity and the retrieval effectiveness based on correlation analysis.

Organization

The remainder of this chapter is organized as follows. In the next section, we give an overview

of related work. In Section 5.3, we present in detail four different static index pruning methods and conduct experiments in order to evaluate those methods in Section 5.5. Finally, in Section 5.6, we conclude this chapter.

5.2 Related Works

In ad-hoc information retrieval (IR), the goal is to retrieve the most relevant documents according to a topic or query. In order to rank, a score of relevance is computed for each document. Indexes are used to compute these scores efficiently at query time. Static index pruning reduces the index size by discarding the postings that are less likely to affect the retrieval performance. The seminal work of Carmel et al. [CCF+01] was based on two simple ideas: each posting is associated to a score, and a threshold on this score is defined to filter out a part of the postings. The main differences between all the approaches that came latter comes from the way the score and the threshold are defined. Scores can be related to IR weighting schemes or relevance estimates.

In the case of Carmel [CCF+01], the score is the TF-IDF value associated with a posting. De Moura et al. [MSA+08] proposed to use term co-occurrence information. More recently, Blanco et al. [BB10b] used the the ratio of the probability that a document is relevant to a query over the probability that a document is not relevant to a query (odd-ratio) is used as a decision criteria. Another way to score postings is to use statistical tools. Chen et al. [CLTH12] proposes to use a measure related to the contribution of a posting to the entropy of the distribution of documents given a term. Thota et al. [TC11] compare the frequency of occurrences of a word in a given document to its frequency in the collection. They use the two sample two proportion statistical test to decide whether to keep a posting or not.

With respect to the thresholds, Carmel et al. [CCF+01] defines a threshold that depends on each term in a way that preserves at least the top-k postings of this term (when ranked by their score). A limitation of this approach is that it may retain too many postings for unimportant terms (those that will not be important when ranking documents). In this case, it is possible to use a global threshold (*uniform pruning*) for all the terms like in [TC11, CLTH12].

There are other approaches to pruning that are not variations of Carmel's one [CCF+01]. There is no guarantee to keep postings for each term with these approaches, they can remove all postings for a term or they study the contribution of a term to the documents ranking.

Instead of ranking the postings per term, postings can be ranked by document. Büttcher and Clarke [BC06] introduced another pruning method that removes the posting lists based on the documents. For each document in the corpus, the terms of the document are ranked based on

their contribution to the Kullback-Leibler divergence between the distribution of terms within the document and the collection. Then, only the postings for the top-k terms in the document are kept in the index.

Another way to prune is to remove whole postings lists for a subset of terms or documents. Blanco et al. [BB07] proposed to remove entire posting list based on the informativeness value of a term, in particular inverse document frequency (idf) and residual inverse document frequency (ridf), and two methods based on Salton's term discriminative model [SYY75], which measures the space density before and after removing a dimension (term). This approach may remove entire terms from the index and should be reserved to a small subset of the vocabulary. Finally, it is possible to fully remove a document from the postings – this is related to the problem of how retrievable a document is [AV08]. Zheng and Cox [ZC09] suggested an entropy based document pruning strategy where a score is given to each each document based on the entropy of each of its terms. Documents below a given threshold are then removed from the index.

In [AOU09], the seminal work of Carmel et al. [CCF+01] that ranks postings by term is compared to another seminal work ranking postings by documents [BC06]. The results showed that ranking postings by term provides a better retrieval effectiveness.

5.3 Methods

In this section, we thus focus on four static index pruning methods that rank postings by term but which use different decision criterias (score and threshold) to prune. We selected them because of their diversity and state-of-the-art results. We next briefly explain these four methods.

Carmel Top-K Pruning (TCP)

The work of Carmel et al. [CCF+01], TCP in short, is a pioneer work. This method is one of the major approach in static index pruning and its different variations can be also found in [MSA+08, BB10a].

As shown in Algorithm 4, for each term in the vocabulary of an index I, their algorithm computes the score of the documents (based on TF-IDF in the original paper) in the posting list. Then, for a term t, the method selects the k^{th} highest score, z_t, and sets the threshold value $\tau_t = \varepsilon * z_t$, where ε is the parameter used to control the pruning rate. Each entry in the term t posting list whose score is lower than τ_t is considered not important and removed from the posting list.

Algorithm 4 Carmel Top-k prune

Input: I index, k , ε
Output: I_p pruned index

```
R empty list
for all term t in I do
    retrieve the posting list P_t from I
    if |P_t| > k then
        for all entry d ∈ P_t do
            compute score s_d for d
            R ← R ∪ (d, s_d)
        end for
        sort R
        let z_t be the k^th highest score in R
        τ_t = ε * z_t
        for all entry d ∈ P_t do
            if s_d ≤ τ_t then
                remove entry d from P_t
            end if
        end for
    end if
    I_p ← I_p ∪ P_t
end for
I_p
```

Information Preservation based Pruning (IP-u)

Chen et al. [CLTH12] proposed a new static index pruning method, IP-u in short, based on the notion of information preservation. By considering an inverted index as a non-parametric predictive model $p(d|t)$, they concluded that pruning this model causes a loss in its predictive power. They proposed to use conditional entropy $H(D|T)$ to quantify the predictive power and minimize the loss after pruning. The conditional entropy is written as:

$$H(D|T) = -\sum_{t\in T} p(t) \sum_{d\in D} p(d|t) log p(d|t) = \frac{1}{|T|} \sum_{t\in T} \sum_{d\in D} A(d,t)$$

$$\text{where } A(d,t) = -\frac{p(t|d)p(d)}{\sum_{d'} p(t|d')p(d')} log \frac{p(t|d)p(d)}{\sum_{d'} p(t|d')p(d')}$$

where D is a set of documents and T is a set of terms, p(t) is the probability of a term being used in a query. The distributions p(t) and p(d) are uniform. in the original work for IP-u [CLTH12]. The prune ratio is controlled by defining a threshold ε. All the entries whose uncertainty A(t,d)

is strictly lower than ε are discarded. Note that in contrast to TCP, this might remove whole posting list from the index.

Two-sample two-proportion based Pruning (2N2P)

Thota et al. [TC11], 2N2P in short, use statistical methods to decide which information to prune. The 2N2P test, a two-proportion Z-test, is used to compare differences between two random samples.

In the context of pruning, the authors compare the distributions of a term in the document and in the collection. If they differ too much, it is necessary to preserve the corresponding posting in the index. Otherwise the collection approximation is enough. Formally, they compute the statistic

$$Z = \frac{\frac{tf_{t,d}}{|d|} - \frac{tf_t}{|C|}}{E} \text{ with } E = \sqrt{P(1-P)(\frac{1}{|d|} + \frac{1}{|C|})}$$
$$P = \frac{tf_{t,D} + ctf_t}{|d| + |C|}$$

where $tf_{t,d}$ is the term frequency in the document d of length $|d|$ and ctf_t is the term frequency in the collection C of length $|C|$ (total number of term occurrences). E corresponds to the standard deviation. Like in the previous approach a threshold ε is used to control the prune ratio. Two different approaches are implemented for this method in [CLTH12]. In this chapter, we use the one with a constant threshold of Z irrespective of the documents length.

Probability ranking principle based Pruning (PRPP)

Probability ranking principle, PRP in short, introduced by [Rob77], states that documents are to be ranked "in order of decreasing probability of usefulness to the user". As stated in [Rob77], the principle is based on the following two assumptions:

> *i) The relevance of a document to a request is independent of the other documents in the collection;*
> *ii) The usefulness of a relevant document to a requester may depend on the number of relevant documents the requester has already seen (the more he has seen, the less useful a subsequent one may be).*

Given a document and a query, represented by the random variables D and Q, a probabilistic IR model calculates the probability of D being relevant $p(r|D,Q)$ and non relevant $p(\bar{r}|D,Q)$. It was shown in [vR79] that the ranking produced by the odds-ratio of relevance and non relevance is equivalent to the ranking produced by PRP.

Blanco et al. [BB10a] use this principle for index pruning purposes. The decision criteria is the

ratio of the probability that a document is relevant to a query to the probability that a document is not relevant to a query (odds-ratio). Given a document d, a query q, the classes of relevance r and non-relevance $\bar{r}$, the probability ranking principle suggests that we should rank the documents in decreasing order of probability ratio $p(r|d,q)/p(\bar{r}[d,q)$. By assuming that the document d is independent of the query q and that query terms occur independently from each other (*term-independence assumption*) [RvRP80], this gives:

$$\frac{p(r|d,q)}{p(\bar{r}[d,q)} = \frac{p(q|d,r)}{p(q|\bar{r})} \frac{p(r|d)}{1-p(r|d)}$$

Every term in the vocabulary is considered as a single term query. As with other approaches, if the score of term-document pair is smaller than a threshold ε, it is discarded. To compute this ratio, the estimation of three probabilities is needed.

1. $p(t|d,r)$, the probability of a term given a document and relevance. This query likelihood is estimated by following a language model[1] with Jelineck-Mercer smoothing by approximating $p(t|d,r)$ by $p(t|d)$.

$$p(t|d) = (1-\lambda)p_{mle}(t|d) + \lambda p(t|C) \text{ where } \lambda \in [0,1]$$

2. $p(r|d)$, the prior probability of the relevance of a document. It is estimated as follows:

$$p(r|d) = \frac{1}{2} + tanh(\frac{dl - mean_{dl}}{sd_{dl}}) * \frac{1}{10}$$

   where dl is the document length, $mean_{dl}$ is the mean, tanh is the hyperbolic tangent function and sd_{dl} is the standard deviation of document length in the collection.

3. $p(t|\bar{r})$, the probability of a term given the non-relevance. This probability is assumed to be equal to collection background model $p(t|C)$ and estimated through maximum likelihood estimation.

Random Pruning

As a lower bound baseline, we implemented a random pruning method. For each term, we randomly choose k postings from its list and discard the rest. Results were averaged over 10 trials.

[1] Instead of overtly modeling the probability $P(R = 1|q,d)$ of relevance of a document d to a query q, as in the traditional probabilistic approach to IR, the basic language modeling approach instead builds a probabilistic language model M_d from each document d, and ranks documents based on the probability of the model generating the query: $P(q|M_d)$. [MRS08]

5.4 Data model

In this section, we explain how search with temporal and textual queries is processed for experiments. We present the time model and then document, query and retrieval model.

5.4.1 Time Model

A discrete time model is used to represent the time, according to [BBAW10] where a temporal expression T is represented as

$$T = (b, e)$$

where b and e are respectively the start and end date of the time window. As we want to handle fuzzy time ranges, each b and e is a time range with a lower bound and upper bound, e.g. $b = [b^l, b^u]$.

When the temporal expression is uncertain (e.g "in 2013"), this time representation allows to capture the inherent uncertainty – e.g. if we only know the start date and end date are in 2013, we use T =([1/12/2013, 31/12/2013], [1/1/2013, 31/12/2013]) which means that T can refer to any interval $[x, y]$ where $x \in [1/12/2013, 31/12/2013]$ and $y \in [1/1/2013, 31/12/2013]$.

While filtering the temporal query results, we need to find if the time range of the document, d_{time}, intersects with the time range of the query, q_{time}. The intersection operation correspond to a time window where the start (resp. end) date is the intersection of the start (resp. end) date ranges. More formally, let $l = \max(b_1^l, b_2^l)$ and $u = \min(b_1^u, b_2^u)$. If $l \leq u$, then the start date of the intersection is the range $[l, u]$, otherwise it is the empty set – and likewise for the end date.

5.4.2 Document and Query Model

A document d in the temporal collection D consists of two different parts: a textual part, denoted d_{text}, and a temporal part, denoted d_{time}. As usual in IR, d_{text} is represented as a bag of words. Following [BBAW10], d_{time} is represented as a bag of temporal expressions (e.g publication date) as defined in Section 5.4.

Temporal queries consist of keywords (in Q) and a set of temporal expressions. The temporal dimension in user input can be embedded in the textual part or not, which gives two types of queries [BBAW10]:

- *Inclusive temporal queries*: the temporal dimension of the query is included in the key-

words (*e.g.* "earthquake 17 august 1999"). In this case, the matching is done using standard IR techniques with the date tokens as keywords.

- *Exclusive temporal queries*: the temporal dimension of the query is distinct from its textual part (*e.g.* "earthquake"@ 08-17-1999) and is used to filter the results or in ranking function. In this paper, we use a strict filter – i.e. a document either match or not the temporal filter given by the query, but there is no associated score. [BBAW10] proposed a time aware ranking method as follows: $P(d|q) = P(d_{content}|q_{content}) \cdot P(d_{time}|q_{time})$ but we are not interested in obtaining a better IR model at this part of our work.

Exclusive queries are hence biased towards precision, as the system does not return documents out of filter, whereas inclusive queries are biased towards recall.

5.4.3 Retrieval Model

For exclusive queries, we use a strict filter *i.e.* a document either matches or not the temporal filter given by the query, but there is no associated score. We define the score function as follows:

$$score(d,q) = \begin{cases} 0 & \text{if } d_{time} \cap q_{time} = \emptyset \\ score_{text}(d_{text}, q_{text}) & \text{otherwise} \end{cases}$$

For inclusive queries only the textual score $score_{text}(d_{text}, q_{text})$ is used. Textual relevance could be computed according to any textual relevance model (*e.g.* TF-IDF, BM25 etc.).

5.5 Experiments

We first describe dataset and settings of experiments. These will be also used for experiments in Chapter 6. Then, we evaluate different methods presented in the previous section, and discuss the results.

5.5.1 Setup

Documents are indexed, retrieved and the indexes are pruned using Apache Lucene. The effectiveness is measured with mean average precision (MAP), normalized discounted cumulative gain (NDCG) and precision at 10 (P10), three standard metrics in ad-hoc IR [MRS08]. The

pruning ratio reported in our results is the percentage of entries in the index removed by the algorithms over the full index. This ratio is linked to the retrieval performance since high pruning ratio results in less information to be processed in the index [CL13]. We use the well established BM25 [MRS08] function for our textual score as follows:

$$score(q,d) = \sum_{i=1}^{|q|} idf(q_i) \frac{tf(q_i,d)(k_1+1)}{tf(q_i,d)+k_1(1-b+b\frac{|d|}{avgdl})}$$

where

- $tf(q_i,d)$ correlates to the term's frequency, defined as the number of times that the query term q_i appears in the document d.
- $|d|$ is the length of the document d in words (terms).
- avgdl is the average document length over all the documents of the collection.
- k1 and b are parameters, usually chosen as k1 = 2.0 and b = 0.75.
- $idf(q_i)$ is the inverse document frequency weight of the query term q_i. It is computed by:

  $$idf(q_i) = log\frac{N-df(q_i)+0.5}{df(q_i)+0.5}$$

  where N is the total number of documents in the collection, and $df(q_i)$ is the number of documents containing the query term q_i.

For TCP, we set k = 10 to maximize the precision of the top 10 documents. For IP-u, we follow the original settings of [CLTH12], and use a Language Model with Jelineck-Mercer smoothing ($\lambda = 0.6$) and use uniform document prior to estimate $p(d|t)$. For PRPP, we also use a Language Model with Jelineck-Mercer smoothing with $\lambda = 0.6$ as indicated in the original work.

5.5.2 Dataset

Web archive collections are specific due to their temporal dimension, so time must be present in the test collection (corpus, relevance judgments etc.). There exists no "standard" web archive dataset with temporal queries and the related relevance judgments.

Two publicly available document collections are used for our experiments. Both of them have time related information, but they differ in their properties, which make them complementary. The first one (LATimes) is a standard test collection for static index pruning experiments

[CCF+01, CLTH12, BB07, BB10b]. This allowed us to ensure that our results were in line with those reported in [CLTH12, CCF+01].

- TREC Los Angeles Times (LATimes) [TRE04] that contains approximately 132000 articles published in Los Angeles Times newspaper in 1989 and in 1990. LATimes contains time-stamped documents but no query and relevance judgments with temporal dimension.
- The English Wikipedia (WIKI) as of July 7,2009 that contains total of 2955294 encyclopedia articles. This collection is used in time-aware retrieval model tests [BBAW10, KN11]. It contains queries with time constraints and temporal expressions extracted from document but no time-stamped documents.

In the next sections, we briefly present the collections, queries and relevance assessments used to evaluate the methods.

5.5.3 TREC Los-Angeles Times (LATimes)

The TREC-LATimes dataset is a standard collection used for index pruning experiments. It consists of randomly selected articles published on the Los Angeles Times newspaper in 1989 and in 1990. It contains approximately 132000 articles. For the evaluation of this dataset we use TREC 6, 7 and 8 *ad hoc* topics (topics 301-450) as done in previous works [CLTH12, CCF+01].

We first present the results without any temporal dimension. Figure 5.1 plots MAP, NDCG and P10 as a function of the amount of pruning for short (title) and long(title+description) queries. 2N2P performs better at high pruning level for both short and long queries. Both PRP and IP-u, whose performance was nearly identical, follow the 2N2P. TCP is doing slightly worse than the rest except for P10. Choosing k=10 to maximize maximize the precision for the top 10 documents can be the reason. All these results show that our results are in line with those reported in [CLTH12, CCF+01].

We now return our attention to the experiments with temporal dimension. Documents' publication dates were used as d_{time}, but as there was no time associated with queries and their relevance judgments, we had to generate them from the original collection.

By using topics 301-450 and corresponding relevance judgments, we created a synthetic dataset. For each topic, we randomly picked a time interval covered by the dataset and ran these temporal queries over initial index. The temporal queries that return at least one result with the non-pruned index were kept for our experiments. By following this method, we obtained 1000 exclusive temporal short (title) and long (title+description) queries for different interval types: daily queries $|q_{time}|$ = 1 day, weekly queries $|q_{time}|$ = 7 days and monthly queries $|q_{time}|$ = 30

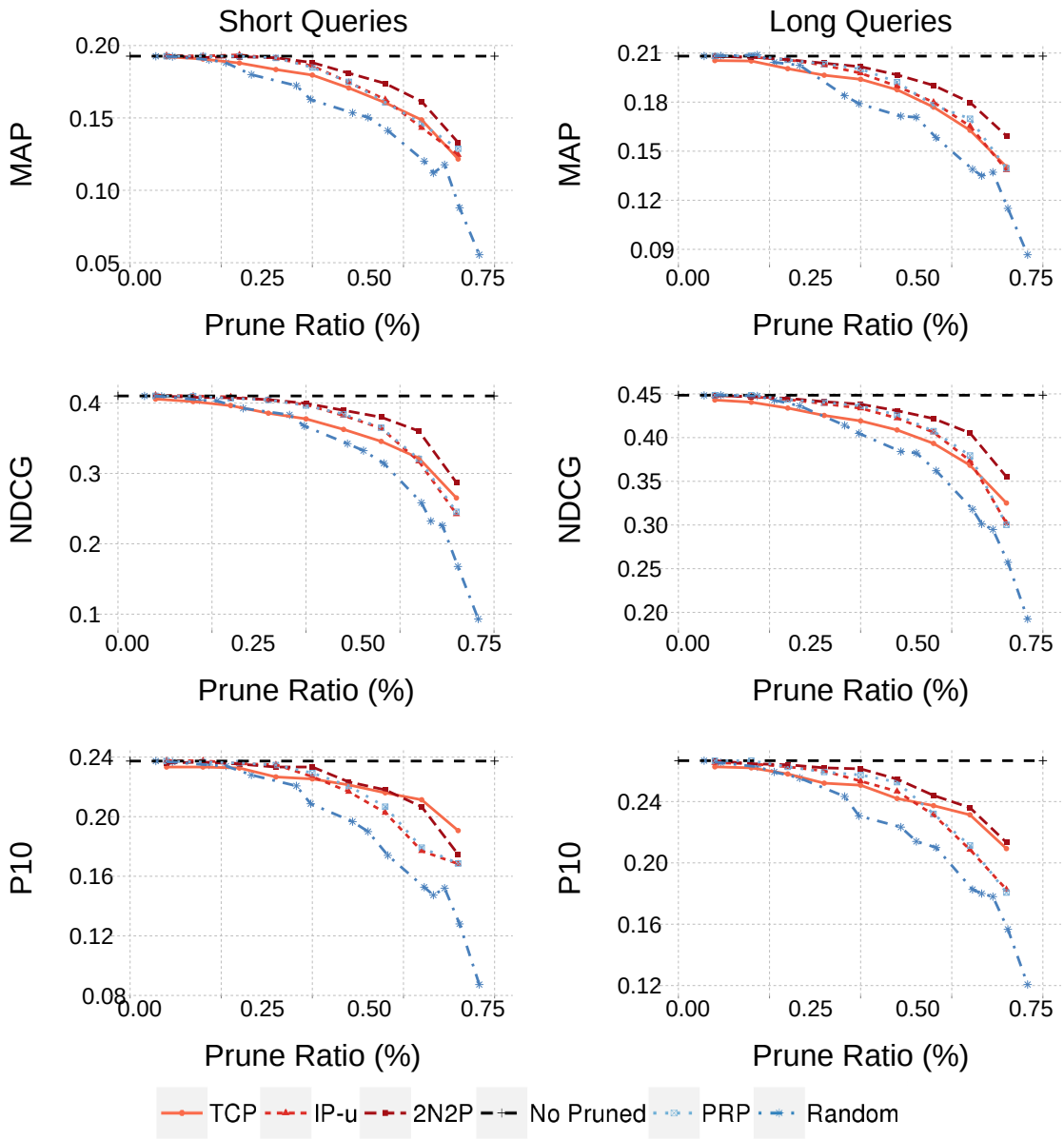


Figure 5.1: LATimes results for topics 301-450

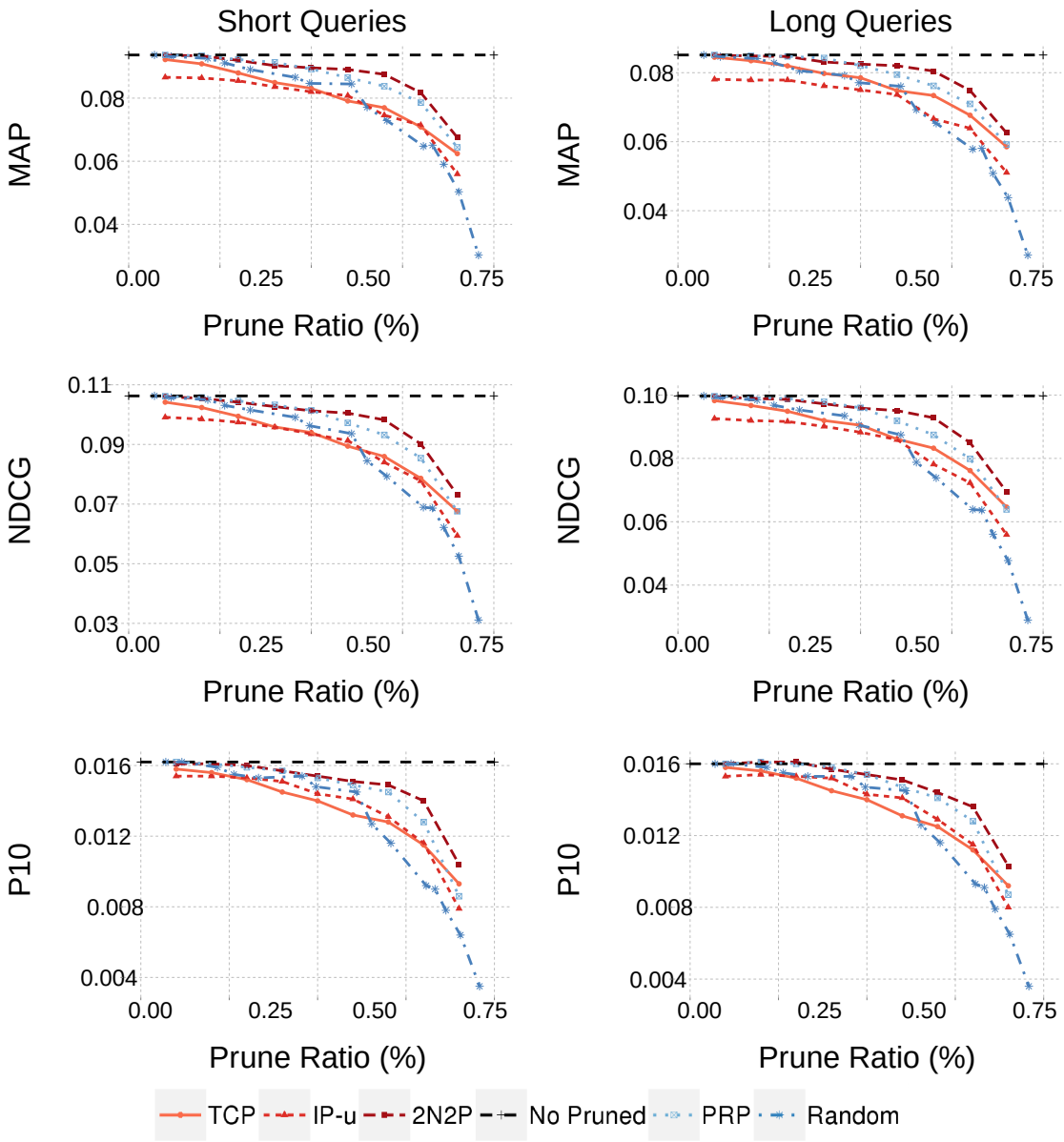


Figure 5.2: Time constraint Test results

days. Experiments were conducted with daily, weekly and monthly queries, but this didn't lead to changes in results – we thus report weekly results only.

We call this simulation *Time constraint Test* where documents out of the time interval are considered as non-relevant but we kept the original relevance for the documents in the time interval. As shown in Figure 5.2, 2N2P performs better overall with a small differences to PRP. IP-u and TCP have achieved the bottom performance with all measures. It is not that surprising to observe that Random performs also well because choosing random documents is a way of diversification and the queries time intervals are also generated randomly.

As explained before, the temporal queries used with this dataset do not reflect the real temporal information needs of the users. More precisely, the random generation of time intervals for queries do not reflect real temporal information needs. The LATimes collection was used in preliminary experiments to confirm the implementations; the Wikipedia corpus was then used to perform more realistic experiments, as described in the next section.

5.5.4 English Wikipedia Dump (WIKI)

Although there is an increasing interest in temporal collections and in research related to temporal information, to the best of our knowledge, there is only one available collection with temporal queries and corresponding relevance judgments which takes the temporal expressions in the documents into account. In [BBAW10], Berberich et al. used *The English Wikipedia* dump of July 7, 2009 and obtained temporal queries for this collection by using Amazon Mechanical Turk (AMT). They selected 40 of those queries and categorized them according to their topic and temporal granularity (daily, weekly, etc.). The relevance judgments for these queries were also collected by using AMT and made publicly available [2].

Time constraints in queries refer explicitly to the time intervals that the relevant Wikipedia articles refer to. In our experiments, each temporal expressions extracted from documents is considered as the validity interval of the documents and used to filter the results of temporal queries. We used 2330277 encyclopedia articles that contain temporal expressions, and, following [BBAW10], considered inclusive and exclusive queries as explained in Section 5.4.2.

Figure 5.3 plots MAP, NDCG and P10 as a function of the amount of pruning for inclusive and exclusive temporal queries. Random performs as theoretically expected unlike LATimes. Because this collection is not based on a simulation. We observe that at low pruning levels, the performance can be higher for pruned indexes than with not pruned ones. This shows that pruning removes postings that would have made non relevant documents retrievable. We now turn to comparing the different pruning methods. All methods perform better than Random. This shows that even if they are not designed with time in mind, they are able to handle temporal

[2] http://www.mpi-inf.mpg.de/ kberberi/ecir2010/

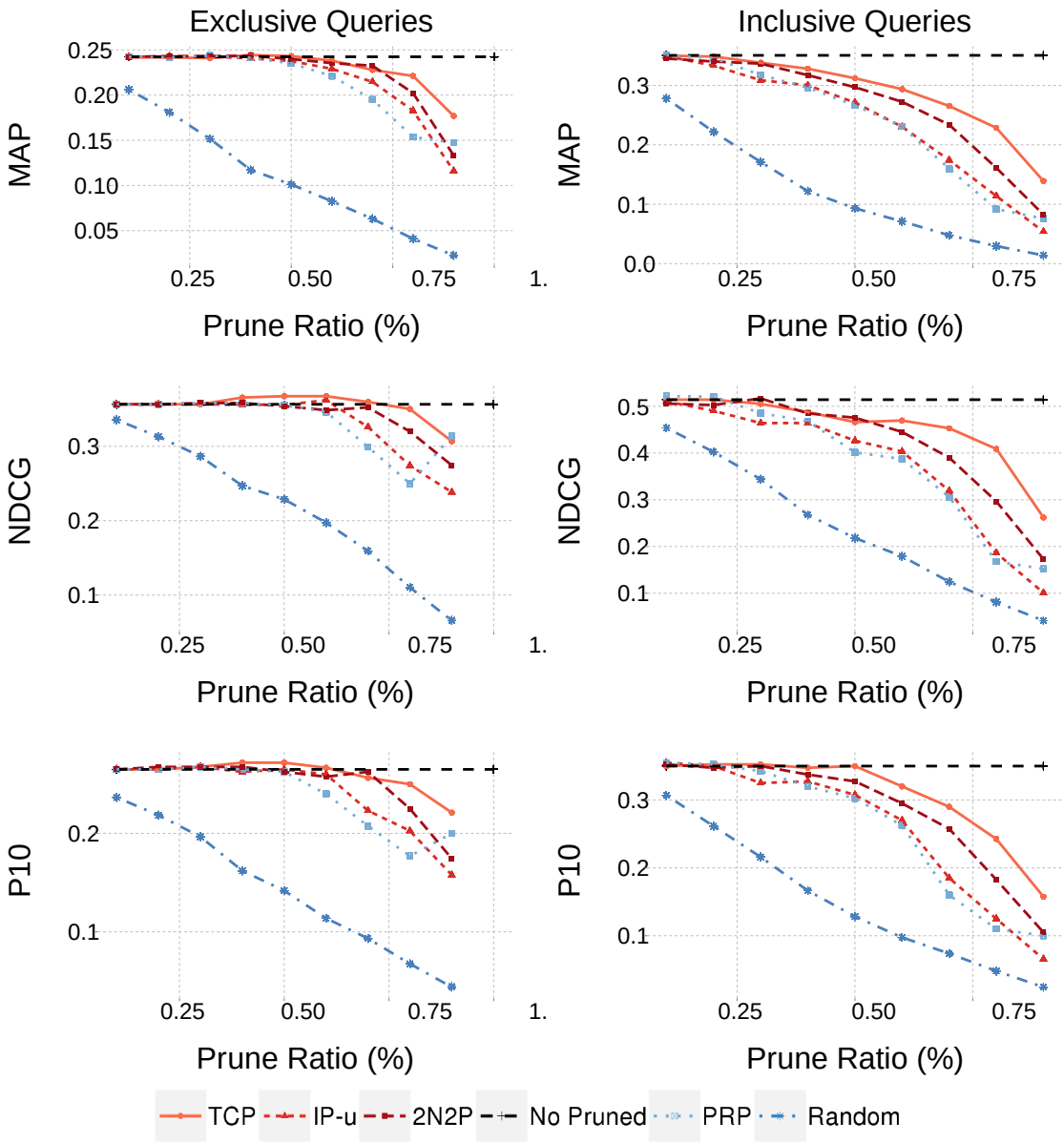


Figure 5.3: Results - WIKI results for Exclusive and Inclusive queries

queries. Our result shows that, at small prune levels (< 0.5), all the methods differ little in performance, and the difference at larger prune levels seems more evident.

TCP outperforms all other methods for inclusive and exclusive queries. This gap is more significant at high levels of pruning and with inclusive queries. One hypothesis is that methods using uniform threshold can end up by pruning every term occurrences, thus they may lead to bigger gaps. However, both PRP and IP-u performances were nearly identical for inclusive queries, the gap increases for exclusive queries and IP-u performs better than PRP.

Another observation is that for not pruned temporal collections, inclusive queries perform better. However, exclusive queries resist better to an increasing prune ratio. It can be related to time filters that remove postings that would have made non relevant documents retrievable.

5.5.5 Correlation Analysis

Retrieval performances can be explained by different factors: prune ratio, query type (daily, monthly etc.), prune method and temporal diversity. The latter corresponds to our hypothesis. We thus further analyzed the relation between these factors and the performance. In this section, we first define different measures for temporal diversification and then we analyze the relation between these measures and the performance. We consider measures of diversification that compare the time distribution of documents in the pruned index with the one in the non pruned index or with itself.

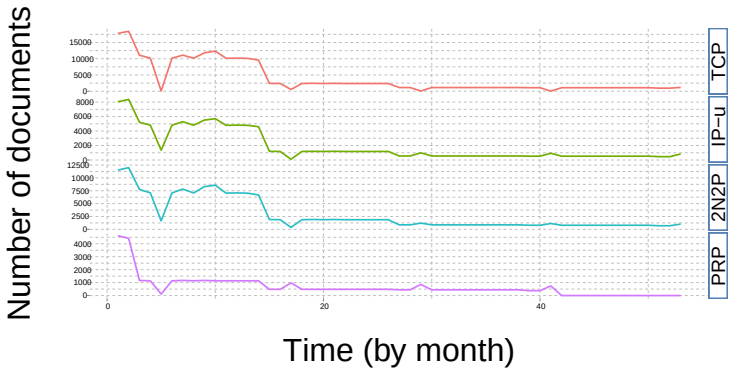


Figure 5.4: Time series for one query at 50% prune ratio

The time distribution of each query is given by the time series associated to the temporal dimension of the documents [Efr10, DGI08, JD07]. For each query q, we can associate a time series, S_t by identifying all the documents in the temporal collection that contain the terms of t. Then we take the sum of the number of documents based on a chosen granularity (*e.g.* hour, day, week etc.) according to the documents temporal part d_{time}. In our work, we used the monthly

granularity. We generate time series for each query, for each prune ratio and for each method where data points represent the number of documents containing the query. Figure 5.4 shows a small proportion of the time series at 50% of prune ratio for different methods. For the measures where we compare two time series (Cross-correlation, SAX-DTW, DTW), each time series is compared to its corresponding initial time series generated from non pruned index.

We use four different methods to compute temporal diversity: *Cross-Correlation*, *Auto-Correlation*, *DTW* and *SAX-DTW*. We first explain in detail these measures.

Cross-correlation

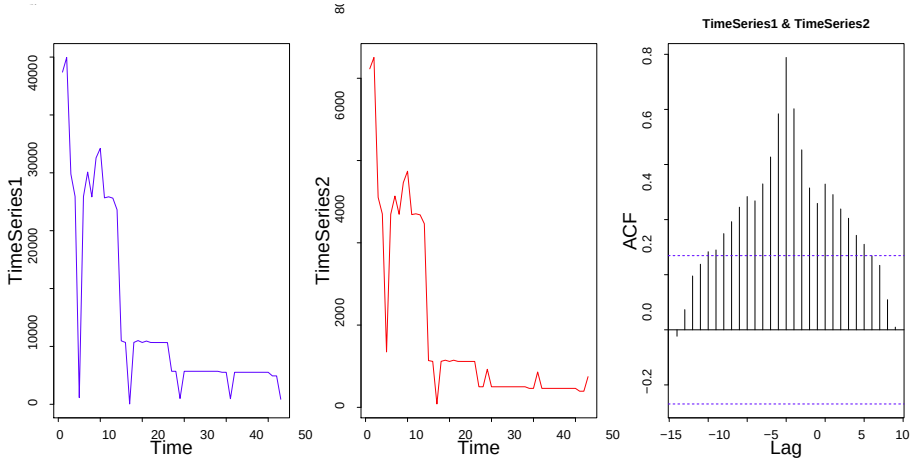


Figure 5.5: Example of cross-correlation between two time series

Cross correlation is a standard method of estimating the degree to which two series are correlated. It is a procedure for measuring the similarity of one time series to another as a function of a time-lag applied to one of them. We first define the *Cross-covariance function*. Let x and y to be two time series. The cross-covariance function, $ccvf_{x,y}$ is defined as covariance of x and y of length N time lag τ as follows:

$$\sigma_{xy}(\tau) = \frac{1}{N-1}\sum_{k=1}^{N}(x(k)-\eta_x)(y(k+i)-\eta_y)$$

where η_x and η_y are mean of time series x and y respectively. Cross-correlation is described as the normalized cross covariance function as follows:

$$r_{xy}(\tau) = \frac{\sigma_{xy}(\tau)}{\sigma_{xx}(0)\sigma_{yy}(0)}$$

where $\sigma_{xx}(0)$ and $\sigma_{yy}(0)$ are the auto-covariance at lag 0 for the time series x and y respectively. We use the mean of the all lags as final measure.

Auto-correlation

Auto-correlation is the cross-correlation of a time series with itself. Autocorrelation refers to the correlations of time series with its own past and future values [Dig11]. Our aim here is to check whether the pruning method is able to keep the initial index *diversification* to fulfill different temporal information needs.

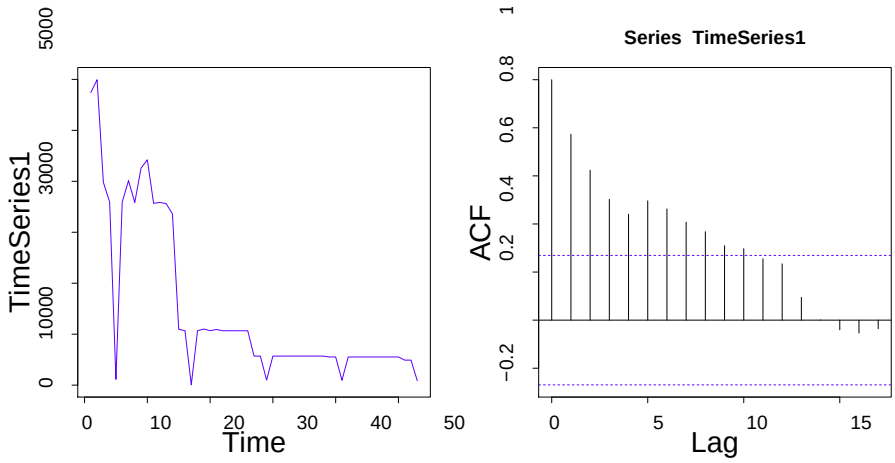
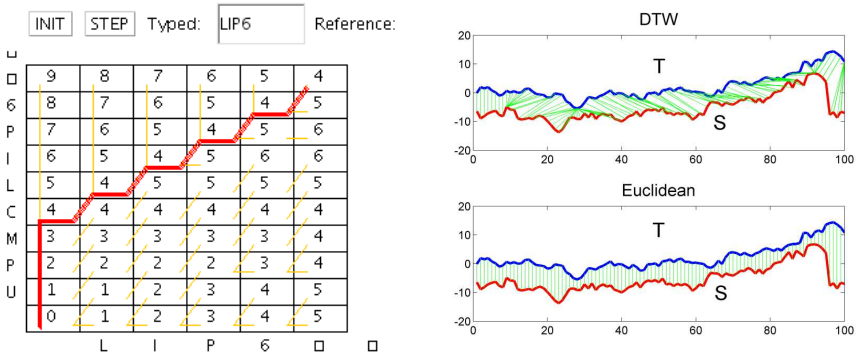


Figure 5.6: Example of auto-correlation of time series

Dynamic Time Warping (DTW)

Dynamic Time Warping, DTW, is a well-known algorithm which aims at comparing and aligning two sequences of data points (e.g. time series). This method replaces one-to-one point comparison, used in Euclidean distance, with many-to-one comparison, thus it can be used to compare the time series of different lengths. Figure 5.7b shows the differences between DTW distance and Euclidean distance [PMP12].

DTW works by warping (hence the name) the time axis iteratively until an optimal match between the two sequences is found. We can construct a $n * m$ distance matrix. In this matrix, each cell (i,j) represents the distance between the i-th element of sequence A and the j-th element of sequence B. The distance metric used depends on the application but a common metric is the euclidean distance. Finding the best alignment between two sequences can be seen as finding the shortest path to go from the bottom-left cell to the top-right cell of that matrix. The length of a path is simply the sum of all the cells that were visited along that path. The further away



(a) DTW example for two strings (b) DTW versus Euclidean for time series

the optimal path wanders from the diagonal, the more the two sequences need to be warped to match together. We use [Gio09] for our implementation.

Symbolic Aggregate Approximation(SAX-DTW)

In our context, it can be more appropriate to measure the structural distance. Symbolic Aggregate Approximation (SAX) is a novel symbolic representation for time series recently introduced in [LKLC03], which has been shown to preserve meaningful information from the original data and produce competitive results for classifying and clustering time series.

The basic idea of SAX is to transform a time-series of length n into the string of arbitrary length ω, where $\omega <= n$ typically, using an alphabet of size greater than 2. To convert a time series into symbols, it is first normalized, and two steps of discretization are performed. First, a time series T of length n is transformed into a Piecewise Aggregate Approximation (PAA) representation[3]. Next, to convert the PAA coefficients to symbols, they determine the breakpoints that divide the distribution space into α equiprobable regions, where α is the alphabet size specified by the user (or it could be determined from the Minimum Description Length). In other words, the breakpoints are determined such that the probability of a segment falling into any of the regions is approximately the same. Once the breakpoints are determined, each region is assigned a symbol. The PAA coefficients can then be easily mapped to the symbols corresponding to the regions in which they reside [RLG+10].

Figure 5.7 shows an example of SAX for two time series. First one is the the time series for one query generated from non pruned index. The second one (second row) is the time series for the same query after pruning. SAX words generated from time series can be compared according to existing distance metrics. In our experiments, we used DTW as explained in detail below and as

[3]PAA divides the time series of length n into w equal-sized segments; the values in each segment are then approximated and replaced by a single coefficient, which is their average. is their average. Aggregating these w coefficients forms the Piecewise Aggregate Approximation (PAA) representation of the time series.

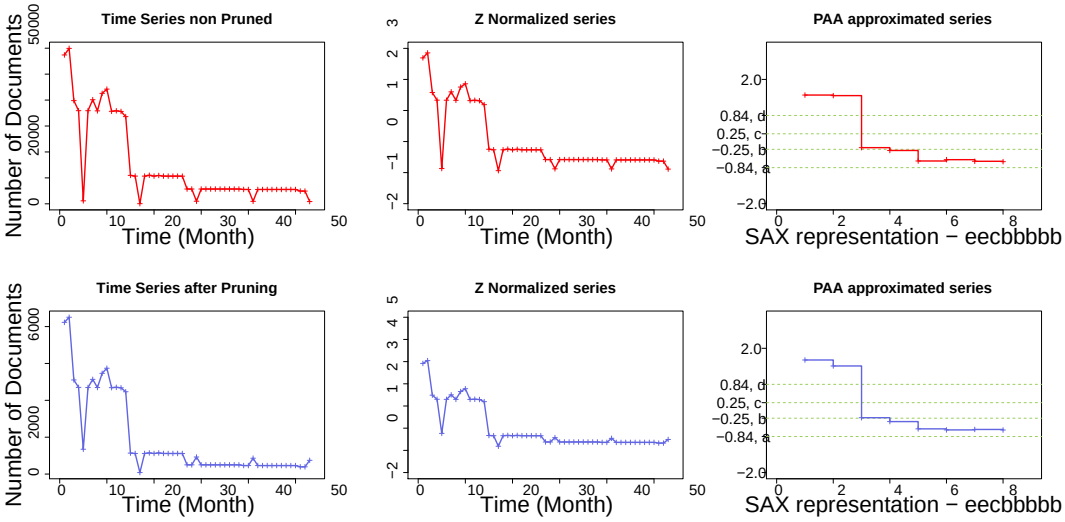


Figure 5.7: Example of conversion of two time series of length 53 into a word of length w=8

seen in Figure 5.7a[4].

We analyzed the relation between the difference of MAP to the not pruned MAP (MAP_d) and those four factors by using 4-way no interaction analysis of variance (ANOVA) as follows:

$$Y_{1,2,3,4} = a_1 + b_2 + c_3 + d_4 + \varepsilon_{1,2,3,4}$$

where $Y_{1,2,3,4}$ is the measured performance (MAP_d), a_1 is the prune ratio effect, b_2 query type effect, c_3 prune method effect, d_4 temporal diversity effect and $\varepsilon_{1,2,3,4}$ is the error.

Effect	Df	Cross-Correlation		Auto-Correlation		SAX-DTW		DTW	
		F value	μ_p^2	F value	μ_p^2	F value	μ_p^2	F value	μ_p^2
Temporal Diversity	F(1,1094)	365.70***	0.05	291.79***	0.05	563.04***	0.11	232.45***	0.02
Prune Ratio	F(1,1094)	111.90***	0.08	188.70***	0.12	75.63***	0.04	235.86***	0.13
Query Type	F(4,1094)	4.07**	0.01	4.5404**	0.01	8.32***	0.01	7.97***	0.01
Prune Method	F(4,1094)	9.852***	0.02	10.9142***	0.02	5.79***	0.01	8.88***	0.01

Table 5.1: 4-way no interaction ANOVA results with Significance codes: '***' 0.001 / '**' 0.01

For the experiments in this section, we cannot take the overall prune ratio as it can vary between topics. We thus take the average prune ratio for terms of each query and group them into intervals. We measured the four different approaches between the non pruned time series for each query and the time series generated for each query, for each method and for each prune

[4]http://www.cnel.ufl.edu/ kkale/dtw.html

ratio. We first did the ANOVA test to see the significance of the relation between factors. The results are covered in Table 5.1. Each row indicated a measure (MAP_d)-effect combination and each main column represents different temporal diversification measures. Statistics, such as F value, degree of freedom (Df), are given for each case. Eta-squared μ_p^2 is used to measure the effect size. For all three methods, all the main effects except "Query Type" are significant for $p < 0.001$. Analysis shows that query type and prune method have relatively small effect sizes, however, prune ration and temporal diversity measures have greater influence on the retrieval performance.

To better interpret these relations, the correlation between two vectors is measured using Pearson's correlation coefficient (r) for each prune ratio pr. One vector contains the MAP_d values at pr and the other vector contains the measure (e.g. cross-correlation, DTW etc.) values at pr for each query. We underline that we are working with a small dataset (40 queries with 1220 documents with relevance judgments). All the results can be found in Table 5.2.

For cross-correlation, when the cross-correlation value increases, it means that two time series are similar and MAP_d value should also increase. So, this correlation should be positive overall which is also the result of our experiments except a few negative correlations. When autocorrelation increases, it means that the time series is stationary and it does not contain any significant temporal diversity, thus MAP_d should decrease. Correlation should be negative. However, the results we obtained for autocorrelation are overall positive. This shows that the autocorrelation is not an appropriate measure to define the relation between temporal diversification and the system performance. For SAX-DTW and DTW when the distance between time series increases, MAP_d should decrease which leads to the negative correlation. According to Table 5.2, we observe that *SAX-DTW* is the best fitting measure.

5.6 Conclusion

In this chapter, we compared the retrieval performance of four static index pruning methods with temporal queries. None of the tested methods were designed with time in mind but we have shown that they work well on our test collection. However, there is room for improvement, in particular, by taking advantage from the link between diversification and performance that we showed in this chapter.

There are several temporal dimensions associated with search results for different queries and the users can be interested in any of these dimensions. In the next chapter, we investigate new pruning methods that keep both most relevant and most temporal diverse postings in the index.

	Method	Prune Ratio	Cross-correlation	Auto-correlation	SAX.DTW	DTW
1	TCP	0.1	-0.046	0.088	-0.178	0.049
2	TCP	0.2	-0.103	0.029	-0.195	0.409
3	TCP	0.3	0.662	0.635	-0.553	0.462
4	TCP	0.4	0.678	0.622	-0.209	0.383
5	TCP	0.5	0.019	0.036	0.078	-0.299
6	TCP	0.6	0.613	0.730	-0.500	0.224
7	TCP	0.7	0.411	0.452	-0.287	-0.111
8	TCP	0.8	0.117	0.179	-0.064	-0.101
9	TCP	0.9	0.366	0.444	-0.421	0.134
10	TCP	1	0.129	0.138	-0.254	-0.048
11	IP-u	0.1	0.019	0.014	-0.028	0.661
12	IP-u	0.2	-0.066	0.006	-0.271	0.079
13	IP-u	0.3	-0.002	-0.003	-0.277	0.670
14	IP-u	0.4	0.054	0.030	0.033	-0.089
15	IP-u	0.5	-0.053	-0.057	-0.112	-0.026
16	IP-u	0.6	0.344	0.109	-0.514	-0.125
17	IP-u	0.7	0.209	0.161	-0.283	0.066
18	IP-u	0.8	0.417	0.390	-0.540	0.507
19	IP-u	0.9	0.153	0.316	-0.282	-0.173
20	IP-u	1	0.333	0.305	-0.373	-0.286
21	2N2P	0.1	0.024	0.022	-0.033	0.228
22	2N2P	0.2	-0.007	-0.029	-0.121	0.224
23	2N2P	0.3	0.025	0.065	-0.237	0.153
24	2N2P	0.4	-0.018	-0.015	-0.014	-0.048
25	2N2P	0.5	0.137	0.176	-0.157	-0.007
26	2N2P	0.6	0.438	0.377	-0.304	0.019
27	2N2P	0.7	-0.077	-0.080	0.081	-0.217
28	2N2P	0.8	-0.066	0.064	-0.297	0.087
29	2N2P	0.9	0.056	0.201	-0.148	-0.181
30	2N2P	1	0.107	0.219	-0.149	-0.456
31	PRP	0.1	0.025	0.023	-0.029	0.244
32	PRP	0.2	0.144	0.003	0.273	-0.411
33	PRP	0.3	0.054	0.077	-0.213	0.049
34	PRP	0.4	0.338	0.238	0.062	-0.250
35	PRP	0.5	0.447	0.450	-0.260	-0.058
36	PRP	0.6	0.377	0.177	-0.716	0.056
37	PRP	0.7	0.054	-0.093	-0.448	-0.243
38	PRP	0.8	0.432	0.161	-0.104	-0.310
39	PRP	0.9	0.364	0.463	-0.669	-0.385
40	PRP	1	0.208	0.290	-0.306	-0.237

Table 5.2: Correlation results values all

6 Diversification Based Static Index Pruning Application to Temporal Collections

Temporal collection are intrinsically big, leading to index files that do not fit into the memory and an increase in query response time. Decreasing the index size is a direct way to decrease this query response time. As discussed in Chapter 5, in the context of web archives, it seems necessary to preserve the *temporal diversity* of the archive after pruning.

In this chapter, we propose a *diversification-based static index pruning* method. It differs from the existing pruning approaches by integrating diversification within the pruning context. We aim at pruning the index while preserving retrieval effectiveness and diversity by maximizing a given IR evaluation metric like DCG. We show how to apply this approach in the context of web archives. Finally, we show on two collections that search effectiveness in temporal collections after pruning can be improved using our approach rather than diversity oblivious approaches.

6.1 Motivation

The major challenges with temporal collections are the efficient storage and retrieval of information. Querying these collections requires the use of temporal queries that combine standard keyword queries with temporal constraints. The results of such queries can be obtained by using

time-aware ranking models that rank the documents based on temporal and textual similarities [BBAW10, KN10] or by filtering out the topically relevant documents with time intervals [Nut03].

As discussed and showed in the previous chapter, pruned indexes should preserve the temporal diversity of postings. This issue is related to the problem of *search result diversification* that aims to improve user satisfaction according to different information needs. Search results should be optimized to contain both relevant and diverse results. A number of search result diversification methods are proposed [WCO11, AGHI09, CG98]. We base our work on coverage based diversification, which is an optimization problem of an objective function related to both relevance and diversity of the results. In our case, the objective function is an IR metric adapted to diversification.

As diversification is computationally intensive, we chose to follow a static pruning approach where postings are pruned off-line. To the best of our knowledge, static index pruning and search result diversification were never studied together. In this chapter, we propose a new approach that we name *Diversification-based static index pruning*.

Assume that for a query q, a set of results are obtained in an initial index without any pruning. There are several aspects associated with each result and the users can be interested in any of these aspects. The question is then: in which order the results should be re-ranked before pruning to be still able to satisfy the users different intents when the same query is executed on the pruned index? We show how to apply our approach on temporal collections by using temporal diversification. Although in our context we focus on temporal coverage, the method can also be applied to non-temporal collections with different diversification categories. To achive this, we need to define related aspects for each term to deal with explicit diversification for index pruning. We do not have this problem with temporal dimension as we define the temporal aspects, we are able to associate them to each term. It is not an impossible task, for example, one can study the related works to term-level diversification or to aspect detection. As it is out of this thesis scope, we do not realize experiments for non-temporal collections.

Contributions

Our main contribution in this chapter is as follows:

- We take into account diversification in the context of index pruning and propose a variation of the standard greedy algorithm.
- We show how to apply this method to temporal diversification index pruning by defining the temporal aspects associated to a query.
- We perform experiments on two datasets that were modified for the task of retrieval in web

archives and show that our approach is stable and works better than other index pruning strategies.

Organization

The rest of this chapter is organized as follows: Section 6.2 puts our work into context with related research. In Section 6.3, we introduce our model for Diversification based static index pruning. Section 6.4 shows how to apply our approach to temporal collections. Section 6.5 describes the series of experiments and analysis. We conclude and discuss future work in Section 6.6.

6.2 Related Works

In ad-hoc information retrieval (IR), the goal is to retrieve most relevant documents according to a topic or *query*. In order to determine a ranking of documents, a score of relevance is computed for each document. Indexes are used to compute these scores efficiently at query time. As we aim to keep both most relevant and most diverse postings in the index, our work is related to two IR domains, namely Static Index Pruning and Search Results Diversification. Related works for static index pruning were presented in Chapter 4. In this chapter, we discuss related works on search result diversification.

Result diversification aims to return a list of documents which are not only relevant to a query but also cover many subtopics of the query. The approaches can be classified as *explicit* and *implicit* [Ahl12].

Implicit methods consist in evaluating a similarity measure between documents and to use this information to select diverse documents by discarding too similar documents. One of the earliest approaches [CG98], *Maximal Marginal Relevance* (MMR), select documents by computing a score which is a linear combination of a novelty score (dissimilarity) and the original relevance score. The various approaches based on MMR differ mostly by how the similarity between documents is computed [ZCL03, WZ09].

Explicit methods suppose that the different aspects of a query can be computed along with the extent to which a document is relevant to each of these aspects. The most representative work based on explicit methods is that of Agrawal et al. [AGHI09] where the authors propose to maximize the probability of an average user finding at least one relevant result. They assume that an explicit list of subtopics is available for each query. Recently, Welch et al. [WCO11] observed that this approach falls into random document selection after choosing one document for each subtopic.

Coverage based diversification corresponds well to our problem of finding temporally diverse results since we can define temporal aspects explicitly as shown in Section 6.4.1. We want to keep in the index the results that cover many different aspects (e.g. temporal aspects) for the given query. A way to avoid this random document selection is to use a criterion based on the maximization of an IR metric. This type of criterion is used to learn to rank in ad-hoc IR [BJKN10, SKB+12] and has recently been applied to diversification [BJKN10]. In this latter work, the authors use a performance measure based on discounted cumulative gain, which defines the usefulness (gain) of a document depending on its position. Based on this measure, they suggest a general approach to develop approximation algorithms for ranking search results that captures different aspects of users' intents. However, no experiments were conducted.

Our work is related to [BJKN10] in that we optimize a IR metric. However, we use it for index pruning and define algorithms to perform this optimization efficiently along with a definition of temporal aspects related to a given user query.

Our aim in this chapter is not to propose a new diversification approach but to use diversification for static index pruning. Thus, we simply apply an approach similar to [BJKN10] by using explicit methods instead of MMR in the original work.

6.3 Diversification Based Static Index Pruning

Existing static index pruning methods can discard relevant documents for some aspects of a query (e.g. subtopics) as they do not take the result diversification into account. For example, with non diversification-aware pruning strategies, the reviews about a product could be restricted to those associated with the highest ratings, or in the case of temporal information, to those associated with a given range in time. In this section, we propose a method for static index pruning by taking into account the diversification of results.

6.3.1 Problem Formulation

In order to define a criterion for diversification-based static index pruning, we need to know which queries might be asked by users, which aspects are associated to the queries and how documents are linked to these aspects (i.e. how relevant is the document with respect to this query aspect). We define Q as a set of queries that a user can ask, W_q as the set of aspects of a query $q \in Q$. We show in Section 6.3.4, how we approximate the distribution over queries Q and in Section 6.4 the distribution over aspects W_q. All the notations that we use throughout this chapter are summarized in Table 6.1.

Notation	Description
q	query
Q	a set of queries
t	term
D	a set of postings
M	an evaluation metric
w	aspect
W_q	a set of aspects of query q
S	a sub-set of postings

Table 6.1: Notation used in this chapter

In order to define a criterion, we chose to follow the approach taken by some works on learning to rank (learning the parameters of a search engine) that try to optimize directly an IR metric like in [CZ06, XLL+08]. This approach is interesting, since it directly tries to optimize a measure which is related to the user satisfaction, and on the other hand, it allows to easily integrate our diversification requirements.

We can now define the criterion that we want to optimize. Given a set of postings D, an evaluation metric M, a number of postings to preserve k, we would like to find a set of postings S^* that maximizes M:

$$S^* = \arg\max_{S \subseteq D, |S|=k} \mathbb{E}(M|S) \tag{6.1}$$

If we suppose we can estimate the probability that each individual query $q \in Q$ is asked, we can compute this expectation as a sum over all the possible queries. This gives:

$$\mathbb{E}(M|S) = \sum_{q \in Q} P(q) \sum_{w \in W_q} P(w|q) \mathbb{E}(M|q, w, S) \tag{6.2}$$

where P(q) is the probability of a user issuing query q and $P(w|q)$ is the distribution on the aspects of query q with $\sum_{w \in W_q} P(w|q) = 1$. In practice, we assume that the queries are reduced to one term (Section 6.3.4) and define what (temporal) aspects a query q is associated with by different means (Section 6.4.1).

6.3.2 Optimization

We now turn to the problem of optimization of Equation 6.1, which is known to be NP-hard [AGHI09, Car11]. However, there are known algorithms that give in practice good approximations. In our work, we use the standard *greedy algorithm heuristic* that we describe in this section.

This heuristic is based on the submodular nature of the function given in Equation 6.1.

Definition 1. Consider a set X and a function f defined on 2^X. f is said to be *submodular* if for any two subsets A and B in 2^X, the following holds:

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$$

Intuitively, a submodular function satisfies the economic principle of diminishing marginal returns, i.e., the marginal benefit of adding a document to a larger collection is less than that of adding it to a smaller collection [AGHI09]. Equation 6.2 has been shown to be submodular for metrics like DCG [BJKN10] that we use in our work.

The greedy algorithm is the most commonly used since it offers guarantees on the objective function value and works well for a wide range of problems. It chooses the best solution that maximizes (or minimizes) the objective function (also known as marginal function or cost function) at each step, independently from the future choices. Finally, all the best local solutions are combined to get the global optimal/suboptimal solution. As proved in [NWF78], the greedy algorithm ensures that the value of the objective function for the approximation is at least (1-1/e) times the optimal one.

We adapt the greedy algorithm to Equation 6.1, as shown in Algorithm 5. The greedy algorithm uses two sets: X for available items and S for selected items. S can be initialized with some items or can be empty. Then, the algorithm iterates over X and adds the best element (according to the objective) to S until $|S| = k$. The posting p_i that has the highest value for objective function is chosen as follows:

$$p_i = \arg\max_p \mathbb{E}(M|S_{i-1} \cup \{p\})$$

Intuitively, the documents that can provide coverage for many query aspects should be placed in the beginning of the ordering so as to maximize the objective function.

Algorithm 5 *Diversify* - Greedy Algorithm

Input: k (target), S the set of all postings, $f(S') = \mathbb{E}(M|S')$
Output: The set of optimal postings S^*

1: $S^* = \emptyset$
2: **while** $|S^*| < k$ **do**
3: $\quad p^* \leftarrow \arg\max_{p \notin S} f(S \cup \{p\})$
4: $\quad S^* \leftarrow S^* \cup \{p^*\}$
5: **end while**
6: **return** S^*

6.3.3 Metric family

We use the DCG measure (Discounted Cumulative Gain) [MRS08] as our metric M since it is a well-established IR metric that allows an efficient pruning algorithm (Section 6.3.5) to be defined. DCG can be written as the sum of gain and discount functions as follows:

$$DCG = \sum_{j \geq 1} c(j) g(d_j) \tag{6.3}$$

where $c(j)$ is a decreasing discount factor[1], and $g(d_j)$ is the gain function for a document d_j, representing the value of the document for the user. This value is usually defined as a function of the relevance of the document, where the relevance can take several value ranging from 0 (not relevant) to 5 (excellent result). In the following, we suppose that M belongs to the DCG family but any other metric that allows an efficient pruning algorithm can be used.

By choosing a decaying function as a discount function, DCG focuses on the quality of the top-k results. Intuitively, in the context of pruning, choosing a metric M that has a bias towards top ranked results is interesting since this will help to establish a balance between relevance (adding one more result dealing with the same query aspect) and diversity (adding one more result dealing with another query aspect).

The definition of DCG as a sum over ranks allows to get a closed form formula where the relevance of each document is explicit. Starting from Equations 6.3, we get:

$$\mathbb{E}[M|q,w,S] = \sum_{j \geq 1} c(j) \mathbb{E}[g(d_j)|q,w,S] \tag{6.4}$$

Since we don't know the relevance of the documents to the query q, we define the expectation $\mathbb{E}[g(d_j)|q,w,S]$ of the gain as the probability of relevance $P(d_j|w,q)$ of document d_j for query q. The probability is directly given by an IR model and it is equal to zero if w is not an aspect of d_j. Note that we could relax this latter requirement by allowing documents to be more or less relevant to the aspects, but we did not consider this in this work. From Equations 6.2 and 6.4, we get:

$$\mathbb{E}[M|S] = \sum_{q} P(q) \sum_{w \in W_q} P(w|q) \sum_{j \geq 1} c(j) P(d_j|q,w,S) \tag{6.5}$$

6.3.4 Query Generation Model

We now turn to the estimation of the distribution over the queries $q \in Q$. As we work on static index pruning which is applied off-line, we do not have any knowledge over Q. As an approximation like in [BB10b], we assume that each query consists of one single term. More sophisticated

[1] Usually set to $1/\log(1+j)$ as in this paper

strategies would involve using query logs and/or using co-occurrence information, but we leave this for future work.

Starting from Equation 6.5, we get:

$$\mathbb{E}[M|S] = \sum_t P(t) \sum_{w \in W_t} P(w|t) \sum_{j \geq 1} c(j) P(d_j|t,w,S) \tag{6.6}$$

As discussed in Section 6.3.2, it is shown that the greedy algorithm still achieves (1-1/e) approximation even in the more general settings of using a submodular function with a logarithmic discount function [BJKN10].

We hence use a greedy algorithm to find the best posting to add to the current postings list. The above equation allows us to define an efficient algorithm to find this posting, as we explain below. First, the probability of a document d to be relevant given a term t, an aspect w and a set of selected postings S is:

$$P(d|t,w,S) = \begin{cases} P(d|t) & \text{if } p_{t,d} \in S \text{ and } w \text{ matches } d \\ 0 & \text{otherwise} \end{cases}$$

We want to choose the posting that will maximize the criterion of Equation 6.5. Given a document d that contains the term t, we first define $\Delta(S, p_{t,d})$ as the change in the value of the criterion that we would get by adding to S the posting $p_{t,d} \notin S$ for a document d and term t.

Let $l = (d_{t,w,j})_j$ be the list of $N_{t,w}$ documents whose postings $\{p_{t,d_{t,w,j}}\}_j$ belong to the already selected postings S, ordered by decreasing probability of relevance[2] $P(d_{t,w,j}|t,w)$, and let $r_{t,w}$ be the first rank where $p(d|t,w) > p(d_{t,w,r}|t,w)$. In this case, adding $p_{t,d}$ to the posting list will change:

- the rank of all the documents after rank $r_{t,w}$, computed in (1) - Equation 6.7
- the gain of the document d will be added to the DCG value, computed in (2) - Equation 6.7

All the above allows us to define the increase in DCG as:

$$\begin{aligned} \Delta(S, p_{t,d}) = P(t) \sum_{w \in W_t} P(w|t) \times \Bigg[& \underbrace{c(r_{t,w}) P(d|t,w)}_{2} \\ & + \sum_{j=r_{t,w}}^{N_{t,w}} \underbrace{(c(j+1) - c(j)) P(d_{t,w,j}|t,w)}_{1} \Bigg] \end{aligned} \tag{6.7}$$

[2]We discard those with a probability 0 as they will not change the value of the criterion

A naive approach to selecting the best posting by computing explicitly the list of documents l would be too computationally expensive $O\left(\sum_{t \in V} |P_t|^3 |W_t|\right)$. We propose the Algorithm 6 that computes the next best posting to select; the overall complexity is given by $O\left(\sum_{t \in V} |P_t|^2 |W_t|\right)$, *i.e.* the complexity is quadratic with respect to the size of the posting list for a term t and linear with respect to the number of considered aspects W_t. It reduces the complexity from cubic time to quadratic time.

The Algorithm 6 shows how to get the next best posting for a given term. For each round of the greedy algorithm, we just need to compute the next best posting for the term corresponding to the selected postings. The proposed algorithm is bottom-up, that is, for each term we start from the less relevant documents first, allowing us to optimize the computation of the sum (1 + 2) in Equation 6.7.

We take as inputs a list of tuples, P_t, a set of selected postings S, a set of aspects W and a mapping m from the documents to the powerset of aspects W, *i.e.* for each document, it keeps the list of aspects such that $P(d|t,w) \neq 0$. P_t consist of tuples $(d_i, p(t|d_i))$ that contain document (d_i), its score based on the probability of relevance $p(t|d_i)$. S is used to track which postings are already selected. The set of aspects W is a set of tuples (t_w, p_w) where t_w is the number of selected postings and $p_w = p(w|t)$ gives the distribution over the aspects for the given term. Aspects have also associated properties (like the times windows in web archives) but this is not relevant for the algorithm. In the following, we access parts of the tuples by using them as functions, e.g $d_i(P_t)$ refers to the d_i entry of the i^{th} element of the list.

In the algorithm, each aspect w in W is associated to a state given by r_w and Δ_w, where r_w holds rank position of the current $p_{d_i,t}$ in the postings associated to aspect w. Δ_w holds the value of the last sum in Equation 6.7 up to rank r_w.

The core of the algorithm is the computation of the Equation 6.7 (lines 10-16) which combines the relevance score of the document d_i with respect to $p(t|d_i)$ and a diversity score. Starting from the bottom of the list P_t, for each document, we traverse each aspect associated with this document. If the posting was not selected in the previous execution of the algorithm, its score is calculated according to Equation 6.7 (line 15) by adding the current Δ_w to the gain for the document. Otherwise, Δ_w is updated (line 12) and the current rank r_w is decremented. When we have checked all the list, we can return the best posting along with the change in the criterion (up to the scaling by $p(t)$).

6.3.5 Pruning

There are different ways to estimate the probability $P(t)$ that should reflect how likely a user is to issue the query t. We could assume that $P(t)$ is uniformly distributed, but preliminary experiments have shown that this did not work well. Another option would be to use query logs to estimate the probability of a term to occur in a query. However, query logs (with non

Algorithm 6 NextBest(P_t, S, W, m): Get the next best posting for term t

Input: P_t list for term t $(d_i, p(t|d_i))$ ordered by decreasing $p(t|d_i)$
S the list of selected postings
W sets of aspects as tuples (t_w, p_w)
m a mapping between documents and a subset of W
Output: Next best posting for term t along with the delta in the criterion

```
1: maxdelta ← 0
2: maxdoc ← 0
3: r ← empty map, Δ ← empty map
4: for w ∈ W do
5:    r_w ← t_w
6:    Δ_w ← 0
7: end for
8: for i = |P_t| → 1 do
9:    score = 0
10:   for all w ∈ m(d_i) do
11:      if p_{t,d_i} ∈ S then
12:         Δ_w ← Δ_w + (c(r_w + 1) − c(r_w)) × p(t|d_i)
13:         r_w ← r_w − 1
14:      else
15:         score ← score + (Δ_w + c(r_w + 1) × p(t|d_i)) × p_w
16:      end if
17:   end for
18:   if score > maxdelta then
19:      maxdelta ← score
20:      maxdoc ← d_i
21:   end if
22: end for
23: return (i, maxdelta)
```

anonymized queries) are not publicly available.

In this work, we circumvent the problem of estimating $P(t)$ by optimizing each term independently of each other. This amounts at setting $P(t)$ such that we preserve a pre-determined number of postings for each term.

Algorithm 7 gives the pseudo-code following this approach. For a given index with vocabulary V, we compute the top-k best postings by calling *NextBest* defined in the previous section. After each call, we update the aspects where the selected posting appear to increment the size of the list associated with the aspects. We keep in the index the top-k postings and discard the other postings. If k is the same for all terms in the vocabulary, it is called *uniform top-k pruning*.

Our approach can be also used with other pruning methods like the ones proposed in [CLTH12, TC11] by defining a global threshold ε rather than choosing top-k for each term. In that case, Algorithm 3 would need to be updated, but we don't detail this in this chapter.

Algorithm 7 *Diversified Top-K Pruning*

Input: for each term, the desired number of postings k_t, the ordered list of postings P_t, the set of aspects W_t and the mappings m_t between documents and the subsets of aspects.
Output: P the optimal set of postings

```
1:  P ← ∅
2:  for all t ∈ V do
3:      S_t ← ∅
4:      while |S_t| < k_t do
5:          (i, Δ) ← NextBest(P_t, S_t, W_t, m_t)
6:          S_t ← S_t ∪ {p_{t,d_i(P_t)}}
7:          for w ∈ m_t(d_i) do
8:              r(w) ← r(w) + 1
9:          end for
10:     end while
11:     P ← P ∪ S_t
12: end for
13: return P
```

6.4 Temporal Static Index Pruning

In this section, we show how to use diversification based static index pruning to ensure temporal diversity in temporal collections. Documents and queries are associated to temporal aspects (*i.e.* a set of time spans). We use [BBAW10] as main time model to define temporal aspects and the mapping the aspects to documents and queries.

We define a temporal aspect as a *time window*, i.e. a time interval. A query (resp. a document) can refer to a set of temporal aspects W_q (resp. W_d). While we suppose that the set of temporal aspects for documents is known (e.g. the validity time interval in web archives), this is not the case for queries since we do not know them in advance.

More precisely, given a term t, we now need to determine what are its different temporal aspects W_t and their importance by defining a distribution over W_t as discussed in Section 6.3.1.

In the following, we define three different strategies. The first two are based on fixed-size time windows spanning the entire time range. The only parameter is the duration (size) of time windows. However, this strategy may fail when the term is related to events whose temporal

span vary. For example, the term "olympics" depend on the year, on the other hand, the term "earthquake" can take a place any time and its time span can vary based on the collection and the magnitude of the earthquake. To handle this case, a dynamic strategy based on a mixture of gaussians is proposed.

Temporal expressions, and how to match a document to a query were presented in detail in Chapter 4. We use the same models in this chapter. We explain in detail our three different strategies to define the temporal aspects W_t of a term.

6.4.1 Temporal Aspects Model

We now turn to the problem of estimating the set of temporal aspects given a term/query t, and the distribution $P(w|t)$, as needed by Equation 6.6. This is the last part we need to define in order to perform static index pruning.

For each term-query t in the vocabulary of the document collection D, we can associate a time series, S_t by identifying all the documents in the temporal collection that contain the term t. Then we take the sum of the term frequencies based on a chosen granularity (*e.g.* hour, day, week etc.) according to the documents temporal part d_{time}. If the document spans various time intervals, we add it to each one. In our work, we used the day granularity. Figure 6.1 shows a time series generated for the term 'disaster' on Los Angles Times collection of TREC.

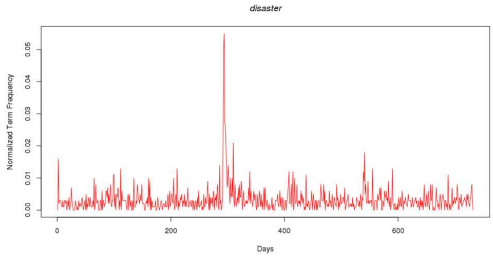


Figure 6.1: Histogram for term 'disaster' in Los-Angeles Times (TREC)

Fixed-sized Windows

In this model, the time series of a term is partitioned into equal, fixed sized (γ) windows.

Selecting the correct window size is an important issue for fixed-windows. When a small size is chosen, the number of total windows will increases and this becomes a bottleneck for big data collections. When a big size is chosen, with the decreasing number of windows, we will slowly start to diversify less. In setting the window size, we suppose that each term has a different time

pattern. Hence, instead of using an uniform window size for all the terms, we propose to use an adaptive window size for each term, denoted γ_t.

There have been many attempts to choose a good bin width for histograms like ours. Scott [Sco79] proposed an approach based on the standard deviation, that was improved by Freedman-Diaconis [FD81]. The latter work replaces the use of standard deviation with the *interquartile range* (IQRN), the distance between the first and third quartile. The Freedman-Diaconis rule is known to be more robust to outliners, which is important for us to detect events in our timelines, and thus we adopted this method. Formally, we use the following window size for term t:

$$\gamma_t = 2IQRN^{-1/3}$$

The simple window strategy consists in using fixed sized non-overlapping windows. Formally, a window is represented as $w_k = [s + k \times \gamma_t, s + (k+1) \times \gamma_t]$ where s is an offset and $k \in \mathbb{Z}$. We further assume that $P(w|t)$ follows uniform distribution, i.e. $P(w|t) = \frac{1}{N}$ where N is the number of non empty windows.

The sliding window strategy uses overlapping windows of fixed length. Formally, a window is represented as $w_k = [s + k/2 \times \gamma_t, s + (k/2+1) \times \gamma_t]$. As in the previous model, we assume that $P(w|t)$ follows a uniform distribution.

Figure 6.2 shows simple windows (in blue on the left) and sliding windows (in green on the right).

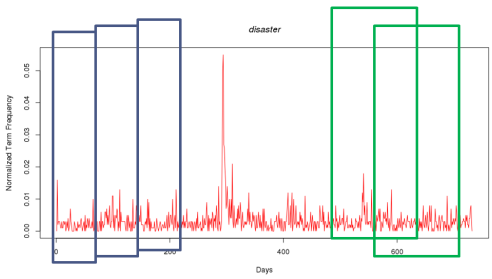


Figure 6.2: Example for fixed windows

Dynamic Windows

In the above models, we considered windows of fixed size. However, events vary in duration. In the dynamic windows model we do not force the windows to have a fixed size but we detect the time windows based on the distribution of the documents over time.

In order to do this, we use a probabilistic clustering based on Gaussian mixture models (GMM) of the data distribution. This probabilistic model assumes that all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. Each distribution can be thought of as a cluster. The probability of generating a time stamp x with the GMM is given [Bis06]:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \sigma_k^2)$$

where the parameters π_k are the mixing coefficients, which must sum to 1 and $\mathcal{N}(x|\mu_k, \sigma_k^2)$ is a Gaussian distribution defined by its mean μ_k and variance σ_k^2.

A standard methodology consists in using the Expectation Maximization (EM) algorithm to estimate the finite the mixture models corresponding to each number of clusters considered and using the Bayesian Information Criteria (BIC) to select the number of mixture components, taken to be equal to the number of clusters [FR98]. The EM algorithm is an iterative refinement algorithm used for finding maximum likelihood estimates of parameters in probabilistic models.

Choosing an appropriate number K of clusters is essential for effective and efficient clustering. The standard way to estimate K is to start with one cluster and slowly increase the number of clusters. At each K, the log-likelihood obtained from the GMM fit is used to compute BIC. The optimal value of K is the one with the smallest BIC. In this paper, this approach is used.

Finally, for each cluster with a mean μ, variance σ^2 and weight ρ, we associate a time window $[\mu - \sigma, \mu + \sigma]$ with $P(w|t) = \rho$.

Smoothing

While we increase the importance of documents associated with several temporal aspects, we do not want to penalize too much highly relevant document associated to a few or only one aspects. In order to do this, we use a smoothing by defining a new window dubbed the *Global window G*. This window contains spans all the time range of term t, i.e. all the documents belong to the global window. Following to this, we redefine the distribution of temporal aspects as follows:

$$P(G|t) = \lambda_w$$

$$P^*(w|t) = (1 - \lambda_w) * P(w|t)$$

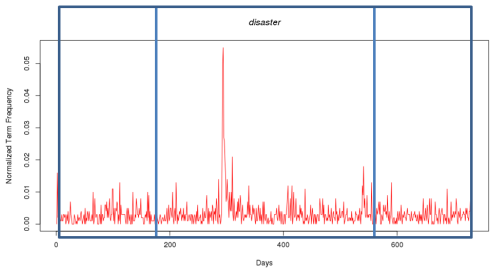


Figure 6.3: Example for dynamic windows

6.5 Experiments

In this section we present our experimental evaluation for the proposed approach. The experiments are conducted by using two publicly available collections presented briefly in Chapter 4:

- TREC Los Angeles Times (LATimes) [TRE04] that contains approximately 132000 articles published in Los Angeles Times newspaper in 1989 and in 1990. LATimes contains time-stamped documents but no query and relevance judgments with temporal dimension. We propose two different simulations to add temporal dimension to the collection: *All relevant Time constraint* and *Time constraint Test*
- The English Wikipedia (WIKI) as of July 7,2009 that contains total of 2955294 encyclopedia articles. This collection is used in time-aware retrieval model tests [BBAW10, KN11]. It contains queries with time constraints and temporal expressions extracted from document but no time-stamped documents.

  In this collection, the temporal expressions are extracted from the text and used as the validity interval of documents.

Settings: Documents are indexed, retrieved and pruned using Apache Lucene[3]. Our implementation does not update the document lengths after pruning. The effectiveness is measured with mean average precision (MAP) and normalized discounted cumulative gain (NDCG), two standard metrics in ad-hoc IR [MRS08]. Both are biased towards top-ranked results, but they exhibit a different behavior. BM25 is used as the scoring function as detailed in Chapter 5.

The pruning ratio reported in our results is the percentage of postings in the index removed by the algorithms. When the query contains a time constraint, we use it to filter out documents for which no time window intersects one of the query, i.e. $q_{time} \cap d_{time} = \emptyset$.

[3] http://lucene.apache.org/

6.5.1 TREC Los-Angeles Times (LATimes)

In addition to the *Time constraint Test* explained in the previous chapter, we also conducted (*All relevant Time constraint*) to check the behavior of our pruning methods. We consider any document from the specified time period containing at least one of the keywords to be relevant.

Figure 6.5 plots MAP, NDCG and P10 measures as a function of the amount of pruning for weekly temporal queries for All relevant Time constraint test. The first observation is that for all kind of queries TCP significantly performs worse than other methods. For queries with pruning ratio smaller than 50%, our three proposed approaches performs slightly better than others. However, 2N2P does better at higher pruning level.

In our second test, called *Time constraint Test*, documents out of the time interval are considered as non-relevant but we kept the original relevance for the documents in the time interval. As shown in Figure 6.4, 2N2P performs better at high pruning level for both long and short queries while our approaches is slightly better at low level pruning. The difference with respect to *All related Time constraint Test* is that IP-u stays behind TCP and the gap between 2N2P and our methods gets bigger at low pruning level.

6.5.2 English Wikipedia Dump (WIKI)

The LATimes collection was used in preliminary experiments to confirm that our approaches perform similarly to the existing ones; the Wikipedia corpus was then used to perform more realistic experiments.

Figure 6.6 plots MAP and NDCG as a function of the amount of pruning for inclusive and exclusive temporal queries. The results for initial index and the results obtained with topics 301-450 without temporal dimension are coherent with the baseline methods [CLTH12, CCF$^+$01]. We can observe than that at low pruning levels, the performance can be higher for pruned indexes than with non pruned ones. This just shows that pruning removes postings that would have made non relevant documents retrievable.

We now turn to comparing the different pruning methods. TCP, which performs worst with LA-Times collections, significantly outperforms both IP-u and 2N2P. Two of our methods, namely *Simple* and *Sliding*, perform better than all the others whatever the query type(inclusive or exclusive), and the gap between methods increases with the pruning level. This shows that our time-based diversification method works well.

However, our third method, based on dynamic windows, did not perform as well contrarily to our expectations. It can be related to number of clusters chosen as input for GMM as discussed in Section 6.4. The clusters obtained did not cover all the documents for some terms - in this

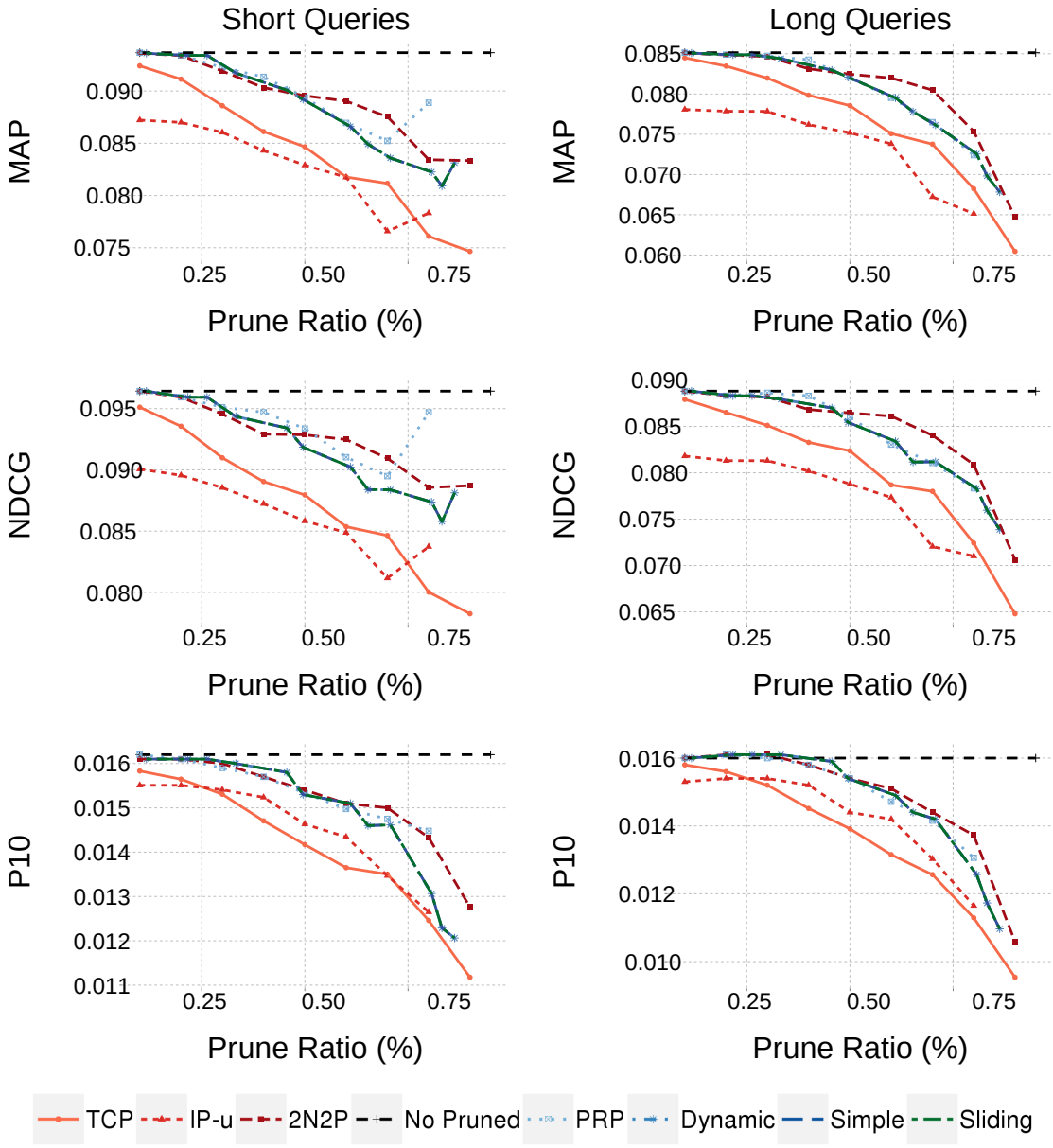


Figure 6.4: Time constraint Test results

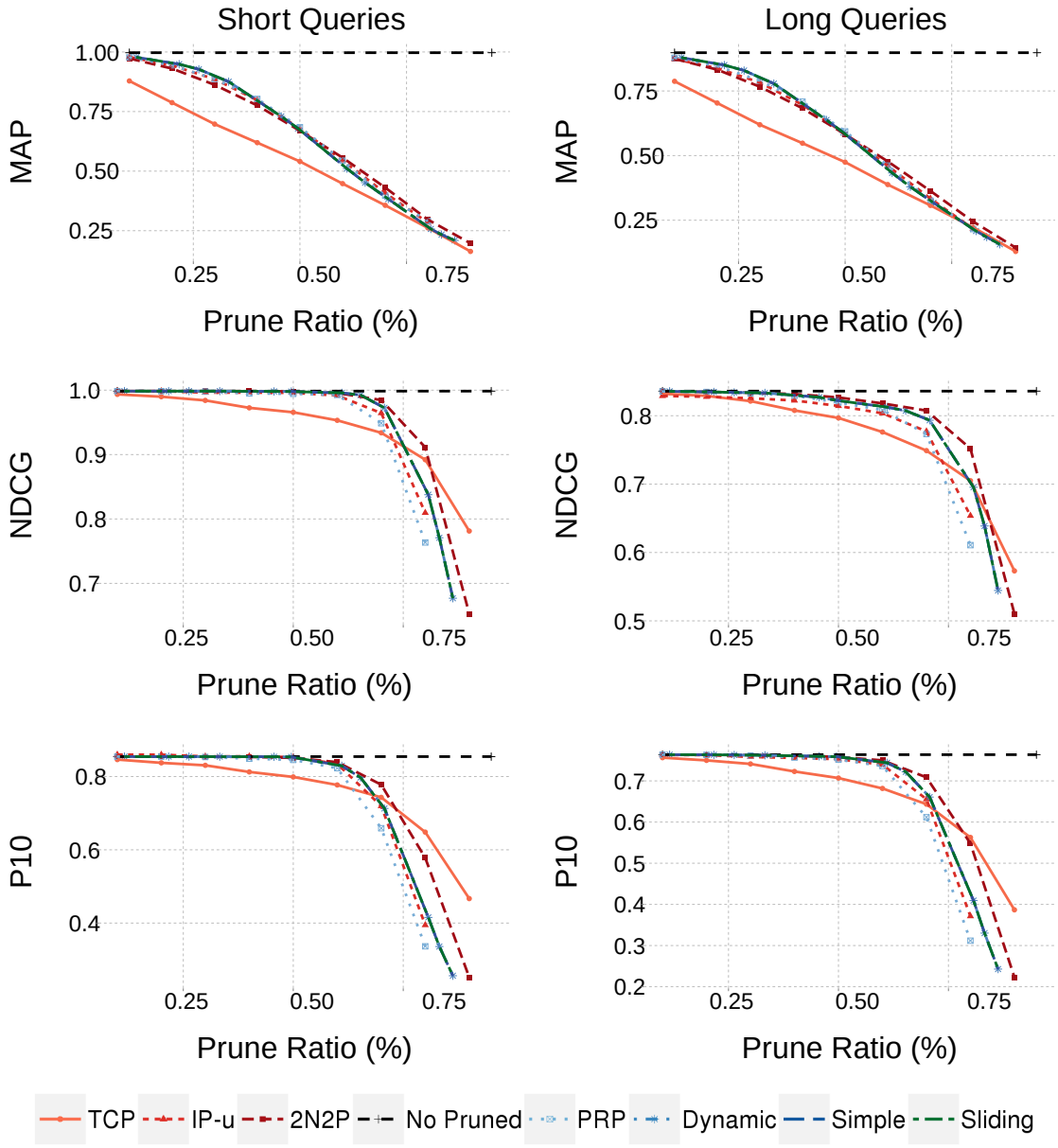


Figure 6.5: All relevant Time constraint results

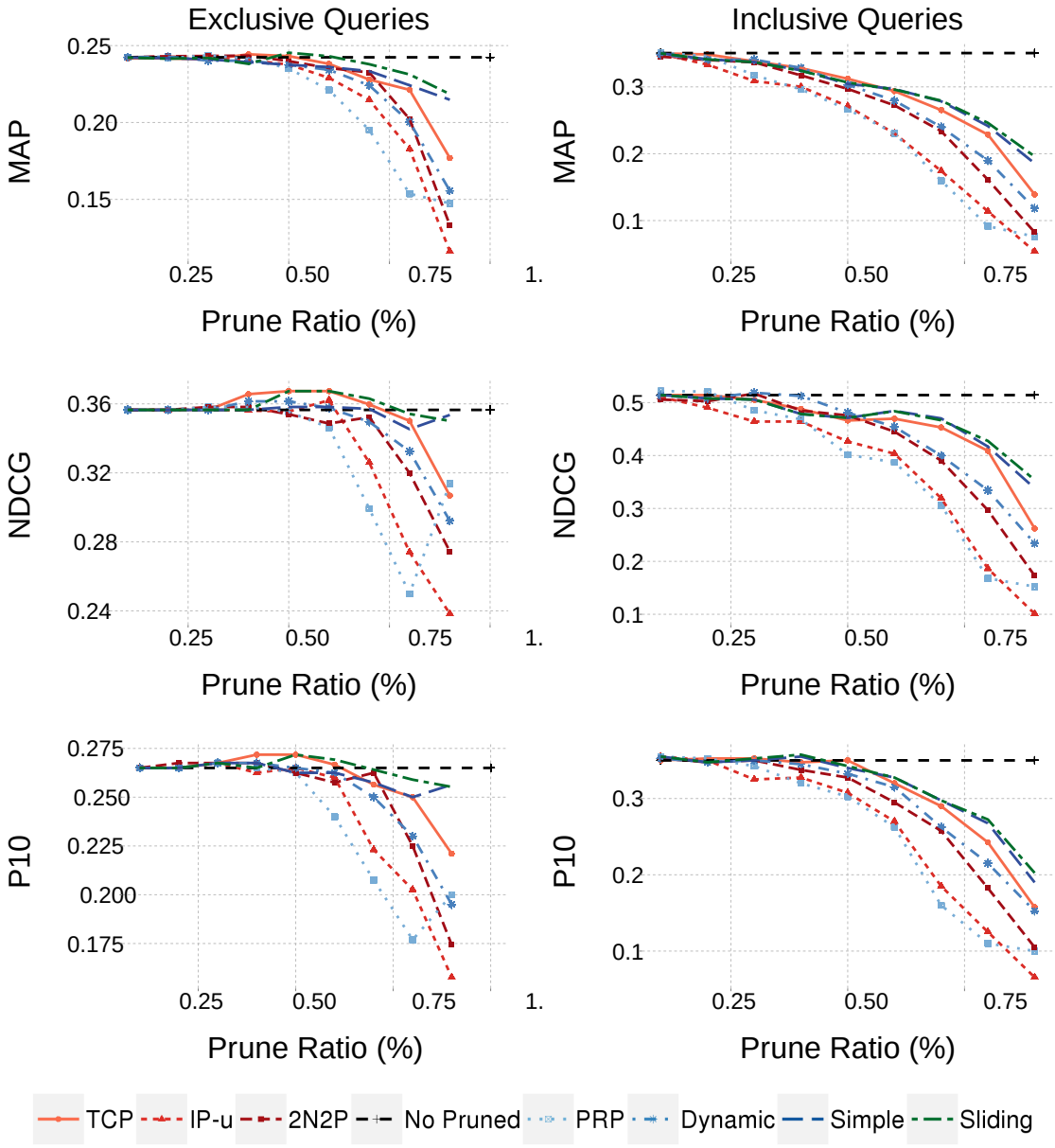


Figure 6.6: WIKI overall results

case, we chose the closest cluster.

6.6 Conclusion and Discussion

We studied the problem of search result diversification in the context of index pruning. We proposed a new approach called diversification based static index pruning that decides which postings to prune by taking into account both the (estimated) relevance of the documents and the diversification of the results: For each term, we preserve the top-k postings that maximize a chosen IR metric (DCG in this paper) using a greedy algorithm, and discard the rest. In order to perform this pruning efficiently, we proposed an algorithm for selecting the next best posting to preserve.

Our original motivation was to explore index pruning strategies for temporal collections, since these collections are characterized by large index sizes. We thus applied our diversification-based index pruning method to take into account the temporal dimension. This was achieved by associating to each possible term-query a distribution over time windows (time range with a start and end date), based on the time windows associated to each of the documents containing the term at hand. We proposed three different approaches to compute namely Simple, Sliding and Dynamic windows.

Our experiments were conducted on two publicly available collections: LATimes and WIKI. The LATimes collection does not contain the temporal information needs, and was used to show that our approach is competitive with state-of-the-art pruning algorithms. The experiments on WIKI showed that Simple and Sliding methods perform better overall for temporal queries, but we were unable to show good results with the more sophisticated dynamic strategy.

This work is the first one dealing with the (temporal) diversification based index pruning. As such, many different extensions and enhancements of our approach are possible and will be explored in future work.

CONCLUSIONS 7

Today we live in a digital world in which information can be communicated rapidly and efficiently on the Internet, over the web. It also means that our cultural heritage is taking a new digital form everyday. However, all the information found on the web has an ephemeral nature, *i.e.* they can disappear or be modified at any moment. As noted by the UNESCO[1], disappearance of heritage in whatever form constitutes an impoverishment of the heritage of all nations. Thus, national libraries and non-profit organizations have been crawling the web since the late 90s to archive it.

The purpose of preserving the digital heritage is to ensure that it remains accessible to the public. The next step is to proceed means of access to these archives, and this has barely started to attract more and more attention. This thesis studied some problems encountered for accessing web archives and focused on different points; namely, new access methods and optimization methods. More specifically, we addressed the following challenges and contributed the following:

- To satisfy information needs of web archives users, we proposed a data model, as well as operators to manipulate them, as the basis of a query language for web archives. This model is inspired from existing models like [RGM03, MMM97]. However, it differs in two different ways: 1) we take different topics in web pages into account by using visual blocks as an unit of retrieval with corresponding importance 2) we propose new operators with temporal dimension taking the web archives issues into account.

- We proposed a new coherence-oriented navigation method. We first started by describing the coherence in web archives and the issue of coherence while navigating. Then, we proposed a new method based on time patterns that enables users to navigate through

[1] https://en.unesco.org/

more coherent versions. The experiments were conducted on simulated data and showed that it improved coherence while navigating.

- We proposed a new change detection algorithm that detects differences between two web pages (or two versions of the same page) based on their visual representation. Preserving the visual structure of web pages while detecting changes gives relevant information to understand modifications which is useful for many applications dealing with web pages. We also presented the different applications of web change detection.

- Pruning is crucial for web archives since they grow quickly and their index does not fit into the memory. We hence studied how the existing static index pruning methods deal with temporal queries. We also discussed the relation of results to temporal diversification.

- We proposed a new static index pruning method based on diversification. It is the first study that integrates diversification within the pruning context. Our aim was to preserve retrieval effectiveness and diversity by maximizing a given IR evaluation metric like DCG while pruning. We showed the application of this new method on time-diversification and proposed three different models to define temporal aspects. The experiments were conducted on two publicly available dataset and show that our approach improves over the traditional pruning methods.

In the rest of this chapter, we will outline our plans for future work, and discuss possible research topics beyond what has been addressed in the thesis.

7.1 Outlook

In this section, we give an outlook for our contributions and in the next section we discuss other related research topics.

Querying: The first important task is to implement our language model proposed in Chapter 2 with an appropriate user-friendly syntax. We want to underline the fact that in this thesis we clarify the requirements of WAC query language formally. It can be implemented as a new query language or as an extension of an existing query language. For example the Pig project and its associated language, Pig Latin, seem appropriate for this: Since a few years, web archiving institutions and researchers turned their attention to the solutions based on Hadoop on which Pig based. For instance, the Internet Archive processes their archives using Hadoop and Pig. The Apache Hadoop software library is a framework that allows for the distributed processing of

large data sets across clusters of computers using simple programming models[2]. Pig is a platform for analyzing large data sets and is built on Hadoop and provides ease of programming, optimization opportunities and extensibility. Pig Latin is a relational data-flow language and is one of the core aspects of Pig. It provides extensive support for user defined functions as a way to specify custom processing that we can implement our new operators with.

Navigation: We proposed a coherence-based navigation based on patterns which is simple time model for changes on the web pages. Thus, a better temporal model to estimate web page changes can be exploited for coherence oriented navigation. Further improvements imply working on *a priori* and *a posteriori* solutions.

To ensure a better *a posteriori* coherence, we would need to go beyond treating navigation as a one-step process, *i.e.* between two pages. Indeed, navigation is a process that consists in following several successive hyperlinks. This raises other research problems like getting lost in time as explained in Chapter 3. Figure 7.1 gives an example: Consider a user who starts to navigate from URL1 at time t. By following the most coherent version at each step (*e.g.* the arrows in green), the user can find herself/himself away from the start point t. When the coherence is measured between pages, another problem raised is to decide which versions to consider (*e.g.* arrows in red) and how to aggregate the coherence measure.

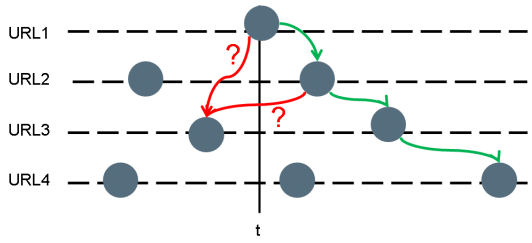


Figure 7.1: Navigation illustration

To ensure a better *a priori* coherence, we could leverage recent works [FLL08, Lev11] for the web that models how users navigate the web topology by attaching probabilities to links, giving rise to a probabilistic automaton, *i.e.* a Markov chain. They present a new metric called *potential gain* for measuring the navigation potential of a web page. By using this metric, they semi-automate the navigation by probabilistically constructing a tree whose most relevant trails are presented to the user. This can be used to optimize crawling as crawlers follows hyperlinks on the pages, by defining a threshold and letting crawlers follow the hyperlinks with a potential gain greater than this threshold. In addition, by extending this model, we can define a temporal potential gain to predict the navigation on the web archives in order to reduce the user access latency problem of web archives.

[2] `http://hadoop.apache.org/`

Change Detection: From the "Preservation" point of view, change detection is necessary to ensure the quality of archives (*e.g.* after format migration like ARC to WARC), to detect format obsolescence due to evolving technologies. We are actually working on integration of Vi-DIFF algorithm on European Project SCAPE for quality assurance on web archives.

Immediate extensions for our work are as follows:

- detecting splitting and merging of blocks as structural changes
- extending simulator to generate new versions including structural changes
- proposed algorithms can be optimized by using more efficient methods *e.g.* by using semantic hashing function [SH09]. It outputs binary strings for documents such that similar documents result in strings with a low Hamming distance[3].

Reducing access time: Extensions on static index pruning is multifold. First, from a model point of view, we could use more realistic distributions over queries (*e.g.* using query logs, or term co-occurrence information) and over time windows (*e.g.* by using other clustering algorithms).

Second, we can explore the behavior of index pruning with more sophisticated time-aware ranking models like those exposed in [BBAW10], where the matching between time intervals is not boolean anymore but is real-valued. This approximate matching could be exploited to reduce further the index size.

Finally, one of the challenges in the context of web archives, besides the temporal dimension, is redundancy: different versions of the same document can be present in the document collections. The changes between the versions are likely to be small for most of documents, and this should be exploited by keeping only a few (ideally one) of the postings (for the same term) corresponding to the different versions of the document. This would however require the development of a specific test collection based on web archive crawls with related temporal queries and relevance judgments.

7.2 Open research topics

The research problems that we addressed in this thesis are among several of the problems that need to be dealt with for efficient access to web archives. Beyond the topics of this thesis, there

[3]The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different

are still a lot of opportunities for further investigation. In the following, we outline two promising topics for future research:

Retrieval Models for Web Archives: A time-aware ranking method ranks documents that are textually and temporally similar to a query and ranks retrieved documents with respect to both similarities. Most of the proposed time-aware ranking models focus on fresh information retrieval. For example, language models that take into account publication times of documents, in order to favor more recent documents are proposed in [DGI08, LC03, EG11]. All these methods use the publication date as temporal dimension. Recently, [BBAW10, KN10] proposed methods for temporal information needs by using both document publication date and temporal expressions in the documents. They adapt the well-known retrieval models for web archives without taking into account the redundancy, incompleteness and coherence in web archives. Hence, designing ad-hoc retrieval models for web archives is an interesting future work direction. For instance, such retrieval models can find a set of relevant and diverse (temporally and textually) results.

Web Archives Recommender Systems: With the explosive growth of information available on Web, it has become more challenging to find relevant information. Recently, Web-based Recommender Systems (WRS) were widely applied in order to guide the user and improve the usability of the Web. However, as far as we know, it has never been studied in the context of web archives. Many data mining techniques such as association rule mining, sequential pattern discovery, clustering etc. used for WRS should be extended based on temporal dimension to be used in web archiving context in order to guide the user and improve the usability of the Web archives.

Part III

Appendix

SUMMARY IN FRENCH A

L'objectif principal de cette thèse est d'étudier comment accéder aux archives web et comment optimiser les méthodes d'accès. Dans ce chapitre, nous décrivons nos motivations et les questions de recherche étudiées dans cette thèse. A la fin de ce résumé, nous présentons l'organisation de la thèse.

A.1 Motivations

Le web joue un rôle crucial dans notre société de l'information en fournissant des informations pour tous types d'événements, des opinions et des développements au sein de la société. Aujourd'hui, le web est un miroir de la population qui l'utilise. Cependant, le web a un caractère dynamique qui signifie que les pages, et souvent des sites entiers, sont en constante évolution, *i.e.* les pages sont fréquemment modifiées ou supprimées. Ntoulas et al. a découvert que 80% des pages web ne sont plus disponibles après un an. L'archivage du web est devenue une nécessité culturelle de préserver les connaissances pour la prochaine générations. Les archives web qui contiennent des milliards de versions de pages obtenues en téléchargant et en revisitant les pages web, représentent donc une source d'information énorme, potentiellement supérieure au web lui-même. Cela fait de l'archivage du Web un domaine de recherche actif avec de nombreuses questions ouvertes comme l'otimisation du crawling, les modèles de stockage etc. Une vue d'ensemble des principaux problèmes est présentée dans différents travaux [Mas06, HY11, WNS+11]. Figure A.1 représente les principaux enjeux du archivage du web.

Internet Archive (IA) est l'archive web pionnière fondée en 1996 en tant que société à but non lucratif à San Francisco, USA. IA contient environ 10 pétaoctets de données et se développe

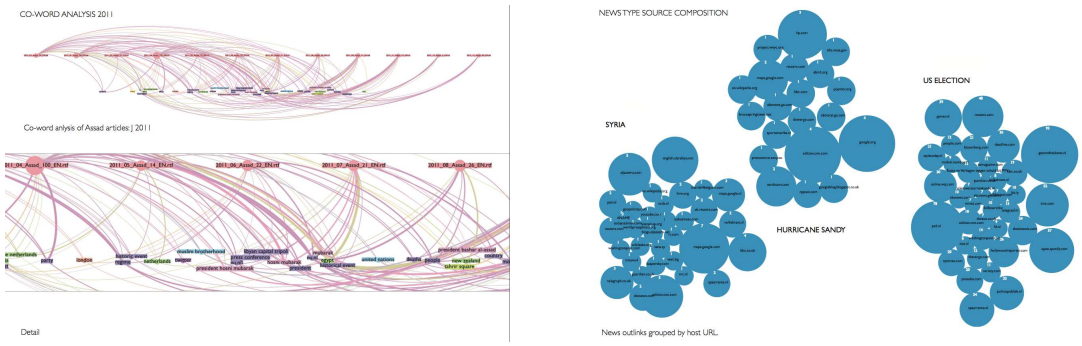
au rythme d'environ 100 téraoctets par mois (données de 2012). En automne 2001, IA a lancé son projet en collaboration avec Alexa Internet appelé "Wayback Machine". Il permet aux utilisateurs de voir des versions capturées de pages Web au fil du temps et c'est le plus connu des méthodes d'accès aux archives web. Cependant, il exige que les utilisateurs sachent à l'avance URL exacte à rechercher. Pour une URL donnée, les résultats sont affichés dans un calendrier en fonction de leurs dates de capture et l'utilisateur peut parcourir une version de la page. Les autres méthodes actuellement fournis sont des méthodes d'accès Web bien connues : la recherche plein texte et la navigation dans l'intervalle de temps requis.

Le nombre croissant des archives nationales web et la diversité des recherches connexes a conduit à la création de l'International Internet Preservation Consortium à Paris en 2003. Le but de l'IIPC est de développer des normes communes, outils et techniques pour l'archives web. Un des projets actuels de l'IIPC, appelé WERA, est une solution d'accès aux archives pour la recherche et la navigation. Il permet une recherche plein texte en addition au wayback machine. Ces méthodes sont suffisantes pour les utilisateurs occasionnels qui représentent la plus grande proportion des utilisateur du web. Selon Web Archiving User Survey [RvB07a], les profils des utilisateurs d'archives web se composent des historiens, des journalistes, des avocats, des étudiants, etc plus que d'utilisateurs occasionnels. Il est également souligné que la principale raison de l'utilisation des archives web est une activité de recherche : les utilisateurs d'archives web ont besoin d'analyser, de comparer et d'évaluer l'information. Pour atteindre cet objectif, les systèmes d'archivage Web doivent fournir des outils pour exécuter des requêtes complexes. Par example, classement des résultats par pertinence en prenant en compte la dimension temporelle pour les requêtes temporelles (mots clés avec des contraintes de temps), comme "Turquie tremblement de terre@[1998-2000], est le défi le plus étudié au monde de la recherche sur les archives web.

FIGURE A.1 – Key issues in web archiving [HY11]

D'autre part, les différents projets et les bibliothèques nationales se sont interressés davantage à l'accès depuis quelques années. UKWAC (Archives de la Bibliothèque nationale web Angleterre) a de nouvelles fonctionnalités, telles que montrer des résultats sous forme de vignette d'instantanés, comme Wall 3D ou visualisation de n-gramme qui enrichissent visuellement les expériences des utilisateurs. Récemment, WebArt (web archive retrieval tools) [1] a proposé de nouvelles interfaces qui permettent de visualiser la fréquence, l'analyse des cooccurences des mots comme le montre la Figure A.2.

[1] http//www.webarchiving.nl/



(a) Co-Word analysis (b) Outlink analysis

FIGURE A.2 – Example of webART interfaces

Les archives web croissent rapidement. Par example, les archives web de la Bibliothèque du Congrès croissent à un rythme d'environ 5 téraoctets par mois. Il y a clairement un besoin pour un accès rapide et efficace vu la taille croissante des archives, comme les rchives du web finlandais pour 148 millions de contenus (par exemple des pages html, PDF, images, etc), Canada Wa 170 millions de contenus, Patrimoine numérique de la Catalogne pour 200 millions de contenus Pour mettre à jour les systèmes d'indexation, certaines approches changent la structure des index inversés en ajoutant une dimension temporelle, en plus de celle que proposent différentes méthodes de partitionnement. Des versions idenqisue d'un même document sont créées lorsque les documents sont récoltés à plusieurs reprises à partir des sites web sélectionnés. Cela induit une perte de mémoire et il va devenir très fastidieux pour ceux qui utilisent une archive web si de nombreuses copies d'un même document web sont présentés à l'utilisateur. Par conséquent, la possibilité de définir quand un document a changé est un autre défi pour l'archivage du web.

L'archivage du Web est une discipline vaste et en expansion. Différents défis et orientations futures de la recherche sont soulignés dans les travaux récents [Kan09, WNS+11]. Il est clair qu'il existe un besoin pour des approches pratiques et efficaces pour l'accès aux archives web. L'objectif de cette thèse est d'identifier et d'étudier certains des problèmes liés étant donné le vaste du sujet et proposer des approches que les solutions à ces problèmes. Dans la section suivante, nous allons présenter les problèmes de recherche abordés dans cette thèse.

A.2 Questions de recherche

Tel que mentionné dans la section précédente, la question principale est *comment accéder aux archives web et comment optimiser cet accès ?* Les motivations montrent que le sujet est vaste et large, donc nous avons réduit notre recherche aux quatre principaux sujets suivants : *Interrogation*, *Navigation*, *Détection des changements*, *Réduction du temps d'accès*.

A.2.1 Interrogation

Comme nous l'avons mentionné précédemment, le profil des utilisateurs d'archives web diffère de celui des utilisateurs occasionnels : ils ont besoin d'analyser, de comparer et d'évaluer l'information. Ainsi, pour l'analyse et l'exploitation efficace, une interface qui prend en charge les requêtes déclaratives et complexes est nécessaire. Ces requêtes doivent être évaluées en interprètant en même temps comme une collection temporelle des documents de texte, en tant que graphe navigable orienté, et comme un ensemble de tables relationnelles stockant des métadonnées des pages Web (longueur, URL, titre, etc.). Le classement (ranking) et les opérateurs relationnelles doivent être également définis sur les pages et les liens pour filtrer et récupérer uniquement les meilleurs résultats pour les requêtes en prenant la dimension temporelle en compte. Ainsi, la première question de recherche que nous nous posons est :

> *RQ2 : Quelles sont les exigences, le modèle de données et les opérations pour définir un langage de requête pour les archives web ?*

A.2.2 Navigation

La navigation est l'une des méthodes de recherche utilisés sur le web. Elle permet de suivre des liens hypertexte et de naviguer sur des pages Web à partir d'une page Web source. Sur les archives web, chaque page doit être reconstruit. Chaque hyperliens dans les pages web archivées sont reconstruits pour permettre aux utilisateurs de naviguer comme *en temps réel* sur le web. Supposons une page web crawlée le 19 Août 2009 contenant un hyperlien vers " www.lip6.fr". Lorsque l'utilisateur commence à naviguer à partir de cette version et suit ce lien hypertexte, elle ne doit pas être redirigé vers le " www.lip6.fr" d'aujourd'hui. Voilà pourquoi toutes les pages Web des liens hypertexte sont modifiées afin de pointer vers les versions correspondantes temporellement dans l'archive. Le défi commence ici parce que les archives web peut ne constituent pas une copie exacte du web tel qu'il était à tout moment - parce qu'elle sont intrinsèquement incomplètes. Cela signifie qu'il n'est pas possible de *geler* le web tout en crawlant pour empêcher les pages de changer, et en même temps il n'est pas possible d'analyser tout le web tout le temps. Si les liens hypertextes de la page source ne sont pas analysés en même temps de la page source, à quelle version l'utilisateur doit être dirigé ? Ainsi, la deuxième recherche question qui nous interesse est :

> *RQ2 : Comment optimiser la navigation pour permettre aux utilisateurs de naviguer à travers les versions les plus cohérentes par rapport au temps de la requête ?*

A.2.3 Detection des changements

Detecter les changements entre les versions de la page est un enjeu important dans le contexte de l'archivage Web. Ce sujet touche à plusieurs questions clés présentées dans la Section A.1. Une approche de détection de changement efficace permet d'optimiser les crawleurs en décidant si la page doit être crawléz ou non à la volée. Différents modèles peuvent être définis en fonction des changements sur les pages et un robot peut être programmé en fonction de ces modèles. Par exemple, si une page contient une mise à jour à un instant donné, alors il est bon de la crawler juste après ce point de temps. [SPG11]. Cependant, le crawling est un processus ouvert aux erreurs. Il peut arriver que les données essentielles est manquée par le robot et donc pas capturée et conservée. Pour créer une archive web de haute qualité, en faisant l'assurance de la qualité est essentielle, par exemple, en comparant la version live de la page avec la juste rampé un. Pour la question clé stockage, une approche de détection de changement permet d'éliminer (ou presque) les doublons déconnecté d'augmenter l'efficacité de la recherche en diminuant la taille de l'archive. Du point de préservation, la détection de changements est nécessaire pour garantir la qualité des archives (par exemple après une migration de format de ARC à WARC), pour détecter l'obsolescence des formats SUITE à l'évolution des technologies. En ce qui concern l'accès, l'évoluation de pages Web peut être présentée visuellement en montrant les changements par rapport aux versions précédentes. Par conséquent, la question de recherche suivante est :

> *RQ3 : Comment savoir, de comprendre et de quantifier ce qui s'est passé (et donc changé) entre deux versions d'une page Web ?*

A.2.4 La réduction du temps d'accès

Une façon courante d'accéder aux archives web est l'utilisation de requêtes temporelles qui combinent mot-clés avec des contraintes temporelles. La taille croissante des archives web nécessite de grands espaces disque, qui conduit au grande fichiers d'index qui ne tiennent pas dans en mémoire. En conséquence, le moteur de recherche a besoin de beaucoup d'accès disque, ce qui augmente le temps de réponse aux requêtes. *L'élagage statique* réduit considérablement la taille des indexes sans modifier significativement les performances de récupération en supprimant les entrées de l'index qui sont moins susceptibles de modifier le classement des documents top-k récupérés par l'exécution des requêtes. Il ne nécessite aucune connaissance sur les requêtes puisqu'appliquée hors ligne. Différentes méthodes d'élagage d'index statiques se sont avérés efficace (*e.g.* [CCF$^+$01, BC06, BB07]). Cependant, aucune des méthodes existantes ne prend la dimension temporelle en compte et ils ne sont jamais testés sur des requêtes temporelles. Ainsi, une question de recherche se pose ests :

> *RQ4 : Comment les méthodes existantes d'élagage d'index statiques fonctionnent en présence de requêtes temporelles ?*

Considérons la requête “ France Election présidentielle”. Elle peut générer différentes requêtes temporelles en lui ajoutant différentes contraintes temporelles(*e.g.* @2012, @2007 ou @[2007-2012]). L'index élagagé devrait être en mesure de fournir des résultats qui répondent aux différents besoins d'information temporelle. Idéalement, la méthode d'élagage devrait garder les documents provenant de différentes périodes de temps, *i.e.* de préserver la diversité temporelle après l'élagage. Notre dernière question de recherche est :

> *RQ5 : Comment élagager les indexes en prenant la diversification de temps en compte ?*

A.3 Contributions

Nos contributions se composent d'amélioration d'approches existantes et aussi de nouvelles approches. Dans cette section, nous donnons un résumé de ces contributions en indiquant la question de recherche correspondant et les chapitres où les détails peuvent être trouvés.

A.3.1 Interrogation

> Contribution 1 : Nous proposons un modèle conceptuel, ainsi que les opérateurs permettant de les manipuler, comme base à langage de requête pour les archives web. Dans notre modèle, nous prenons en compte les différents sujets dans les pages web en utilisant des blocs visuels comme une unité de récupération, avec une importance associé. L'approche basée sur les blocs est utilisé pour la recherche d'information sur le web, toutefois, à notre connaissance, elle n'est jamais utilisée avec dimension temporelle. Les opérateurs de navigation avec dimension temporelle permettent aux utilisateurs d'exécuter des requêtes sur des archives web structure hypertexte temporelle. Le modèle et les opérateurs enrichis avec la dimension temporelle permettent de faciliter l'interrogation archives web. Cette contribution est une solution à RQ1, qui seront discutés en détail dans le Chapitre 2.

A.3.2 Navigation

> Contribution 2 : En exploitant des patterns réguliers, nous proposons une nouvelle navigation axée sur la cohérence qui améliore la qualité de la navigation dans les archives web en permettant aux utilisateurs de naviguer dans la page la plus cohé-

rente versions à la fois de la requête donnée. Cette contribution est une solution à RQ2 sera discutée en détail au Chapitre 3.

A.3.3 Detection des changements

Contribution 3 : Nous proposons un nouvel algorithme de détection de changement qui détecte des différences entre deux pages web (ou deux versions de la même page) sur la base de leur représentation visuelle. Préserver la structure visuelle des pages Web tout en détectant les changements permet de comprendre les modifications, ce qui est utile pour de nombreuses applications traitant de pages Web. Il détecte deux types de changements, les changements structurels et le contenu. Cette contribution est une solution à RQ3 discutée en détail dans le Chapitre 4.

A.3.4 La réduction du temps d'accès

Contribution 4 : Nous effectuons la première étude sur une comparaison empirique des méthodes d'élagage d'index statiques pour des requêtes temporelles. En outre, nous étudions, par des test statistiques, la relation entre la diversification temporelle et la qualité de l'index après élagage. Cette contribution est une solution à RQ4 discutée en détail dans le Chapitre 5.

Contribution 5 : Nous proposons la première méthode pour l'élagage statique l'index qui prend la diversification de résultats en compte. Ensuite, nous montrons comment utiliser l'élagage statique index basée sur la diversification pour assurer la diversité temporelle dans les collections temporelles. Cette contribution est une solution à RQ5 qui sera discuté en détail au Chapitre 6.

A.4 Outline

Cette thèse comprend deux parties differents. La prémière présente des méthodes d'accès aux archives web, tandis que la seconde présente les méthodes d'optimisation de l'accès aux archives web. Chaque partie contient différents chapitres qui correspondent aux différentes questions de recherche.

Dans la première partie, Chapitre 2 présente les conditions requises pour un langage de requête pour accéder aux archives web. Il présente un modèle de données, les opérateurs pour ce lan-

guage. Dans le chapitre 3, la nouvelle méthode de navigation est proposé pour les archives web en prenant la cohérence en compte.

Dans la deuxième partie, Chapitre 4 présente un algorithme de détection de changement et de ses différentes applications dans le cadre de l'archivage web. Le chapitre suivant présente la première étude qui compare comportement de quatre méthodes d'élagage statiques d'index avec des requêtes temporelles. Le Chapitre 6 présente une nouvelle méthode l'élagage statique index basée sur la diversification et on montre son application aux collections temporelles. Nous concluons en décrivant futures directions de recherche.

EXISTING OPERATORS FOR WACQL B

This section present existing operators of WACQL (Chapter 2). Like SQL, the WACQL algebra works on sets/bags of tuples with an additional data type blocks. We present the predicates and the operators.

B.1 Predicates and Functions

In this section, we present two kind of predicates: *temporal predicates* and *logical full-text predicates*.

B.1.1 Temporal predicates

Our model uses Allen's interval-based representation of temporal data [All83b] where intervals are the primitive time elements. Each element is temporally stamped by a time range, called *validity*, defined as a time interval $[t_s, t_e)$ where t_s represents a starting time point and t_e is an ending time point. *now* is introduced to point the current time and is assumed to be larger than any known end time point. In order to explain operators we will use two periods: $p1 = [t1_s, t1_e)$ and $p2 = [t2_s, t2_e)$. Operators have the following common signature as follows:

$$op : period \times period \rightarrow bool$$

- **Before and After**: The before/after operator takes two periods as input, checks if the ending point of the first period is smaller/greater than the starting point of the second one and returns a boolean.

  p1 BEFORE p2 is true if and only if $t1_e < t2_s$, otherwise it is false. p1 AFTER p2 is true if and only if $t2_e < t1_s$, otherwise it is false. p2 AFTER p1 is true if p1 BEFORE p2 is false.

- **Overlaps**: This operator checks if two periods has overlapping sub-period. p1 OVERLAPS p2 is true if and only if $t1_s \leq t2_e$ and $t2_s \leq t1_e$ are both true.

- **Meets**: Two period meet if and only if the ending point of first one is the starting point of second one. p1 MEETS p2 is true if and only if $t1_e = t2_s$. if p1 MEETS p2 is true, p2 MEETS p1 is true.

- **Equals**: This operator checks if two periods are equal. p1 EQUALS p2 is true if and only if $t1_s = t2_s$ and $t1_e = t2_e$ are both true.

- **During**: This operator checks if one period is included in the other one. p1 DURING p2 is true if and only if $t1_s \leq t2_s$ and $t1_e \geq t2_e$ are both true. It worths to note that $p1DURINGp2 \neq p2DURINGp1$

- **Starts**: The starts operator checks if the second period starts with the first period. p1 STARTS p2 is true if and only if $t1_s = t2_s$ and $t1_e \leq t2_e$ are both true.

- **Finishes**: The finishes operator checks if the first period end with the second period. p1 FINISHES p2 is true if and only if $t1_e = t2_e$ and $t2_s \leq t1_s$ are both true.

B.1.2 Logical Text Predicates

AND, OR, NOT, CONTAINS, LIKE are supported.An *AND* finds matches that satisfy all of the operand full-text selections simultaneously. An *OR* finds all matches that satisfy at least one of the operand full-text selections. A *NOT* selects matches that do not satisfy the operand full-text selection. CONTAINS supports complex searches for terms. LIKE compares values using simple pattern matching with wildcards.

B.1.3 Temporal functions

In our model, in addition to Allen's predicates, we define also the following temporal operators:

- **Max/Min**: We define MAX(t1,t2) as returning t2 and MIN(t1, t2) to return t1 if $t1 < t2$.

$$MAX/MIN : period \rightarrow instant$$

- **T-Union**: This operator takes two period as input and returns a period. p1 T-UNION p2 returns a new period $p3 = [MIN(t1_s, t2_s), MAX(t1_e, t2_e))$ if p1 OVERLAPS p2 or p1 MEETS p2 is true, otherwise it is undefined.

$$T-UNION : period \times period \rightarrow period$$

- **T-Intersect**: This operator takes two periods as input and returns the overlapping period for these two periods. p1 T-INTERSECT p2 returns a period $p3 = [MAX(t1_s, t2_s), MIN(t1_e, t2_e))$

$$T-INTERSECT : period \times period \rightarrow period$$

- **Minus**: This operator takes two periods. If they overlap, it returns a new period, otherwise it is undefined. p1 MINUS p2 returns a period $p3 = [t1_s, MIN(t2_s, t1_e))$ if $t1_s < t2_s$ and $t1_e \leq t2_e$ are true. It returns $p3 = [MAX(t2_e, t1_s), t1_e)$ if $t1_e > t2_e$ and $t1_s \geq t2_s$ are true.

B.2 Operators

B.2.1 Unary Operators

The unary operators that we use are listed as follows;

- **Select**: This operator retains only some subset of the data that is of interest according to a given predicate, while discarding the rest. Unlike conventional selection, temporal selection takes a temporal predicate. The predicate is also checked for nested elements.
- **Project**: The projection selects certain attributes from a set/bag and discards the other attributes.
- **GroupBy**: It is often necessary to group the data that is related in some way, so that they can be processed together. This operator needs to be extended for temporal and nested elements.

B.2.2 Binary Operators

The binary set operators that we use are listed as follows:

- **Union**: The union operator is similar to its relational counterpart; it combines two sets. Union calls T-Union for flat timestamped elements. On the other hand, for the (temporal) nested elements (*e.g.* blocks), union is defined recursively.
- **Intersect**: The intersection of two sets A and B is the set/bag that contains all elements of A that also belong to B. Union uses T-Intersect for flat timestamped elements. Like union, it is defined recursively for the (temporal) nested elements.
- **Difference**: The difference of sets/bags A and B is the set/bag of all elements of A which are not also elements of B. For temporal attributes Minus operator is used. Like other set/bag operators, difference is defined recursively for the (temporal) nested elements.
- **Cartesian Product**: For sets/bags A and B, the Cartesian product *AxB* is the set of all ordered pairs (a, b) where $a \in A$ and $b \in B$.
- **Temporal Cartesian Product**: Temporal Cartesian product is a conventional Cartesian Product that is applied independently at each point in time, and requires an intersection of the timestamp attributes (*e.g.* T-Intersect defined in Section B.1.3).
- **Join**: It returns cartesian product of two sets/bags filtered by predicates.
- **Temporal Join**: It returns temporal cartesian product of two sets/bags filtered by predicates.

B.2.3 Aggregation Operators

Typical aggregation operators (count, sum, average, min and max) are supported. “Count” returns the number of elements in a set/bag. “Min” find the minimum value in a set/bag while “Max” finds the maximum value.“Average” computes the average of values in a set/bag. “Sum” gets the total of the values in a set/bag.

B.2.4 Various Operators

- **Rank**: This operator takes a set/bag of blocks as input and orders it by applying a ranking function. This ranking function is different from traditional web ranking functions, because it should take into account the temporal dimension and the block-based structure.

- **Order By**: This operator is used to sort items in the subset by a specific attribute.

- **Distinct**: This operator eliminates the duplicates in a bag and returns a set.

- **Collapse/Expand**: COLLAPSE, also referred in literature as coalesce, combines the tuples, which have the same non-temporal values and consecutive or overlapping validities, into one tuple with validity that is the union of the constituent validities.

  EXPAND expands a tuple into several tuples by splitting its validity into consecutive validities in a given scale. In our approach this scale can be following keywords: YEAR, MONTH, DAY. These operators can be combined with *IN period* to limit the range. In Figure B.1, EXPAND BY YEAR IN [2001,2004) returns the first three tuples.

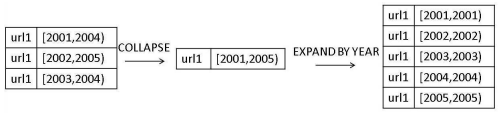


Figure B.1: Collapse/Expand

PUBLICATIONS, WORKSHOP, DEMOS C

- Zeynep Pehlivan, Benjamin Piwowarski, Stéphane Gançarski: Diversification Based Static Index Pruning - Application to Temporal Collections. CoRR abs/1308.4839 2013
- Zeynep Pehlivan, Benjamin Piwowarski, Stéphane Gançarski: A Comparison of Static Index Pruning Methods with Temporal Queries TAIA/SIGIR 2013
- Myriam Ben Saad, Zeynep Pehlivan, Stephane Gançarski: Coherence-oriented Crawling and Navigation for Web Archives using Patterns. In 15th International Conference on Theory and Practice of Digital Libraries, TPDL 2011
- Zeynep Pehlivan, Anne Doucet, Stéphane Gançarski Changing Vision for Access to Web Archives TWAW/WWW2011
- Zeynep Pehlivan, Myriam Ben Saad, Stéphane Gançarski: Vi-Diff: Understanding Web Pages Changes In the 21st International Conference on Database and Expert Systems Applications, DEXA 2010
- Myriam Ben Saad, Stéphane Gançarski, Zeynep Pehlivan A Novel Web Archiving Approach based on Visual Pages Analysis. In the 9th International Web Archiving Workshop (IWAW), Septembre 2009
- Myriam Ben Saad, Zeynep Pehlivan, Stéphane Gançarski : Crawling Et Navigation orientés cohérence pour les archives Web basés sur les patterns Dans 27ème Journées des Bases de Données Avancées , BDA 2011
- Myriam Ben Saad, Stéphane Gançarski, Zeynep Pehlivan Archiving Web Pages based on Visual Analysis and DIFF. Dans 25ème Journées des Bases de Données Avancées (BDA), Demonstration Poster, 2009

LIST OF FIGURES

List of Algorithms

BIBLIOGRAPHY

[ACMS02] Serge Abiteboul, Gregory Cobena, Julien Masanes, and Gerald Sedrati. A First Experience in Archiving the French Web. In *ECDL '02: Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, 2002.

[ADFW08] Eytan Adar, Mira Dontcheva, James Fogarty, and Daniel S Weld. Zoetrope: interacting with the ephemeral web. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, page 239–248, New York, NY, USA, 2008. ACM.

[AF94] James F. Allen and George Ferguson. Actions and events in interval temporal logic. Technical report, Rochester, NY, USA, 1994.

[AGB07] Omar Alonso, Michael Gertz, and Ricardo Baeza-Yates. On the value of temporal information in information retrieval. *SIGIR Forum*, 41:35–41, December 2007.

[AGHI09] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, page 5–14, New York, NY, USA, 2009. ACM.

[Ahl12] Dirk Ahlers. Local Web Search Examined. In Dirk Lewandowski, editor, *Web Search Engine Research*. Emerald, 2012. to appear.

[All83a] James F Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26:832–843, November 1983.

[All83b] James F Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26:832–843, November 1983.

[AM99] Gustavo O. Arocena and Alberto O. Mendelzon. Weboql: Restructuring documents, databases, and webs. *TAPOS*, 5(3):127–141, 1999.

[AOU09] Ismail Sengor Altingovde, Rifat Ozcan, and Ozgur Ulusoy. A practitioner's guide for static index pruning. In *ECIR'09*, 2009.

[ASBYG11] Omar Alonso, Jannik Strötgen, Ricardo Baeza-Yates, and Michael Gertz. Temporal Information Retrieval: Challenges and Opportunities. In *Proceedings of the 1st International Temporal Web Analytics Workshop (TWAW 2011)*, pages 1–8, 2011.

[AV08] Leif Azzopardi and Vishwa Vinay. Retrievability: an evaluation measure for higher order information access tasks. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 561–570, New York, NY, USA, 2008. ACM.

[BB07] Roi Blanco and Alvaro Barreiro. Static pruning of terms in inverted files. ECIR'07, Berlin, Heidelberg, 2007. Springer-Verlag.

[BB10a] Roi Blanco and Alvaro Barreiro. Probabilistic static pruning of inverted files. *ACM Trans. Inf. Syst.*, 2010.

[BB10b] Roi Blanco and Alvaro Barreiro. Probabilistic static pruning of inverted files. *ACM Trans. Inf. Syst.*, 28(1):1:1–1:33, January 2010.

[BBAW10] Klaus Berberich, Srikanta Bedathur, Omar Alonso, and Gerhard Weikum. A language modeling approach for temporal information needs. In *ECIR'10*. 2010.

[BC06] Stefan Büttcher and Charles L. A. Clarke. A document-centric approach to static index pruning in text retrieval systems. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, page 182–189, New York, NY, USA, 2006. ACM.

[BCF+08] Adam Brokes, Libor Coufal, Zuzana Flashkova, Julien Masanès, Johan Oomen, Radu Pop, Thomas Risse, and Hanneke Smulders. Requirement analysis report living web archive. Technical Report FP7-ICT-2007-1, 2008.

[BFG+09] Emmanuel Bruno, Nicolas Faessel, Hervé Glotin, Jacques Le Maitre, and Michel Scholl. Indexing by permeability in block structured web pages. In *Proceedings of the 9th ACM symposium on Document engineering*, DocEng '09, pages 70–73, New York, NY, USA, 2009. ACM.

[BFMS09] E. Bruno, N. Faessel, J. Le Maitre, and M. Scholl. Blockweb: An ir model for block structured web pages. In *Content-Based Multimedia Indexing, 2009. CBMI 2009. Seventh International Workshop on*, pages 219–224, 2009.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[BJKN10] Nikhil Bansal, Kamal Jain, Anna Kazeykina, and Joseph Naor. Approximation algorithms for diversified search ranking. In *Proceedings of the 37th international colloquium conference on Automata, languages and programming: Part II*, ICALP'10, pages 273–284, Berlin, Heidelberg, 2010. Springer-Verlag.

[Bla10] Karen Blakeman. Tracking changes to web page content, 2010.

[BMN03] Sourav S. Bhowmick, Sanjay K. Madria, and Wee K. Ng. *Web Data Management*. Springer, 1 edition, November 2003.

[BSG11] Myriam Ben Saad and Stéphane Gançarski. Archiving the Web using Page Changes Pattern: A Case Study. In *JCDL '11: Proceedings of ACM/IEEE Joint Conference on Digital Libraries*, Ottawa, Canada, 2011.

[BSGP09] Myriam Ben-Saad, Stéphane Gançarski, and Zeynep Pehlivan. A Novel Web Archiving Approach based on Visual Pages Analysis. In *9th International Web Archiving Workshop (IWAW'09)*, Corfu, Greece, 2009.

[CAH02] Grégory Cobéna, Talel Abdessalem, and Yassine Hinnach. A comparative study for xml change detection. Technical report, 2002.

[CAM02] G. Cobena, S. Abiteboul, and A. Marian. Detecting changes in XML documents. In *ICDE '02: Proceedings of 18th International Conference on Data Engineering*, 2002.

[Car11] Ben Carterette. An analysis of NP-completeness in novelty and diversity ranking. *Inf. Retr.*, 14(1):89–106, February 2011.

[Cat03] Warwick Cathro. Development of a digital services architecture at the national library of Australia. EduCause, 2003.

[CCF+01] David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoelle S. Maarek, and Aya Soffer. Static index pruning for information retrieval systems. SIGIR '01, New York, NY, USA, 2001. ACM.

[CG98] Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, page 335–336, New York, NY, USA, 1998. ACM.

[CGM97] Sudarshan S. Chawathe and Hector Garcia-Molina. Meaningful change detection in structured data. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 1997.

[CHWM04] Deng Cai, Xiaofei He, Ji-Rong Wen, and Wei-Ying Ma. Block-level link analysis. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 440–447, New York, NY, USA, 2004. ACM.

[CL13] Ruey-Cheng Chen and Chia-Jung Lee. An information-theoretic account of static index pruning. SIGIR '13, 2013.

[CLTH12] Ruey-Cheng Chen, Chia-Jung Lee, Chiung-Min Tsai, and Jieh Hsiang. Information preservation in static index pruning. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 2487–2490, New York, NY, USA, 2012. ACM.

[CLV04] D. Jevtuchova C. Lampos, M. Eirinaki and M. Vazirgiannis. Archiving the greek web. In *4th International Web Archiving Workshop (IWAW04)*, Bath, UK, 2004.

[CMM04] Constantinescu N. Cosulschi M. and Gabroveanu M. Classification and comparison of information structures from a web page. In *The Annals of the University of Craiova*, 2004.

[CNDZ08] YuJuan Cao, ZhenDong Niu, LiuLing Dai, and YuMing Zhao. Extraction of informative blocks from web pages. In *Proceedings of the 2008 International Conference on Advanced Language Processing and Web Information Technology*, pages 544–549, Washington, DC, USA, 2008. IEEE Computer Society.

[CRGMW96] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. Change detection in hierarchically structured information. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 1996.

[CYWM03a] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. VIPS: a vision-based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft Research, 2003.

[CYWM03b] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. VIPS: a Vision-based Page Segmentation Algorithm. Technical report, Microsoft Research, 2003.

[CYWM04] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Block-based web search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 456–463. ACM Press, 2004.

[CZ06] David Cossock and Tong Zhang. Subset ranking using regression. In *Proceedings of the 19th annual conference on Learning Theory*, COLT'06, Berlin, Heidelberg, 2006. Springer-Verlag.

[DGI08] Wisam Dakka, Luis Gravano, and Panagiotis G. Ipeirotis. Answering general time sensitive queries. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, page 1437–1438, New York, NY, USA, 2008. ACM.

[Dig11] A.G. Diggle, P.J.and Chetwynd. *Statistics and Scientific Method: an Introduction for Students and Researchers*. Oxford University Press, 2011.

[dSNS+09] Herbert Van de Sompel, Michael L. Nelson, Robert Sanderson, Lyudmila Balakireva, Scott Ainsworth, and Harihar Shankar. Memento: Time travel for the web. *CoRR*, abs/0911.1112, 2009.

[EDG+02] Milo Kova Evi, Michelangelo Diligenti, Marco Gori, Marco Maggini, and Veljko Milutinovi. Recognition of Common Areas in a Web Page Using Visual Information: a possible application in a page classification. In *the proceedings of 2002 IEEE International Conference on Data Mining ICDM'02*, 2002.

[Efr10] Miles Efron. Linear time series models for term weighting in information retrieval. *J. Am. Soc. Inf. Sci.*, 61(7):1299–1312, 2010.

[EG11] Miles Efron and Gene Golovchinsky. Estimation methods for ranking recent information. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 495–504, New York, NY, USA, 2011. ACM.

[FD81] David Freedman and Persi Diaconis. On the histogram as a density estimator:L2 theory. *Probability Theory and Related Fields*, 57(4):453–476, December 1981.

[FLL08] Trevor Fenner, Mark Levene, and George Loizou. Modelling the navigation potential of a web page. *Theor. Comput. Sci.*, 396(1-3):88–96, May 2008.

[FR98] Chris Fraley and Adrian E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *Comput. J.*, 41(8):578–588, 1998.

[GCMC02] Xiao-Dong Gu, Jinlin Chen, Wei-Ying Ma, and Guo-Liang Chen. Visual Based Content Understanding towards Web Adaptation. In *Second International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH2002)*, 2002.

[Gio09] Toni Giorgino. Computing and visualizing dynamic time warping alignments in r: The dtw package. *Journal of Statistical Software*, 31(7):1–24, 8 2009.

[Gla07] Henry M. Gladney. *Preserving digital information*. Springer, 2007.

[GMC11] Daniel Gomes, João Miranda, and Miguel Costa. A survey on web archiving initiatives. In *Proceedings of the 15th international conference on Theory and practice of digital libraries: research and advanced technology for digital libraries*, TPDL'11, page 408–420, Berlin, Heidelberg, 2011. Springer-Verlag.

[GSS06] Daniel Gomes, André L. Santos, and Mário J. Silva. Managing duplicates in a web archive. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, 2006.

[HY11] Helen Hockx-Yu. The past issue of the web. In *Proceedings of 3rd International Conference on Web Science*, ACM Web Science 2011, 2011.

[IA96] IA. Internet archive, 1996.

[IIP03] IIPC. International internet preservation consortium, 2003.

[IIP06] IIPC. Use cases for access to internet archives. Use case report, 2006.

[JD07] Rosie Jones and Fernando Diaz. Temporal profiles of queries. *ACM Trans. Inf. Syst.*, 25(3), July 2007.

[JKN+06] Adam Jatowt, Yukiko Kawai, Satoshi Nakamura, Yutaka Kidawara, and Katsumi Tanaka. A browser for browsing the past web. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 877–878, New York, NY, USA, 2006.

[Kan09] Nattiya Kanhabua. Exploiting temporal information in retrieval of archived documents. In *SIGIR*, page 848, 2009.

[KDG+02] Milos Kovacevic, Michelangelo Diligenti, Marco Gori, Marco Maggini, and Veljko Milutinovic. Recognition of common areas in a web page using visual information: a possible application in a page classification. In *in the proceedings of 2002 IEEE International Conference on Data Mining (ICDM'02*, page 250. IEEE Computer Society, 2002.

[Kin08] Andrew B. King. *Website optimization*. O'Reilly, 2008.

[KN10] Nattiya Kanhabua and Kjetil Nørvåg. Determining time of queries for re-ranking search results. In *ECDL '10*, pages 261–272, 2010.

[KN11] Nattiya Kanhabua and Kjetil Nørvåg. A comparison of time-aware ranking methods. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 1257–1258, New York, NY, USA, 2011. ACM.

[KRH08] Dirk Kukulenz, Christoph Reinke, and Nils Hoeller. Web contents tracking by learning of page grammars. In *ICIW '08: Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services*, pages 416–425, Washington, DC, USA, 2008. IEEE Computer Society.

[LC03] Xiaoyan Li and W. Bruce Croft. Time-based language models. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 469–475, New York, NY, USA, 2003. ACM.

[Lev11] Mark Levene. *An Introduction to Search Engines and Web Navigation*. John Wiley & Sons, January 2011.

[LF01] Robin La-Fontaine. A Delta Format for XML: Identifying Changes in XML Files and Representing the Changes in XML. In *XML Europe*, 2001.

[LKLC03] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, DMKD '03, pages 2–11, New York, NY, USA, 2003. ACM.

[LL01] Mark Levene and George Loizou. Web interaction and the navigation problem in hypertext written for encyclopedia of microcomputers, 2001.

[LPT00] Ling Liu, Calton Pu, and Wei Tang. Webcq - detecting and delivering information changes on the web. In *In Proc. Int. Conf. on Information and Knowledge Management (CIKM*, pages 512–519. ACM Press, 2000.

[Mas06] Julien Masanés. *Web Archiving*. Springer Berlin Heidelberg, 2006.

[MMM97] Alberto O. Mendelzon, George A. Mihaila, and Tova Milo. Querying the world wide web. *Int. J. on Digital Libraries*, 1(1):54–67, 1997.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[MSA+08] Edleno Silva de Moura, Celia Francisca dos Santos, Bruno Dos santos de Araujo, Altigran Soares da Silva, Pavel Calado, and Mario A. Nascimento. Locality-based pruning methods for web search. *ACM Trans. Inf. Syst.*, 26(2):9:1–9:28, April 2008.

[NCO04] Alexandros Ntoulas, Junghoo Cho, and Christopher Olston. What's new on the web?: the evolution of the web from a search engine perspective. In *Proceedings of the 13th international conference on World Wide Web*, WWW, pages 1–12, New York, NY, USA, 2004. ACM.

[Nut03] NutchWAX. Iipc, 2003.

[NWF78] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions–i. *Mathematical Programming*, 14(1):265–294, 1978.

[OS11] Marilena Oita and Pierre Senellart. Deriving dynamics of web pages: A survey. In *TWAW*, pages 25–32, 2011.

[Pan99] Pandora. Australia's web archive, 1999.

[Pet05] Luuk Peters. Change Detection in XML Trees: a Survey. In *3rd Twente Student Conference on IT*, 2005.

[PMP12] Andrea Cannata Carmelo Cassisi Placido Montalto, Marco Aliotta and Alfredo Pulvirenti. *Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining*. 2012.

[RGM03] Sriram Raghavan and Hector Garcia-Molina. Complex queries over web repositories. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '2003, pages 33–44. VLDB Endowment, 2003.

[RLG+10] ChotiratAnn Ratanamahatana, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh, Michail Vlachos, and Gautam Das. Mining Time Series Data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, page 1049–1077. Springer US, 2010.

[Rob77] S. E. Robertson. The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.

[RvB07a] Marcel Ras and Sara van Bussel. WEB ARCHIVING user survey. Technical report, National Library of the Netherlands (Koninklijke Bibliotheek), Netherlands, 2007.

[RvB07b] Marcel Ras and Sara van Bussel. WEB ARCHIVING user survey. Technical report, National Library of the Netherlands (Koninklijke Bibliotheek), Netherlands, 2007.

[RvRP80] Stephen E. Robertson, C. J. van Rijsbergen, and Martin F. Porter. Probabilistic models of indexing and searching. In *SIGIR*, pages 35–56, 1980.

[Sco79] D. W. Scott. On optimal and data-based histograms. *Biometrika*, pages 605–610, 1979.

[SDM+09] Marc Spaniol, Dimitar Denev, Arturas Mazeika, Gerhard Weikum, and Pierre Senellart. Data quality in web archiving. In *Proceedings of the 3rd workshop on Information credibility on the web*, WICOW '09, pages 19–26, New York, NY, USA, 2009. ACM.

[SH09] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, July 2009.

[SJ09] Sangchul Song and Joseph JaJa. Search and Access Strategies for Web Archives. In *Proceedings of IST Archiving 2009*, 2009.

[SKB+12] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 139–146, New York, NY, USA, 2012. ACM.

[Son04] Ruihua Song. Learning block importance models for web pages. In *In Intl. World Wide Web Conf. (WWW*, page 203–211. ACM Press, 2004.

[SPG11] Myriam Ben Saad, Zeynep Pehlivan, and Stéphane Gançarski. Coherence-oriented crawling and navigation using patterns for web archives. In *TPDL*, pages 421–433, 2011.

[SYY75] G. Salton, C. S. Yang, and C. T. Yu. A theory of term importance in automatic text analysis. *Journal of the American society for Information Science*, 26(1):33–44, 1975.

[TC11] Sree Lekha Thota and Ben Carterette. Within-document term-based index pruning with statistical hypothesis testing. In *Proceedings of the 33rd European conference on Advances in information retrieval*, ECIR'11, pages 543–554, Berlin, Heidelberg, 2011. Springer-Verlag.

[TDLH09] Jaime Teevan, Susan T. Dumais, Daniel J. Liebling, and Richard L. Hughes. Changing how people view changes on the web. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 237–246, New York, NY, USA, 2009. ACM.

[Tof07] Brad Tofel. Wayback for accessing web archives. In *IWAW 07*, Vancouver, British Colombia, Canada, 2007.

[TRE04] TREC. Trec 2004 robust track guidelines, June 2004.

[UKW96] UKWAC. Uk web archive, 1996.

[vR79] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[W3C99] W3C. Html protocol, http://www.w3.org/protocols/rfc2616/rfc2616-sec14.html, 1999.

[WCO11] Michael J. Welch, Junghoo Cho, and Christopher Olston. Search result diversity for informational queries. In *Proceedings of the 20th international conference on World wide web*, WWW '11, page 237–246, New York, NY, USA, 2011. ACM.

[WDC03] Y. Wang, D.J. DeWitt, and J.-Y. Cai. X-Diff: an effective change detection algorithm for XML documents. In *ICDE '03: Proceedings of 19th International Conference on Data Engineering*, March 2003.

[WNS+11] G. Weikum, N. Ntarmos, M. Spaniol, P. Triantafillou, A. Benczúr, S. Kirkpatrick, P. Rigaux, and M. Williamson. Longitudinal analytics on web archive data: It's about time! In *Proceedings of the 5th biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA, January 9-12*, pages 199–202, 2011.

[WO05] Charles A. Wood and Terence T. Ow. WEBVIEW: an SQL extension for joining corporate data to data derived from the web. *Commun. ACM*, 48(9):99–104, September 2005.

[WZ09] Jun Wang and Jianhan Zhu. Portfolio theory of information retrieval. In *In SIGIR '09: Proc. 32nd Int. ACM SIGIR Conf. on Research and Development in IR*, pages 115–122. ACM, 2009.

[XLL+08] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 107–114, New York, NY, USA, 2008. ACM.

[ZC09] L. Zheng and I. J. Cox. Entropy-based static index pruning. http://eprints.ucl.ac.uk/135592/, 2009.

[ZCL03] Cheng Xiang Zhai, William W. Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 10–17, New York, NY, USA, 2003. ACM.

[ZS89] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, December 1989.

[ZTY+05] Yanchun Zhang, Katsumi Tanaka, Jeffrey Xu Yu, Shan Wang, Minglu Li, Shengping Li, Shen Huang, Gui-Rong Xue, and Yong Yu. Block-Based language modeling approach towards web search. In *Web Technologies Research and Development - APWeb 2005*, volume 3399 of *Lecture Notes in Computer Science*, pages 170–182. Springer Berlin / Heidelberg, 2005.