UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Web Modelling for Web Warehouse Design

Daniel Coelho Gomes

DOUTORAMENTO EM INFORMÁTICA
ESPECIALIDADE ENGENHARIA INFORMÁTICA

2006

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Web Modelling for Web Warehouse Design

Daniel Coelho Gomes

DOUTORAMENTO EM INFORMÁTICA
ESPECIALIDADE ENGENHARIA INFORMÁTICA

2006

Tese orientada pelo Prof. Doutor Mário Jorge Costa Gaspar da Silva

Abstract

Users require applications to help them obtaining knowledge from the web. However, the specific characteristics of web data make it difficult to create these applications. One possible solution to facilitate this task is to extract information from the web, transform and load it to a Web Warehouse, which provides uniform access methods for automatic processing of the data. Web Warehousing is conceptually similar to Data Warehousing approaches used to integrate relational information from databases. However, the structure of the web is very dynamic and cannot be controlled by the Warehouse designers. Web models frequently do not reflect the current state of the web. Thus, Web Warehouses must be redesigned at a late stage of development. These changes have high costs and may jeopardize entire projects. This thesis addresses the problem of modelling the web and its influence in the design of Web Warehouses. A model of a web portion was derived and based on it, a Web Warehouse prototype was designed. The prototype was validated in several real-usage scenarios. The obtained results show that web modelling is a fundamental step of the web data integration process.

Keywords: Web warehousing, crawling, web characterization.

Resumo

Os utilizadores da web recorrem a ferramentas que os ajudem a satisfazer as suas necessidades de informação. Contudo, as características específicas dos conteúdos provenientes da web dificultam o desenvolvimento destas aplicações. Uma aproximação possível para a resolução deste problema é a integração de dados provenientes da web num Armazém de Dados Web que, por sua vez, disponibilize métodos de acesso uniformes e facilitem o processamento automático. Um Armazém de Dados Web é conceptualmente semelhante a um Armazém de Dados de negócio. No entanto, a estrutura da informação a carregar, a web, não pode ser controlada ou facilmente modelada pelos analistas. Os modelos da web existentes não são tipicamente representativos do seu estado presente. Como consequência, os Armazéns de Dados Web sofrem frequentemente alterações profundas no seu desenho quando já se encontram numa fase avançada de desenvolvimento. Estas mudanças têm custos elevados e podem pôr em causa a viabilidade de todo um projecto. Esta tese estuda o problema da modelação da web e a sua influência no desenho de Armazéns de Dados Web. Para este efeito, foi extraído um modelo de uma porção da web, e com base nele, desenhado um protótipo de um Armazém de Dados Web. Este protótipo foi validado através da sua utilização em vários contextos distintos. Os resultados obtidos mostram que a modelação da web deve ser considerada no processo de integração de dados da web.

Palavras Chave: Armazenamento de Dados Web, Recolha de Dados da Web, Caracterização da Web.

Resumo Alargado

A web ainda é vista principalmente como um meio de publicação destinado a ser interpretado por humanos. Esta perspectiva é muito limitativa face às suas potencialidades. A criação de aplicações que processem automaticamente dados da web para extracção de conhecimento é por outro lado possível e torna-se por isso necessária. Estas aplicações podem ter finalidades distintas, como a criação de índices de páginas web, estudos da linguagem natural, criação de arquivos históricos ou angariação de estatísticas. No entanto, se construídas de raiz, todas estas aplicações se irão debater com problemas comuns:

- A informação disponível na web é vasta e encontra-se dispersa, o que dificulta a localização de dados relevantes;
- A heterogeneidade de formatos de publicação e o desrespeito pelas especificações constituem um entrave à interpretação automática;
- A volatilidade da informação faz com que o seu acesso seja pouco fiável.

A integração de dados provenientes da web em Armazéns de Dados Web (*Web Warehouses*), que disponibilizem métodos de acesso uniformes destinados ao processamento automático, facilita a criação de novas ferramentas. O processo de integração de dados da web é conceptualmente semelhante ao de integração de dados relacionais. No entanto, a abordagem tradicional de integração de dados usada nos sistemas de Armazenamento de Dados (*Data Warehousing*) tem-se revelado inadequada no contexto da web, uma vez que os pressupostos no desenho deste tipo de sistemas não são aplicáveis. A arquitectura de um sistema de Armazenamento de Dados de negócio pressupõe que:

- As características das fontes de informação são bem conhecidas à partida;
- Se destina a apoiar sistemas de suporte à decisão que actuem sobre informação de pequena granularidade, tipicamente relacional;

- O processo de integração é feito em fases independentes, sendo a fase de recolha de informação pouco complexa.

Em contraposição, o desenho da arquitectura de um Armazém de Dados Web deverá pressupor que:

- As fontes de informação não são bem conhecidas;
- Destina-se a suportar sistemas de recuperação de informação hipertextual;
- O processo de integração impõe uma estrita cooperação entre as diferentes fases, sendo a recolha de informação dificultada por problemas de acessibilidade à informação.

Os Armazéns de Dados Web são carregados com dados de uma determinada porção da web. Vários estudos revelaram que cada porção da web apresenta as suas características peculiares. Assim sendo, é importante delimitar estas porções e modelá-las de modo a que um Armazém de Dados Web possa ser desenhado considerando as características da informação que irá processar. O processo de integração de dados da web compreende as fases de modelação da fonte de informação, recolha, transformação, armazenamento e acesso. Num Armazém de Dados Web cada uma delas coloca novos desafios:

Modelação. A web não é uma fonte de informação uniforme, pelo que será interessante integrar apenas uma parte da informação que disponibiliza. Impõe-se assim, a definição de critérios de selecção que permitam definir porções de informação. Após a definição da fronteira de uma porção da web, esta deverá ser modelada para permitir o dimensionamento do sistema e definição de abordagens. No entanto, esta modelação é normalmente dificultada pela ausência de estatísticas e caracterizações. Surge assim o interesse em metodologias que permitam efectuar a modelação de partições da web de forma sistemática;

Recolha e transformação. A tradução do critério de selecção numa política de recolha afirma-se como o primeiro desafio desta fase. O processo de recolha e transformação de dados deverá ser eficiente e ao mesmo tempo robusto.

A vastidão e diversidade da web impossibilitam a previsão e teste de todas as situações que poderão vir a ser encontradas;

Armazenamento e acesso. O armazenamento de dados requer estruturas de dados diferentes das usadas na fase de carregamento, de modo a proporcionar um modelo de acesso eficiente e uniforme. O grande volume de informação impõe que esta esteja acessível a pessoas e máquinas, pelo que os mecanismos de acesso deverão suportar processamento paralelo e gestão escalável da informação. A dimensão temporal é importante porque permite a análise histórica de dados.

As abordagens tomadas para a solução destes problemas têm sido feitas de forma disjunta e dentro de contextos específicos, não apresentando uma solução completa para o problema da integração de informação proveniente da web.

Esta tese foca o problema da modelação da web e a sua influência no desenho de Armazéns de Dados Web. A metodologia adoptada foi principalmente experimental. Foi extraído um modelo de uma porção da web e com base nele, foi desenhado um protótipo de um Armazém de Dados Web, denominado Webhouse. Cada um dos componentes deste sistema trata a resolução dos problemas de uma das etapas do processo.

A web portuguesa, definida como o conjunto de documentos de interesse cultural e sociológico para os portugueses, foi usada como caso de estudo. Esta porção da web foi recolhida e alvo de modelação em 2003. A maioria dos sítios da web portuguesa estavam alojados sob o domínio .PT e o número de sítios em construção era elevado. O uso de meta-dados adequados e descritivos era baixo.

O Webhouse é constituído por dois componentes principais: o repositório Versus e o batedor Viúva Negra (VN). O Versus gere os meta-dados e constitui o núcleo do sistema. Este componente fornece estruturas de dados e mecanismos para processamento paralelo, sincronização, gestão de faltas e suporte temporal. O VN foi desenhado como um cliente do Versus que efectua a recolha e carregamento em paralelo de dados provenientes da web.

Os componentes do Webhouse foram avaliados separadamente através de várias experiências. Os resultados obtidos mostraram que o VN é robusto e escalável. Por sua vez, o Gestor de Conteúdos do Versus suplantou significativamente o

desempenho do NFS, mostrando que o algoritmo proposto para a eliminação de duplicados melhora o débito de armazenamento, ao mesmo tempo que poupa espaço em disco. O Webhouse, como sistema completo, foi avaliado através da sua utilização em vários contextos distintos. A construção deste protótipo contribuiu para a experimentação científica e tecnológica de novas hipóteses no campo da recuperação de informação da web, permitindo efectuar simulações realistas em ambiente controlado.

Os resultados obtidos mostraram que a modelação deve ser considerada no processo de integração de dados da web. Este trabalho contribuiu para encontrar respostas às seguintes questões de investigação:

- Quais as características que deverão ser consideradas num modelo da web?

  A caracterização de sítios, conteúdos e estrutura de ligações de uma porção da web é crucial para desenhar um Armazém de Dados Web que a processe. Algumas características derivadas de análises da web global poderão não ser representativas de porções mais pequenas, como por exemplo, as webs nacionais. Contudo, as webs nacionais podem ser de grande interesse para grandes comunidades de utilizadores. Além disso, a caracterização de uma porção da web relativamente pequena é acessível em termos de recursos necessários e pode ser feita com grande precisão. Algumas características da web só podem ser modeladas a partir de diversas amostras recolhidas ao longo do tempo, como por exemplo a persistência de dados da web. Estas métricas devem ser incluídas num modelo da web porque permitem inferir tendências de evolução que têm uma forte influência no desenho de Armazéns de Dados Web que guardem colecções web construídas incrementalmente;

- Como podem ser definidas as fronteiras de uma porção da web?

  Uma porção da web poderá ser delimitada através de um conjunto de critérios de selecção. A porção da web deverá conter a informação necessária para satisfazer as necessidades das aplicações que a irão processar. Os critérios de selecção deverão ser facilmente concretizáveis como políticas de recolha automática. O recurso a algoritmos de classificação de conteúdos

e a restrição das recolhas a sítios web alojados em determinados domínios são opções que se revelaram adequadas;

- O que pode influenciar um modelo da web?

  A metodologia usada para recolher amostras da web influencia os modelos obtidos. A recolha automática de dados da web é um método de amostragem adequado ao contexto dos Armazéns de Dados Web, uma vez que emula o processo de extracção de dados normalmente usado nestes sistemas. Contudo, a configuração e tecnologia usadas nos sistemas de recolha de dados da web, e a existência de situações prejudiciais ao processamento automático de dados da web, poderão influenciar os modelos obtidos. Por isso, a interpretação de estatísticas obtidas de uma porção da web requer conhecimentos adicionais que extravasam uma análise matemática pura. São também necessários conhecimentos tecnológicos e sociais acerca da comunidade responsável pela criação dos conteúdos que compõem a porção da web;

- Qual é o grau de persistência da informação disponível na web?

  Durante esta investigação foram recolhidas amostras da web portuguesa durante três anos, a fim de modelar a persistência dos seus URL e conteúdos. Verificou-se que estas duas métricas podem ser modeladas através de distribuições logarítmicas. A maioria dos URL têm tempos de vida curtos e a taxa de mortalidade é mais elevada nos primeiros meses de vida. Existe porém uma pequena percentagem de URL que persiste durante vários anos. Estimou-se que passados dois meses, 50% dos URL de uma colecção web tenham morrido. As principais causas de morte detectadas foram a substituição de URL e a desactivação de sítios web. Os URL persistentes tendem a ser estáticos, curtos e a receberem referências de páginas alojadas noutros sítios web. Por sua vez, os sítios web apresentam tempos de vida mais longos do que os URL. Estimou-se que levariam 556 dias até que a taxa de mortalidade dos sítios web de uma colecção atingisse os 50%.

  Os modelos obtidos para a persistência de conteúdos sugerem que passados dois dias, 50% dos conteúdos de uma colecção web sofrem alterações. Com-

parando este resultado com outros obtidos em estudos anteriores, conclui-se que o tempo de vida dos conteúdos tende a diminuir. Verificou-se que cerca de metade dos conteúdos persistentes referenciam um conteúdo que permanece inalterado durante a sua vida;

- Qual a influência das características da web no desenho de Armazéns de Dados Web?

  Os modelos da web são ferramentas de suporte à decisão muito importantes no desenho de Armazéns de Dados Web que permitem fazer suposições realistas nas fases iniciais dos projectos. A duplicação de conteúdos é frequente na web. No entanto, é difícil evitar a recolha de conteúdos duplicados, uma vez que, estes são frequentemente referenciados por URL distintos e aparentemente não relacionados. Uma colecção de conteúdos web que tenha sido compilada incrementalmente apresenta um número adicional de duplicados causados pelos conteúdos que permanecem inalterados ao longo do tempo e são sucessivamente recolhidos e armazenados. Perante este cenário, a eliminação de duplicados ao nível de armazenamento é uma funcionalidade apelativa em Armazéns de Dados Web. No entanto, os mecanismos adoptados para suportar a eliminação de duplicados deverão ter em consideração a precariedade dos URL, pois poderão pôr em causa a implementação de algoritmos baseados em análises históricas.

  Um modelo para a previsão do tempo de vida dos URL permite a escolha de estruturas de dados e algoritmos adequados ao seu processamento num Armazém de Dados Web. Por outro lado, estes Armazéns necessitam de actualizar periodicamente a informação que detêm. Um modelo do tempo de vida dos conteúdos disponíveis na web permite medir a actualidade de uma colecção de dados da web armazenada e agendar operações para o seu refrescamento. Os formatos de publicação na web estão em permanente evolução, pelo que um Armazém de Dados Web deverá preservar os conteúdos detidos de modo a que permaneçam acessíveis após terem deixado de estar disponíveis na web. A extracção de informação é uma tarefa muito sensível porque o componente de software por ela responsável interage directamente

com a web e tem de enfrentar situações imprevistas que podem ser prejudicais ao seu bom funcionamento. Os modelos da web contribuem para o desenho de sistemas de recolha da web robustos a estas situações. Um Armazém de Dados Web deverá apresentar uma arquitectura distribuída que permita tratar de grandes quantidades de dados extraídos da web. As características da web influenciam a definição de estratégias de particionamento adoptadas para efectuar o equilíbrio de carga entre os processos que compõem um Armazém de Dados Web.

O desenho de Armazéns de Dados Web eficientes é portanto uma tarefa complexa que exige a combinação de conhecimentos em vários domínios. As principais contribuições enquadram-se principalmente em três áreas: a Caracterização da Web, que visa a sua monitorização e modelação; a Recolha de Dados Web, que estuda a recolha automática de dados da web; e o Armazenamento de Dados Web, que estuda o processo de integração de dados da web. As principais contribuições em cada um destes campos foram as seguintes:

Caracterização da Web:

- Uma caracterização detalhada da estrutura da web portuguesa (Gomes & Silva, 2005);
- Novos modelos que permitem estimar a persistência dos URL e conteúdos na web (Gomes & Silva, 2006a);
- Descrição detalhada de situações na web prejudiciais ao processamento automático dos seus dados.

Recolha de Dados Web:

- Um nova arquitectura distribuída para sistemas de recolha de dados da web, que se destaca pelas suas características de escalabilidade e robustez (Gomes & Silva, 2006b);
- Uma análise de técnicas de particionamento dos URL pelos processos que compõe um sistema de recolha de dados da web distribuído;

- Um estudo de técnicas que evitam a recolha de duplicados e URL inválidos de modo a poupar largura de banda e espaço em disco;

Armazenamento de Dados Web:

- Uma nova arquitectura para Armazéns de Dados Web que compreende todas as fases do processo de integração deste tipo de dados;
- Uma análise do impacto das características da web no desenho e desempenho de Armazéns de Dados Web;
- Um algoritmo para a eliminação de duplicados num sistema distribuído (Gomes *et al.*, 2006b).

O Webhouse também se revelou como uma ferramenta de suporte útil a outros trabalhos de investigação:

- Foi usado no estudo de: mecanismos de indexação e ordenação utilizados em motores de busca na web (Costa, 2004); semelhança entre documentos (Martins, 2004); identificação de língua em páginas web (Martins & Silva, 2005a); análise linguística de colecções de documentos web (Martins & Silva, 2004b); detecção e atribuição de âmbito geográfico a recursos na web (Silva *et al.*, 2006) e na execução de avaliações de sistemas de recuperação de informação multi-língue (Cardoso *et al.*, 2005b);
- É um dos principais componentes de um motor de busca (`www.tumba.pt`) e de um protótipo de um sistema de arquivo para web portuguesa (`tomba.tumba.pt`) (Gomes *et al.*, 2006a; Silva, 2003);
- Foi usado na criação do corpus de texto WPT03 para suporte à investigação na área de processamento da linguagem natural (disponível em `poloxldb.linguateca.pt`);
- Os componentes de software do Webhouse podem ser utilizados individualmente em diversos contextos que partilhem problemas com o Armazenamento de Dados Web. Por exemplo, o Gestor de Conteúdos do Versus foi utilizado como repositório de artigos científicos (Jul, 2005).

A criação de sistemas que permitam arquivar a informação publicada na web para fins históricos, à semelhança do que acontece com o depósito legal das publicações impressas, tem ganho uma importância crescente. A investigação apresentada nesta tese tem uma grande potencialidade de aplicação prática nestes sistemas. No entanto, o arquivo da web levanta novos problemas, designadamente de preservação dos dados, que seriam alvos de estudo interessantes para trabalhos futuros.

Acknowledgements

This is one giant leap for a man, one small step for mankind. A leap that would not be possible if he took it alone. I would like to thank:

- My princess Catarina for her love and understanding. A great part of this work is due to her, even though she basically hates computers. My little Madalena, you inspired me and gave me strength even before you were born;
- My advisor Mário J. Silva for his professionalism and unshakeable enthusiasm. He always finds time for his students and gives his best on helping them to improve. He is a role model for professors;
- André L. Santos, Bruno Martins and Sérgio Freitas for their participation in the design and development of the software used in my research. Francisco Couto, Nuno Cardoso, Marcirio Chaves and Paulo Sousa for their precious reviews;
- My friends João Campos, Miguel Costa and Norman Noronha for the encouragement they gave me when it was most needed;
- My parents, Antónia and Vitalino Gomes for educating me and giving me the chance of studying. My parents-in-law, Ju, Zé, Teresa and Luiz for treating me as a son. My grandparents Agostinho, Carminha, Daniel and Inácia for being examples that the greatness of people is not in their wealth or scholarship;
- My present and past colleagues at XLDB/LaSIGE: Ana Maria Afonso, Ana Paula Afonso, Andreas Wichert, Alysson, Feliciano Grosso, Henrique Moniz, João Antunes, Marcirio Chaves, Leonardo Andrade, Lili, Marquinho, Mocito, Nuno Maria and Rui Lopes for their support and interesting (or not) discussions;

- FCCN-Fundação para Computação Científica Nacional (FCCN), LaSIGE-Laboratório de Sistemas Informáticos de Grande Escala and Fundação para a Ciência e Tecnologia (scholarship grant SFRH/BD/11062/2002), for their financial support and Marktest LDA. for the access to the Netpanel statistics.

In memory of Ricardo "Adolpho Dias" Ribeiro and Beto Ribeiro.

Contents

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

The web is the largest source of information ever built. It provides a quick, cheap and simple publishing media. However, its full potential is far from being completely explored. Users require applications for aiding them in finding, summarizing and extracting useful knowledge from web data. However, the web was designed to provide information to be interpreted by humans and not automatically processed by software applications. The size and transience of web data make it difficult to design efficient systems able to harness its complexity in useful time and the heterogeneity and disrespect of standards make it difficult to automatically interpret the data. Thus, the automatic processing of web data is a challenging task.

One possible solution to address the above challenges is adopting a Data Warehousing approach. The idea is to extract information from the web, transform and load it to a system, called a Web Warehouse (WWh), which provides uniform access methods to facilitate the automatic processing of the data. Besides overcoming accessibility problems, Web Warehousing extends the lifetime of web contents and its reuse by different applications across time. Web warehousing has been widely used to support numerous applications. The integration of web data for offline automatic processing by web mining applications is a requirement of a wide range of systems. Web companies, such as Google (www.google.com) or Amazon A9 (www.a9.com), rely on Web Warehouses to support their businesses. The Internet Archive harvests and preserves web data for historical purposes, following a Web Warehousing approach (www.archive.org). Marketeers analyze web data

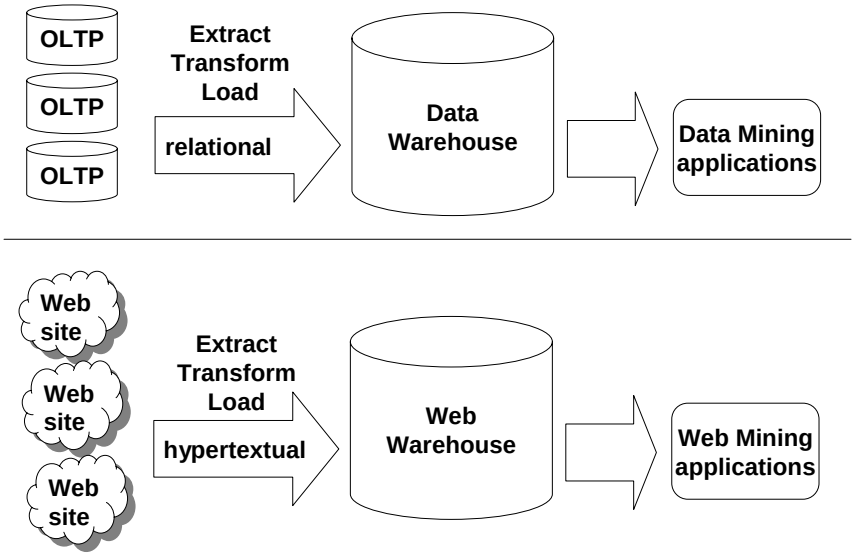


Figure 1.1: Data vs. Web Warehousing.

to determine commercial strategies (www.webanalyticsassociation.org). So, Web Warehousing is a research area that has received a growing interest in the last years.

Web Warehousing is conceptually similar to Data Warehousing. Figure 1.1 presents the data integration process in both. Data Warehouses integrate data gathered from tables in relational databases. The data is migrated from its source models into an uniform data model. Then, Data Mining applications generate statistical reports that summarize the knowledge contained in the data. Web Warehouses integrate hypertextual documents gathered from sites on the web. Web Warehouses also store information according to an uniform model that enables its automatic processing by Web Mining applications.

The characteristics of data influence the design of an information system, so the first step in the design of a Warehouse is to analyze the data sources. Data Warehousing assumes the existence of a well-defined model of the data sources. They are usually On-Line Transaction Processing (OLTP) databases that respect relational data models. On the other hand, the source of information that feeds Web Warehouses is the web and not relational databases. Unlike

relational databases, the structure of the web cannot be controlled by the people that design the WWh and does not follow a static structured data model. Models and characterizations of the web are scarce and frequently outdated, not reflecting its current state, making it difficult to make realistic assumptions in the design of a Web Warehouse. Frequently, Web Warehouses must be redesigned at a late stage of development because problems are detected only when the WWh leaves the experimental setup and begins to integrate information gathered from the real web. These changes have high costs and may jeopardize entire projects.

Models of the web can be used to aid in the design of efficient Web Warehouses. Besides, they are useful to tune other applications that process web data, such as proxies or crawlers. However, modelling the web is not straightforward. The web is enormous and permanently changing. So, the available models of the web may not be representative of the web portion used as data source of the WWh. Numerous characteristics of the web, such as the size of the contents or their formats, can be analyzed but it is hard to identify which ones will affect the design of the WWh. There are several methodologies used to gather samples of the web and derive models from them, such as the analysis of query logs, web crawls or web server traffic but the choice of the sampling methodology must be done carefully to reflect the characteristics of the web portion that will feed the WWh. Given the complexity and large size of the web, modelling it requires adequate tools and a significant amount of resources, such as storage space or bandwidth.

1.1 Objectives and methodology

This work addresses the problem of modelling the web and its influence on Web Warehousing as the main objective. This thesis aims to answer the following research questions:

- Which features should be considered in a model of the web?
- How can the boundaries of a portion of the web be defined?
- What can bias a web model?

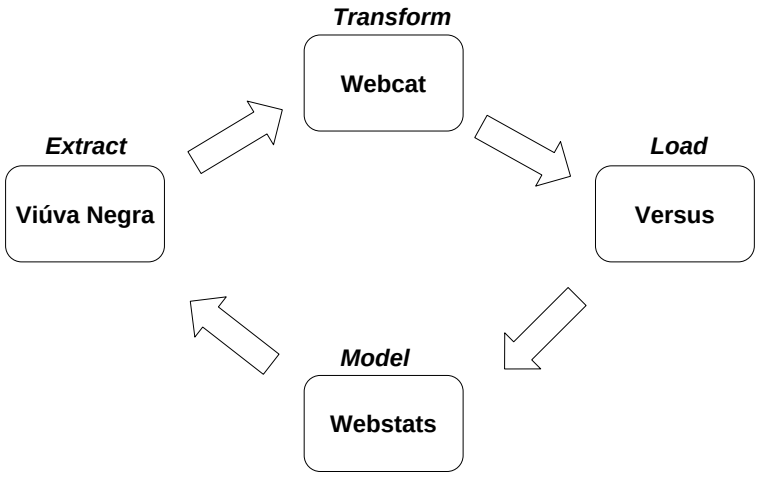


Figure 1.2: Components of Webhouse.

- How persistent is information on the web?
- How do web characteristics affect the design of Web Warehouses?

I believe that the task of modelling the web must be part of the process of web data integration, because accurate models are crucial in making important design decisions at an early WWh development stage. Web models also enable the tuning of a WWh to reflect the evolution of the web.

The methodology used in this research was mainly experimental. I derived a model of a portion of the web and, based on it, I developed Webhouse, a WWh for investigating the influence of web characteristics in Web Warehouses design. This development was performed in collaboration with other members of my research group.

Figure 1.2 presents an overview of the components of Webhouse. Each one addresses one stage of the integration process: modelling, extraction, transformation and loading. Although the integration process is decomposed in several steps, they are not independent from each other.

- Viúva Negra (VN): extracts information from the web by iteratively following the linked URLs embedded in web pages. These systems are broadly known as crawlers;

- WebCat: transforms web documents into an uniform data model (Martins & Silva, 2005b). This component was designed and developed by Bruno Martins;
- Versus: loads and stores web data;
- Webstats: models the web, generating statistics on the documents and correspondent meta-data stored in Versus.

The influence of web characteristics was studied during the design of each one of them. The extraction most sensitive stage of the integration process, because the software component interacts directly with the web and must address unpredictable situations. This thesis focuses mainly on the aspects of extracting information from the web and loading it into the WWh. The transformation of web data is not thoroughly discussed in this work. The efficiency of Webhouse as a complete system was validated through its application in several real usage scenarios.

This research was validated by applying the Engineering Method (Zelkowitz & Wallace, 1998). Several versions of Webhouse were iteratively developed and tested until the design could not be significantly improved. The Portuguese Web was chosen as a case study to analyze the impact of web characteristics in the design of a WWh. Models of the web were extracted through the analysis of the information integrated in the WWh. On its turn, a WWh requires models of the web to be designed. The Engineering Method enabled the identification of the web characteristics that influenced the performance of each version of the WWh and gradually improve it. So, although this thesis presents a sequential structure, the actual research was conducted as an iterative process.

1.2 Contributions

Designing Web Warehouses is complex and requires combining knowledge from different domains. This thesis provides contributions in multiple aspects of web data integration research:

Web Characterization: concerns the monitoring and modelling of the web;

Web Crawling: investigates the automatic extraction of contents from the web;

Web Warehousing: studies the integration of web data.

My specific contributions in each field are:

Web Characterization:

- A thorough characterization of the structural properties of the Portuguese Web (Gomes & Silva, 2005);
- New models for estimating URL and content persistence on the web. Despite the ephemeral nature of the web, there is persistent information and this thesis presents a characterization of it (Gomes & Silva, 2006a);
- A detailed description of hazardous situations on the web that make it difficult to automate the processing of web data.

Web Crawling:

- A novel architecture for a scalable, robust and distributed crawler (Gomes & Silva, 2006b);
- An analysis of techniques to partition the URL space among the processes of a distributed crawler;
- A study of bandwidth and storage saving techniques, that avoid the download of duplicates and invalid URLs.

Web Warehousing:

- A new architecture for a WWh that addresses all the stages of web data integration, from its extraction from the web to its processing by mining applications;
- An analysis of the impact of web characteristics in the design and performance of a Web Warehouse;
- An algorithm that eliminates duplicates at storage level in a distributed system (Gomes *et al.*, 2006b).

Webhouse was developed to answer the research questions of this thesis. However, it has also been an useful tool that supported other research studies:

- Webhouse was used to study indexing and ranking strategies for searching web documents (Costa, 2004), inter-document similarities (Martins, 2004), identification of language in web pages (Martins & Silva, 2005a), linguistic analysis of web corpora (Martins & Silva, 2004b), assignment of geographical scopes to web resources (Silva *et al.*, 2006) and to execute cross language evaluations (Cardoso *et al.*, 2005b);

- It is one of the main components of a search engine (`www.tumba.pt`) and a web archive (tomba.tumba.pt) for the Portuguese Web (Gomes *et al.*, 2006a; Silva, 2003);

- It was used to create a text corpus (WPT03), for Natural Language Processing research containing pages of the Portuguese Web, (available at `poloxldb.linguateca.pt`);

- The software components of Webhouse can be reused in different contexts that share problems with Web Warehousing. For instance, the Webhouse storage manager was used as a repository for scientific texts (Jul, 2005).

1.3 Structure of the thesis

This thesis focuses mainly on modelling the web and its influence on the extraction and loading of web data to a WWh. This chapter introduced the problem of modelling the web, described the adopted methodology and highlighted the contributions of this research. The thesis has six additional chapters.

Chapter 2 presents related work on web characterization, crawling and Web Warehousing projects. Chapters 3 and 4 present a model of the Portuguese Web, analyzing its structural properties and persistence of information across time. Chapter 5 describes the architecture of Webhouse and discusses the influence of the derived model of the Portuguese Web on the design of the system. Chapter 6 presents the Webhouse evaluation results. Finally, Chapter 7 draws the conclusions of this thesis and suggests directions for future research.

Chapter 2

Background and Related Work

The design of efficient Web Warehouses requires combining knowledge from Web characterization and Crawling. Web Characterization concerns the analysis of data samples to model characteristics of the web. Crawling studies the automatic harvesting of web data. Crawlers are frequently used to gather samples of web data in order to characterize it. Web warehouses are commonly populated with crawled data. Research in crawling contributes to optimizing the extraction stage of the web data integration process.

This chapter presents an overview of previous works related with Web characterization, Crawling and Web Warehousing. Section 2.1 presents web characterization concepts and studies. Section 2.2 presents an overview on the crawling process and discusses the design of several crawlers. Section 2.3 presents examples of state-of-the-art Web Warehousing projects and discusses the followed approaches. Finally, Section 2.4 draws the conclusions.

2.1 Web characterization

A characterization of the web is of great importance. It reflects technological and sociological aspects and enables the study of the web evolution. An accurate characterization of the web improves the design and performance of applications that use it as a source of information (Cho & Garcia-Molina, 2000a). This section introduces the terminology adopted to clarify web characterization concepts. It discusses sampling methodologies and the identification of contents belonging to

web communities. Finally, it presents previous works on the characterization of the structural properties and information persistence on the web.

2.1.1 Terminology

As the web evolves, new concepts emerge and existing terms gain new meanings. Studies in web characterization are meant to be used as historical documents that enable the analysis of the evolution of the web. However, there is not a standard terminology and the current meaning of the terms may become obscure in the future.

Between 1997 and 1999, the World-Wide Web Consortium (W3C) promoted the Web Characterization Activity with the purpose of defining and implementing mechanisms to support web characterization initiatives (W3C, 1999a). The scope of this activity was to characterize the web as a general distributed system, not focusing on specific users or sites. In 1999, the W3C released a working draft defining a web characterization terminology (W3C, 1999b). The definitions used in this thesis were derived from that draft:

Content: file resulting from a successful HTTP download;

Media type: identification of the format of a content through a Multipurpose Internet Mail Extension (MIME) type (Freed & Borenstein, 1996a);

Meta-data: information that describes the content. Meta-data can be generated during the download of a content (e.g. time spent to be downloaded), gathered from HTTP header fields (e.g. date of last modification) or extracted from a content (e.g. HTML meta-tags);

Page: content with the media type text/html (Connolly & Masinter, 2000);

Home page: content identified by an URL where the file path component is empty or a '/' only;

Link: hypertextual reference from one content to another;

Site: collection of contents referenced by URLs that share the same host name (Fielding *et al.*, 1999);

Invalid URL: an URL that references a content that cannot be downloaded;

Web server: a machine connected to the Internet that provides access to contents through the HTTP protocol;

Duplicates: a set of contents that are bytewise equal;

Partial duplicates: a set of contents that replicate a part of a content;

Duplicate hosts (duphosts): sites with different names that simultaneously serve the same content (Henzinger, 2003);

Subsite: cluster of contents within a site, maintained by a different publisher than that of the parent site;

Virtual hosts: sites that have different names but are hosted on the same IP address and web server;

Publisher or author: entity responsible for publishing information on the web.

Some of the definitions originally proposed in the draft are controversial and had to be adapted to become more explicit. The W3C draft defined that a page was a collection of information, consisting of one or more web resources, intended to be rendered simultaneously, and identified by a single URL. According to this definition, it is confusing to determine the contents that should be considered as part of a page. For instance, consider an HTML content and its embedded images. This information is meant to be rendered simultaneously but the images are referenced by several URLs different from the URL of the HTML content. Researchers commonly describe their experimental data sets providing the number of pages (Cho & Garcia-Molina, 2000a; Fetterly *et al.*, 2003; Lavoie *et al.*, 1997). According to the W3C definition, a data set containing one million pages should include embedded images. However, most researchers considered that a page was a single HTML document.

A set of bytewise equal contents are duplicates. However, there are also similar contents that replicate a part of another content (partial duplicates). Defining a criterion that identifies contents as being similar enough to be considered the

same is highly subjective. If multiple contents only differ on the value of a visit counter that changes on every download, they could reasonably be considered the same. However, when the difference between them is only as short as a number on the date of a historical event, this small difference could be very significant.

The definition of site is one of the most controversial on web terminology. It is commonly accepted that a site is a collection of inter-related contents but it is not straightforward to define the boundaries of a site. O'Neill (1999) recommended that web characterization studies should consider a site as the set of pages located at the same IP address. However, web hosting services that support thousands of different and completely independent pages under the same IP address should not be considered a single site. The W3C draft considered that the contents of a site shared the same host name but they must be accessible through a link path from the site's home page. This definition excludes contents that did not receive a link from another page within the site, even if they are referenced by links from other sites. Moreover, this definition requires the analysis of all the links within a site to determine if a given content should be considered as part of it. The definition of site adopted in this thesis enables the identification of the pages that compose a site exclusively through the analysis of their URLs. I believe that this is the most adequate definition because domain registrations are cheap and publishers tend to group related information under different site names to increase its visibility (Thurow, 2002).

Duplicate hosts are sites with different names that simultaneously serve the same content. They were identified as the single largest source of duplicates on the web (Henzinger *et al.*, 2002). Duphosts are mainly caused by mirrors and host aliases. Mirrors are sites that replicate the same contents across several web servers to backup data, reduce the load on the original site or to be in closer to the users (Bharat *et al.*, 2000). Host aliases are alternative names to the same site, such as `www.acm.org` and `acm.org`. However, some duphosts present completely different names and frequently change their contents, making it difficult to automatically identify them.

In traditional media, the author is the writer of a text, while the publisher is the responsible for its distribution. On the web, the process of publishing is cheaper and simpler than on traditional media, so the authors are usually also

the publishers. Hence, the terms author or publisher are used indistinguishably in this thesis to refer to the people responsible for the creation of web contents.

2.1.2 Sampling methods and identification of community webs

Web characterizations are derived from samples of the web. Ideally, each sample would be instantly gathered to be a representative snapshot of the web. However, contents cannot be accessed immediately, because of the latency times of the Internet and web server responses. Hence, samples must be gathered within a limited time interval named the timespan of the sample. Structural properties of the web are derived from a snapshot of the web extracted within a short timespan (Heydon & Najork, 1999). On the other hand, researchers also harvest samples with a long timespan to study the evolution of the web (Fetterly *et al.*, 2003). There are two main sources of samples:

Traffic logs. The accesses to web contents through a given service are registered on log files. Traffic logs can be obtained from web proxies (Bent *et al.*, 2004; Mogul, 1999b), web servers (Arlitt & Williamson, 1997; Davison, 1999; Iyengar *et al.*, 1999; Marshak & Levy, 2003; Rosenstein, 2000), search engines (Beitzel *et al.*, 2004; Silverstein *et al.*, 1999), web clients (Cunha *et al.*, 1995; Gribble & Brewer, 1997) or gateways (Caceres *et al.*, 1998; Douglis *et al.*, 1997). The samples gathered from traffic logs are representative of the portion of the web accessed by the users of a given service and not of the general information available on the web;

Crawls. Contents are automatically harvested by a crawler to be characterized off-line. Search engines are responsible for the largest crawls of the web. However, these crawls are biased because search engines are mostly interested in gathering popular pages to present them as search results to their users (Cho *et al.*, 1998). Henzinger *et al.* (2000) showed that the usage of random walks combined with the visit ratio and the PageRank values of the pages visited can be used to gather unbiased samples of the web. O'Neill *et al.* (2003) sampled the web by getting a list of randomly generated IP

addresses and then attempting to connect to the default HTTP port (80) at each address to find and harvest sites. The main problem with this approach is that virtual hosts are not analyzed (OCLC, 2001).

The web is designed to break all the geographical barriers and make information universally available. However, a WWh cannot store all the information from the web. So, it gathers data from selected and well-defined web portions.

As the web is the product of multiple user groups, it is possible to identify portions within it containing the sites of interest to them. These are designated as community webs and can be defined as the set of documents that refer to a certain subject or are of interest to a community of users. The detection of a community web is not always obvious, even if methods for identifying its boundaries are available. If one is interested in a small and static set of contents, then enumerating all the URLs that compose the community web can be adequate. However, it becomes very expensive to maintain the list of URLs if it grows or changes frequently (Webb, 2000).

Communities are very cohesive in the sense that members of the community are more tightly coupled to each other than to non-members (Flake *et al.*, 2000). Hence, the link structure of the web can be used to identify them. However, there will be difficulties in identifying contents loosely interlinked, even if they refer to the same subject. For instance, the sites of several concurrent companies in the same business, will not likely link to each other. Web communities can be viewed as containing a core of central authoritative pages that provide content linked together by hub pages (Gibson *et al.*, 1998). They exhibit a hierarchical topic generalization that can be inferred directly from the link structure between the contents.

National webs, broadly defined as the set of pages of cultural and sociological interest to the people of a country, are community webs of great interest to a large number of users. Baeza-Yates *et al.* (2007a) analyzed web characterization studies for 24 national webs and compared the obtained results. There are similarities and differences between national webs. The language and the distribution of domain registrations are the most varying characteristic across the national domains studied.

To populate a national web archive, Albertsen (2003) harvested documents from the web published on sites with the Norwegian ccTLD (.NO) or written in the Norwegian or Sami languages. However, when a language is not exclusive to a single country, it may not be a selective criterion by itself. For instance, most of the pages written in the English language do not belong to the British community web.

There are two main classes of top-level domains (TLD): generic (gTLDs) and country code (ccTLDs). The gTLDs were meant to be used by particular classes of organizations (e.g. COM for commercial organizations). The ccTLDs are delegated to designated managers, who operate them according to local policies adapted to best meet the economic, cultural, linguistic, and legal circumstances of the country. Hence, sites with a domain name under a ccTLD are strong candidates to be part of a national community web. However, this approach excludes the numerous contents related to a country hosted outside the ccTLD (Zook, 2000). On the other hand, this rule also includes sites not related to the country, but being hosted under its ccTLD. For instance, multi-national companies commonly register their name under many domains to protect their brands from domain squatters that buy domain names desirable to specific businesses to make profit on their resale.

2.1.3 Structural analysis

A structural analysis of the web consists on studying the characteristics of its contents and the links among them. The web can be characterized according to numerous metrics, such as page size, number of contents hosted per site or most common content formats.

Baldi *et al.* (2003) proposed probabilistic methods and algorithms to be used in web modelling. They analyzed previous experimental measurements and theoretical analysis of the web to map mathematical concepts with web phenomena. In particular, they studied techniques for crawling, analyzing texts and links, and modelling human behaviors while browsing and using search engines.

Next, I will present a summary of structural web characterization studies and discuss the obtained results for some of the analyzed metrics.

Author	Methodology	Sample size	Scope
Woodruff *et al.* (1996)	Crawl	2.6M	General
Smith (1997)	Crawl	-	General
Broder *et al.* (1997)	Crawl	30M	General
Mogul (1999a)	Proxy trace	29.4M	Accessed by clients
Lawrence & Giles (1999)	Crawl	16M	General
Baeza-Yates & Castillo (2000)	Crawl	730 000	National: Chile
Najork & Heydon (2001)	Crawl	77.4M	General
Punpiti (2000)	Crawl	769 000	National: Thailand
Boldi *et al.* (2002a)	Crawl	2M	Continental: Africa
Kelly & Mogul (2002)	Client trace	40M	Accessed by clients
O'Neill *et al.* (2003)	Crawl	3.08M	General
Bent *et al.* (2004)	Web server logs	41M	Enterprise sites
Nanavati *et al.* (2004)	Crawl	1M	General
Baeza-Yates *et al.* (2005)	Crawl	16M	National: Spain
Baeza-Yates *et al.* (2007b)	Crawl	7M	National: Chile

Table 2.1: Web characterization studies.

2.1.3.1 Summary of web characterizations

Measurements of web characteristics obtained using distinct methodologies may not be comparable. Web characterizations may have different scopes. Some researchers try to generate general models of the web, while others focus on particular web communities to gather more representative models. Measurements gathered from very small samples of the web are more prone to be biased by sporadic phenomena, such as a mal-functioning web servers.

Table 2.1 presents the sampling methodology, the sample size and the scope of several web characterization studies.

Woodruff *et al.* (1996) examined the characteristics of pages collected by the Inktomi crawler from the world-wide web. They observed that 40% of the pages contained at least one syntax error. Eight years later, Nanavati *et al.* (2004) conducted a similar study and observed that most of the analyzed metrics presented significantly distinct values.

Lawrence & Giles (1999) studied the accessibility of information on the web and drew conclusions about the size, extracted text and usage of meta-data in

HTML pages. They observed that the major public search engines collectively covered about 60% of the web and that the largest coverage of a single engine was about one-third of the estimated total size of the web.

Najork & Heydon (2001) performed a large crawl of the web and gathered statistics on the outcome of download attempts, replication and distribution of content format and size. They witnessed that the distribution of pages over web servers followed a Zipfian distribution.

Bent *et al.* (2004) studied the properties of a large number of commercial sites hosted by a major ISP and performed a simulation analysis to estimate potential performance benefits of content delivery networks (CDNs). The authors observed that the web data presented peculiar characteristics that require mandatory cache validations and prevent caching contents to increase performance. For instance, the wide-spread indiscriminate usage of cookies and the prevalence of sites that do not use the cache-control features of the HTTP 1.1 protocol prevents the use of many content delivery optimizations.

Several authors studied the characteristics of web communities. Baeza-Yates & Castillo (2000) presented the first study of the characteristics of a national web in 2000. The authors studied the pages under the Chilean ccTLD domain (.CL) harvested for the Todo.CL search engine. Since then, subsequent studies provided updated information about the characteristics of the Chilean web (Baeza-Yates & Castillo, 2005; Baeza-Yates & Poblete, 2006; Baeza-Yates *et al.*, 2003). The most recent one studied a data set of 7 million pages and analyzed 36 metrics about domains, sites and pages (Baeza-Yates *et al.*, 2007b). Castillo (2004) compared the characteristics of the Chilean web with the Greek web.

Baeza-Yates *et al.* (2005) studied the contents, links and technologies of Spanish pages, sites and domains in 2004. Some of the characteristics of this collection resemble those of the web at large, while others are specific to the Spanish web.

Boldi *et al.* (2002a) studied the structural properties of the African web, analyzing HTTP header fields and contents of pages and Punpiti (2000) presented measurements and analysis of contents hosted under the .TH domain.

Study	Duplication
Broder *et al.* (1997)	18%
Najork & Heydon (2001)	8.5%
Mogul (1999a)	16.3%
Kelly & Mogul (2002)	5%
Castillo (2004)	6.12%

Table 2.2: Duplication comparison.

2.1.3.2 Links

One main difference between the texts published on the web and those published in traditional media is the existence of hypertextual links among them. Considering that a page is a node and a link is an edge, the web structure can be seen as a graph (Kumar *et al.*, 2000). The links tend to follow a power-law indegree distribution (Kleinberg, 1999). Thus, a small amount of pages receives most of the links from other pages. However, 91% of the pages are reachable from one another by following either forward or backward links (Broder *et al.*, 2000). The distribution of links between sites is related with to the TLD of the site (Bharat *et al.*, 2001). For instance, the sites with the same ccTLD are strongly connected. The characteristics of the web graph are particularly interesting to determine the popularity of pages. Based on the idea that the most cited publications are the most relevant ones, the link structure of the web can be used to identify popular pages and increase the relevance of search engine results (Page *et al.*, 1999).

2.1.3.3 Duplication

Some contents are exact duplicates of others available at different URLs. The existence of duplication on the web has been studied to syntactically cluster web pages (Broder *et al.*, 1997), identify near-replicas (Shivakumar & Garcia-Molina, 1999) and find sites with mirrored content (Bharat & Broder, 1999). In fact, a few sites are responsible for most of the duplicates (Douglis *et al.*, 1997; Mogul, 1999a). Table 2.2 presents a comparison of the duplication levels found in several web characterization studies. The results show that duplication is prevalent on the web.

Study	Page size (KB)	Site size (Nr. of pages)
Woodruff *et al.* (1996)	4.4	-
Smith (1997)	5	300
Broder *et al.* (1997)	5	-
Lawrence & Giles (1999)	18.7	289
Mogul (1999a)	9.5	-
Punpiti (2000)	9.8	90
Boldi *et al.* (2002a)	12.9	-
Kelly & Mogul (2002)	17	-
O'Neill *et al.* (2003)	10-20	441
Bent *et al.* (2004)	9	-
Nanavati *et al.* (2004)	22	-
Castillo (2004)	16	67
Baeza-Yates *et al.* (2005)	-	52

Table 2.3: Size comparison.

The detection of partial duplication has been studied to detected plagiarized documents yielding good accuracy results (Brin *et al.*, 1995; Finkel *et al.*, 2002; Shivakumar & Garcia-Molina, 1995). However, the presented methods are too demanding to be applied to large collections of documents.

2.1.3.4 Size

The storage capacity of a WWh must be dimensioned according to the size of the contents and sites that will feed it. However, the estimation of these parameters may not be straightforward, because studies executed approximately at the same time present contradictory results. For instance, Table 2.3 shows that Lawrence & Giles (1999) observed an average page size of 18.7 KB, while Mogul (1999a) obtained approximately half of that value. The main difference between the two studies was that the first one used crawled samples, while the former used a web proxy trace. On the other hand, Punpiti (2000) obtained an average page size close to Mogul (1999a), although they used crawled samples.

Author	Scope	First Lang.	Second Lang.
Punpiti (2000)	Thailand	English (66%)	Thai (<50%)
Boldi *et al.* (2002a)	Africa	English (74.6%)	French (7.33%)
O'Neill *et al.* (2003)	World	English (72%)	German (7%)
Castillo (2004)	Chile	Spanish (71%)	English (27%)
Baeza-Yates *et al.* (2005)	Spain	Spanish (52.4%)	English (30.7%)

Table 2.4: Language comparison.

2.1.3.5 Language

Language can be a determinant criterion to define the portion of the web that will feed a WWh. In the early days of the web, pages were almost exclusively written in English. Later, people began publishing contents in their native languages at a fast pace (Funredes, 2001; Grefenstette & Nioche, 2000). Table 2.4 presents a comparison of the dominant languages on different portions of the web. Nonetheless, from 1999 to 2002, English maintained a share of 72% of the pages publicly available on the wide-web (O'Neill *et al.*, 2003). Besides the historical background of the web, there are two main reasons for the dominance of this language. First, most sites originate from the United States (49% in 1999, 55% in 2002). Second, natural speakers of languages that are not widely known tend to write contents in English to increase the visibility of their pages (Punpiti, 2000).

2.1.4 Information persistence

Most web data is ephemeral. However, there is persistent information. The persistence of web data should be considered in the design and operation of WWh because it repeatedly collects and stores information from the web.

Capacity planning. A WWh is dimensioned based on the estimated amount of data to process. A WWh designed to harness billions of contents has different requirements than a smaller one. The identification and reuse of persistent data may reduce the storage requirements;

Author	Age	Content persistence
Brewington & Cybenko (2000)	100 days	50%
Cho & Garcia-Molina (2000a)	1 day	77%
Fetterly *et al.* (2003)	7 days	65%
Ntoulas *et al.* (2004)	1 year	10%

Table 2.5: Content persistence on the web.

Author	Age	URL persistence
Koehler (2002)	1.9 years	50%
Cho & Garcia-Molina (2000a)	1 month	70%
Fetterly *et al.* (2003)	2.8 months	88%
Ntoulas *et al.* (2004)	1 year	20%

Table 2.6: URL persistence on the web.

Scheduling maintenance operations. A WWh periodically refreshes its data and prunes stale information. These operations are costly and their scheduling should be optimized considering the persistence of web data.

A WWh that daily harvests a portion of the web could increase its performance by avoiding the redundant download of contents that have not changed since the last harvest. This could be achieved through the estimation of the contents frequency of change referenced by an URL based on its past changes (Cho & Garcia-Molina, 2003). Unfortunately, historical data for most web resources is not available. It is also difficult to gather enough historical data on the data referenced by a given URL because the lifetime of URLs is short. There are new URLs permanently being created while others disappear (Ntoulas *et al.*, 2004; Spinellis, 2003). The problem of URL persistence could be solved through the usage of Universal Resource Names (URN), registered and maintained by the holders of the Fully Qualified Domain Name registration (Daigle *et al.*, 2002). But so far, the existence of URNs is insignificant. National Libraries tried to maintain URNs to selected publications, but the human intervention required made URNs unbearable at web scale (Noronha *et al.*, 2001).

Author	Age	URL persistence
Spinellis (2003)	1 year	80%
Markwell & Brooks (2003)	4.7 years	50%
Lawrence *et al.* (2000)	1 year	77%

Table 2.7: URL persistence on digital libraries.

Tables 2.5 and 2.6 present the obtained results in previous works on the persistence of contents and URLs cited from pages. An URL is considered persistent if the referenced content was successfully downloaded in two samples, independently from content changes. A content is considered persistent if it was successfully downloaded in two samples independently from the URLs that referenced it. The age of an URL or a content is the time elapsed between the samples. For instance, in Table 2.5, Brewington & Cybenko (2000) gathered two samples of the web with an interval of 100 days and observed that 50% of the contents in the first sample were still available on the second one.

Koehler (2002) examined the accessibility of a collection of 361 URLs randomly selected from a crawl during 4 years and concluded that once a collection has sufficiently aged, it tends to stabilize. The author witnessed the periodic resurrection of pages and sites, sometimes after protracted periods of time. A reason for this situation is that site domains are resold and URLs resurrect referencing completely different and unrelated contents.

Cho & Garcia-Molina (2000a) studied the frequency of change of pages by harvesting a selection of 270 popular sites during 4 months on a daily basis, summing a total of 720 000 pages. They proposed estimators for the frequency of change of pages and counted how many days each URL was accessible to derive its lifespan.

Fetterly *et al.* (2003) studied the evolution of pages by executing weekly crawls of a set of 150 million URLs gathered from the Yahoo! home page (`www.yahoo.com`). The study spanned 11 weeks in 2002. The authors focused on identifying characteristics that may determine the frequency and degree of change of a page. They found that most changes consisted of minor modifications, often of markup tags.

Ntoulas *et al.* (2004) studied the evolution of contents and link structure of 150 top-ranked sites picked from the Google directory for one year. They witnessed high levels of birth and death of URLs and concluded that the creation of new pages is a much more frequent cause of change on the web than changes in existing pages.

Brewington & Cybenko (2000) studied the change rate of pages by recording the Last-Modified timestamp and the time of download of each page accessed by the users of a clipping service. This analysis ignored those pages not relevant to the users' standing queries and the pages were observed over an average of 37 days. The authors proposed a model to schedule web data refreshment operations.

The problem of URL persistence has been studied by the digital libraries community, motivated by the increasing number of citations to URLs on scientific publications (Lawrence *et al.*, 2000). Table 2.7 presents the obtained results on the persistence of URLs cited by articles in digital libraries. The methodology used was to extract citations to URLs from archive documents over several years and verify if they were still available. Changes to contents were not evaluated, because the digital libraries did not keep copies of the cited documents.

Spinellis (2003) visited URLs extracted from research articles gathered from the ACM and IEEE digital libraries. The author witnessed that one year after the publication of the research articles, 80% of the cited URLs were accessible, but this number decreased to 50% after four years.

Markwell & Brooks (2003) monitored 515 web pages from distance learning courses for 24 months. During this time, the authors witnessed that over 20% of the URLs became nonviable, moving without automatic forwarding or having their content changed.

Lawrence *et al.* (2000) analyzed the persistence of information on the web, looking at the percentage of invalid URLs contained in academic articles published since 1993 within the CiteSeer database and validated them in May, 2000. They studied the causes for the invalid URLs and proposed solutions for citing and generating URLs to improve citation persistence.

The results suggest that the URLs cited from articles kept in digital libraries tend to be more persistent than those cited from web pages.

2.2 Crawling

Designing a crawler to harvest a small set of well-defined URLs is simple. However, harvesting information spread across millions of pages requires adequate selection criteria and system architectures. Most crawlers adopt distributed architectures that enable parallel crawling to cope with the large size of the web.

Web warehouses use crawlers to extract data from the web. This section presents crawler types and their operation. Then, it discusses architectural options to design a crawler and strategies to divide the URL space among several crawling processes. Finally, it provides crawler examples and compares their design and performance.

2.2.1 Crawler types and functioning

Crawlers can be classified in four major classes according to their harvesting strategies:

Broad. Collect the largest amount of information possible within a limited time interval (Najork & Heydon, 2001);

Incremental. Revisit previously fetched pages, looking for changes (Edwards *et al.*, 2001);

Focused. Harvest information relevant to a specific topic, usually with the help of a classification algorithm, to filter irrelevant contents (Chakrabarti *et al.*, 1999);

Deep. Harvest information relevant to a specific topic but, unlike focused crawlers, have the capacity of filling forms in web pages and collect the returned pages (Ntoulas *et al.*, 2005; Raghavan & Garcia-Molina, 2001).

Although each type of crawler has specific requirements, they all present a similar functioning. A crawl of the web is bootstrapped with a list of URLs, called the *seeds*, which are the access nodes to the portion of the web to crawl. For instance, to crawl a portion of the web containing all the contents hosted in the .GOV domain, URLs from that domain should be used as seeds. Then, a

crawler iteratively extracts links to new URLs and collects their contents. The seeds should be carefully chosen to prevent the crawler from wasting resources visiting URLs that do not reference accessible or relevant contents. They can be gathered from different sources:

User submissions. The seeds are posted by the users of a given service. However, many of them are invalid because they were incorrectly typed or reference sites still under construction;

Previous crawls. The seeds are extracted from a previous crawl. The main problem of this source of seeds is that URLs have short lives and an old crawl could supply many invalid seeds;

Domain Name System listings. The seeds are generated from domain names. However, the domains reference servers on the Internet and some of them are not web servers. So, the generated seeds may not be valid. Another problem is that the lists of the top-level domains of the web portion to be crawled are usually not publicly available.

2.2.2 Requirements

There are several types of crawlers. Although each one has specific requirements, they all share ethical principles and address common problems. A crawler must be:

Polite. A crawler should not overload web servers. Ideally, the load imposed while crawling should be equivalent to that of a human while browsing. A crawler should expose the purposes of its actions and not impersonate a browser, so that webmasters can track and report inconvenient actions. A crawler must respect exclusion mechanisms and avoid visits to sites where it is not welcome. The Robots Exclusion Protocol (REP) makes the definition of access rules on a file named robots.txt that is automatically interpreted by crawlers (Koster, 1994). An author of an individual page can also indicate if it should be indexed and if the links should be followed by a crawler through the `ROBOTS` HTML meta-tag (The Web Robots Pages, 2005);

Robust. The publication of information on the web is uncontrolled. A crawler must be robust against hazardous situations that may affect its performance or cause its mal-functioning;

Fault tolerant. Even a small portion of the web is composed by a large number of contents, which may take several days to be harvested. Crawlers frequently present a distributed architecture comprising multiple components hosted on different machines. A crawler must be fault tolerant so that its performance may degrade gracefully if one of its components fails, without compromising the progress of the crawl on the remaining machines;

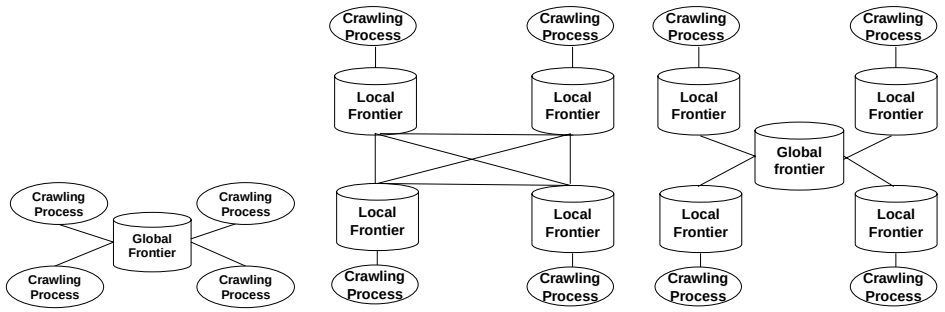
Able to collect meta-data. There is meta-data temporarily available only during the crawl (e.g. date of crawl). A crawler should keep these meta-data because it is often required by the WWh clients. For instance, the Content-Type HTTP header field identifies the media type of a content. If this meta-data element is lost, the content type must be guessed later;

Configurable. A crawler should be highly configurable to enable the harvesting of different portions of the web without suffering major changes;

Scalable. The crawl of a portion of the web must be completed within a limited time and the download rate of a crawler must be adequate to the requirements of the application that will process the harvested data. A WWh that requires weekly refreshments of data cannot use a crawler that takes months to harvest the required web data. The download rate of the crawler is always limited by the underlying resources, such as the number of machines. However, a crawler must be designed to scale its performance proportionally to available resources;

Economic. A crawler should be parsimonious with the use of external resources, such as bandwidth, because they are outside of its control. A crawler may to connect the Internet through a large bandwidth link but many of the visited web servers do not;

Manageable. A crawler must include management tools that enable the quick detection of its faults or failures. For instance, a hardware failure may



(a) Centralized frontier. (b) Distributed frontier. (c) Hybrid frontier.

Figure 2.1: Crawler architectures.

require human intervention. On the other hand, the actions of a crawler may be deemed unacceptable to some webmasters. So, it is important to keep track of the actions executed by the crawler for latter identification and correction of undesirable behaviors.

2.2.3 Architectural options

A crawler is composed by a *Frontier* that manages the URLs and the *Crawling Processes* that iteratively harvest contents from the web and store them locally. A Crawling Process (CP) iteratively gets a seed from the Frontier, downloads the referenced content, parses it, extracts the linked URLs and inserts them in the Frontier to be harvested. A crawl finishes when there are no seeds left to visit or a limit date is reached.

A simple crawler can be assembled from only one CP and one Frontier. This is what is known as an off-line browser. Simple crawlers like this are suitable for storing local copies of sites by individual users. However, the download rate provided by this architecture is not scalable. Large-scale crawlers parallelize crawling using several Crawling Processes at the cost of increasing its complexity. The URL space must be partitioned to enable parallel harvesting and the Crawling Processes must be synchronized to prevent multiple harvests of the same URLs.

The Frontier is the central data structure of a crawler. Some URLs are linked from many different pages. Thus, every time a CP extracts an URL from a link

embedded in a page, it must verify if the URL already existed in the Frontier to prevent overlapping. This verification is known as the *URL-seen test* and demands permanent access to the Frontier (Heydon & Najork, 1999). There are three approaches for organizing the Frontier:

Centralized. In this organization, the Crawling Processes share a single *Global Frontier* (see Figure 2.1(a)). The URL-seen test is permanently being executed on the Global Frontier by all the Crawling Processes. This architecture is conceptually simple but the Frontier becomes a potential hot-spot of the system;

Distributed. The Frontier is a cooperative set of *Local Frontiers* (see Figure 2.1(b)). There is not a central point of congestion because the URL-seen tests are distributed by the Local Frontiers. However, the URL-seen test imposes frequent synchronization among the Local Frontiers, which may become a bottleneck;

Hybrid. The Frontier is distributed among several Local Frontiers that periodically synchronize with a central Global Frontier (see Figure 2.1(c)). This approach does not concentrate the load of the URL-seen test on a single component. It does not either require frequent synchronization among the Local Frontiers. However, the design of the crawler is more complex than with previous approaches.

2.2.4 Web partitioning and assignment

A partitioning function maps an URL to its partition. The main objective of partitioning the URL space is to distribute the workload among the Crawling Processes, creating groups of URLs that can be harvested independently. After partitioning, each CP is responsible for harvesting exclusively one partition at a time. The partitioning strategy has implications on the operation of the crawler. In general, the following partitioning strategies may be considered:

IP partitioning. Each partition contains the URLs hosted on a given IP address;

Site partitioning. Each partition contains the URLs of a site. This partitioning schema differs from the above, because several sites may be hosted on the same IP address (virtual hosts) and each will be crawled separately;

Page partitioning. Each partition contains a fixed number of URLs independently from their physical location. A partition may contain URLs hosted on different sites and IP addresses. Page partitioning is suitable to harvest a selected set of independent pages spread on the web.

Initially, each partition contains only a set of seeds. A partition is assigned to a CP that becomes responsible for harvesting the correspondent URLs. The assignment process can be *static* or *dynamic* (Cho & Garcia-Molina, 2002).

In the static assignment, the partitions are assigned before the beginning of the crawl. Each CP knows its partition and assigns the URLs extracted from pages to the partitions responsible for their crawl. The static assignment imposes that the number of Crawling Processes is constant during the crawl to guarantee that all partitions are harvested. The partitions assigned to a CP would not be harvested if it failed and could not be recovered. Moreover, one cannot increase the number of Crawling Processes to accelerate a crawl, because all the partitions were mapped to the initial set of Crawling Processes.

In the dynamic assignment, a central coordinator assigns the partitions during the crawl. This approach supports having a variable number of Crawling Processes. The performance of the system degrades if a CP fails, but this does not compromise the coverage of the crawl. There are two strategies for dynamic assignment according to the behavior of the coordinator:

Push. The coordinator sends partitions to the Crawling Processes. It keeps an accurate state of the system to balance the load efficiently. The coordinator is responsible for monitoring the set of active Crawling Processes and contact them to assign partitions. So, it has the overhead of establishing connections, detecting and managing possible failures of the Crawling Processes. This approach enables the concentration of the crawler management on a single component which facilitates administration tasks;

Pull. The coordinator waits for requests of partitions to crawl by Crawling Processes. It does not have to permanently monitor the system because the Crawling Processes demand work on a need basis. The number of Crawling Processes may be variable without imposing any overhead on the coordinator because it simply responds to requests for uncrawled partitions.

2.2.5 Crawler examples

Web crawling has been receiving increasing interest from the research community since Brian Pinkerton presented the WebCrawler (Pinkerton, 1994).

Heydon & Najork (1999) presented a detailed description of the architecture and implementation of a broad crawler named Mercator, exposing situations hazardous to crawling found on the web. Later, Broder *et al.* (2003) investigated URL caching techniques to improve the detection of visited URLs in Mercator.

The Googlebot is present in the Google original research paper (Brin & Page, 1998). The WebBase crawler was a result of the continuation of that academic project (Cho *et al.*, 2004).

Silva *et al.* (1999) described the CobWeb crawler, one of the components of a search engine for the Brazilian web that used proxy servers to reduce implementation costs and save network bandwidth when updating a set of documents.

Boldi *et al.* (2002b) presented the Ubicrawler, giving special attention to its fault tolerance and scalability features.

Shkapenyuk & Suel (2002) presented the design and implementation of a high performance distributed crawler named Polybot.

Yan *et al.* (2002) presented the architectural design and evaluation of the Webgather crawler aimed to collect pages hosted in Chinese IP addresses.

The *Kspider* is a cluster based web crawler (Koht-Arsa, 2003; Sanguanpong & Koht-Arsa, 2003). Its authors presented technical optimizations and studied the partitioning of URLs among the Crawling Processes to ensure load balancing and avoid overloading the visited web servers.

Castillo (2004) discusses effective crawling techniques, presenting a crawler's implementation and the results obtained in several experiments performed while harvesting the Chilean web.

Crawler name	Frontier	Partitioning	Assignment	Meta-data	Focused crawls
Googlebot	centralized	?	?	no	no
Kspider	distributed	page	static	no	no
Mercator	distributed	site	static	yes	no
Polybot	distributed	page	dynamic-push	no	no
Ubicrawler	distributed	site	dynamic-?	no	no
WebBase	hybrid	site	dynamic-push	no	no
Webgather	distributed	IP	static	no	no

Table 2.8: Crawler design options.

The Internet Archive introduced an open-source web crawler called Heritrix, specially designed to collect and archive large collections of documents from the web (Mohr *et al.*, 2004).

Cho *et al.* (1998) studied how to order a fixed set of URLs so that the most important pages could be crawled first, implemented an incremental crawler (Cho & Garcia-Molina, 2000a), proposed and evaluated several architectural alternatives (Cho & Garcia-Molina, 2002).

2.2.5.1 Design comparison

Table 2.8 compares the design of some of the crawlers described in previous works.

In the Googlebot, the Frontier is centralized on a single URL server that distributes URLs among a set of Crawling Processes.

The Kspider is composed by a cluster of Crawling Processes that collaboratively maintain the Frontier and harvest the web. The URL space is partitioned uniformly among the Crawling Processes through the application of a hash function on the URLs. However, each CP groups the URLs hosted on the same site to enable the reuse of connections. When a CP finds an URL that does not belong to its partition, it sends it to the correspondent CP.

The Mercator crawler distributes the URLs to visit among several Crawling Processes that communicate using TCP. The URL space is divided using a site partitioning strategy.

In the Polybot crawler, the Frontier is distributed among a set of inter-communicating Crawling Applications and the URL space is partitioned uniformly through a hash function. The Crawling Applications send packets of URLs to the Crawl Manager, which assigns each one of them to a Downloader that crawls it. The Crawler Manager ensures that courtesy pauses are respected, caches DNS lookups and robots exclusion files. The communication among the processes is done through a NFS-mounted file system.

The Ubicrawler is composed by inter-communicating agents responsible for managing the Frontier and crawling the web. The assignment is dynamic, but there is not a central node of coordination because it is achieved through consistent hashing. Inside each agent, the URLs are partitioned per site among the threads, each one dedicated to visiting a single site. The number of agents can vary during the crawl, which makes Ubicrawler robust to failures and able to to increase download rate with additional agents. It is unclear if the agents pull URLs to crawl or if they are pushed from other agents.

The Webgather is composed by a static set of Main Controllers that communicate among each other to jointly manage the Frontier. The URL space is partitioned among the Main Controllers in the beginning of the crawl through the application of a hash function to the IP addresses of the machines to visit (IP partitioning strategy). Each Main Controller has an associated Gatherer that is responsible for crawling the web.

In the WebBase crawler, the URL space is partitioned by site. The Crawling Processes pull seeds from a central point of coordination named the Seed-URL Dispenser. The management of the Frontier is hybrid. The Seed-URL Dispenser manages the seeds and each CP manages the URLs extracted from the harvested pages. A CP harvests several sites simultaneously. To prevent DNS resolution from becoming a bottleneck during the crawl, all the site names are resolved in advance and the correspondent IP addresses are kept along with the seeds. However, this approach was time-consuming: it took three days to resolve 310 000 domain names and imposed a heavy continuous load on the DNS servers. The WebBase crawler only harvests the sites contained in the seeds list. The links to new sites found during the crawl are not harvested, they are added instead as seeds of the following crawl. This approach simplifies the crawling task because

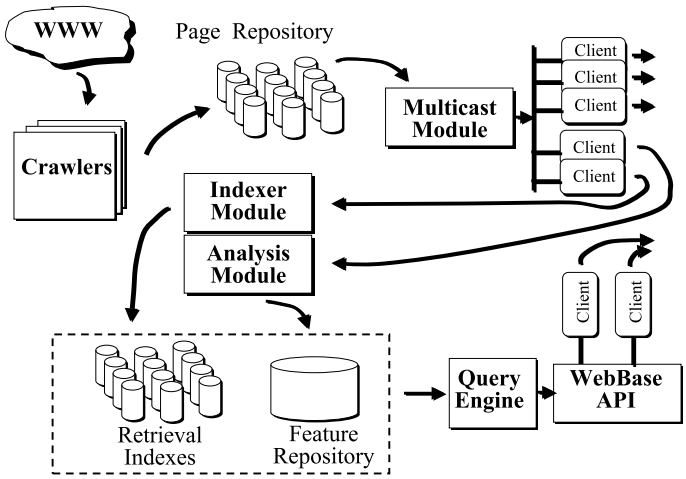


Figure 2.2: WebBase architecture (source: Cho *et al.* (2004)).

there is no exchange of URLs between the Crawling Processes, thus, no need for synchronization between the parts of the Frontier managed by them. However, this approach requires an exhaustive list of seeds that covers completely the portion of the web that will be harvested. For instance, to crawl all the sites from the .COM domain, the WebBase crawler requires seeds for all the sites registered under this domain.

2.3 Web Warehousing projects

The interest in Web Warehousing has grown in the last years. The emergence of the web as a main source of information and the economical success of web-based companies, such as Amazon, Google and Yahoo!, raised the interest in Web Warehousing projects by commercial and academic institutions. This section describes state-of-the-art Web Warehousing projects.

2.3.1 Stanford WebBase

The Stanford WebBase project aims to study effective algorithms for the acquisition, storage and indexing of large-scale web collections (Cho *et al.*, 2004). The

researchers involved in the project described the architecture of a repository of pages (Hirai *et al.*, 1999) and how to select and refresh them (Cho, 2001; Cho & Garcia-Molina, 2000b, 2002).

As experimental setup, the researchers developed a WWh that periodically harnesses web data. The architecture of the WebBase system is presented in Figure 2.2. The *Crawlers* harvest contents from the web and store them in a *Page Repository*. The *Multicast Module* distributes the stored pages among the clients trough streaming. These streams also feed the *Indexer* and *Analysis Modules* that construct indexes on the web data. The *WebBase API* and *Query Engine* enable the execution of queries over the stored data combining information from several indexes.

The WebBase WWh has been used by research and teaching organizations world-wide, mostly for research in web characterization and linguistic analysis (Boldi & Vigna, 2004; Richardson & Domingos, 2002; Sydow, 2004). The main advantage of using this WWh was that the researchers did not have to crawl the web to begin their studies.

2.3.2 WebFountain

IBM developed a similar project called WebFountain to perform large-scale analysis of texts (Gruhl *et al.*, 2004; McCurley & Tomkins, 2004). Over 300 researchers worldwide were involved in this project during four years.

The Semantic Web envisions the creation of machine-interpretable web data that enhances the extraction of knowledge from it (W3C, 2004). However, its implementation has been slow due to the divergence of standards and formats. The WebFountain project developed systems to enable the automatic generation of semantic annotations from the existent contents: the Seeker is a platform for large-scale text analytics and SemTag is an application written on that platform to perform automated semantic tagging on pages (Dill *et al.*, 2004). The Seeker architecture includes a centralized storage node, a full-text indexer, a scalable crawler and a query processing component named the Joiner. The analysis agents are applications that run within the Seeker environment. They were classified in two classes:

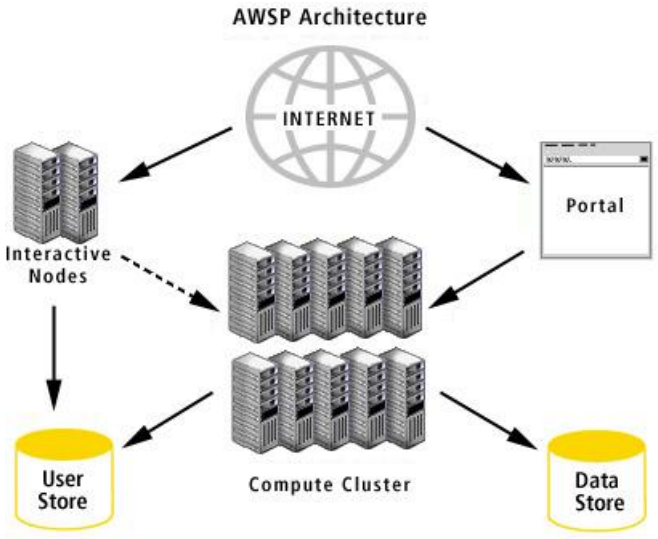


Figure 2.3: Architecture of the Alexa Web Search Platform (source: Sherfesee & O'Driscoll (2005)).

Annotators: process each content independently to generate meta-data;

Miners: require the processing of several contents in conjunction to extract additional information.

The characteristics of the web graph and the algorithms used to harness it were thoroughly studied during the WebFountain project (Eiron *et al.*, 2004; Kumar *et al.*, 2000, 2005).

2.3.3 Alexa Web Search Platform

The Alexa Web Search Platform (AWSP) is a commercial application that provides access to a vast crawl of the web (Sherfesee & O'Driscoll, 2005). Users can process contents using the AWSP tools and hardware infrastructure. The workflow begins with the definition of the portion of contents to process. Then, the user writes an application to extract the desired information, runs it and finally collects the results as an XML feed. The obtained results can be viewed interactively, transferred or published as a new service.

Figure 2.3 presents the AWSP architecture. The *Data Store* aims to keep an up-to-date snapshot of the web, containing contents and meta-data. The contents are kept for six months and are then replaced with new ones. The *User Store* provides storage space where the users can keep their private data, such as source code, applications or results from content analysis. The users can also use a part of their storage space to share information with other users. The *Compute Cluster* is a set of collaborative computers that enable the parallel execution of user tasks. The *Interactive Nodes* are computers that can be accessed exclusively by a single user. The *Portal* is the gateway to access the Alexa AWSP, where users can manage their accounts and reserve resources to run their jobs.

2.3.4 Whoweda

Bhowmick *et al.* (2003) motivated the necessity of web warehouses with the differences existent between web data and the relational data that usually feeds data warehouses. They proposed a WWh architecture and focused on the *coupling engine* and the *web manipulator* modules. The coupling engine is responsible for extracting relevant data from multiple sites and storing it in the WWh. The web manipulator is responsible for manipulating the integrated data. The main purpose of the research was to explore a data model and query language for representing and storing relevant data from the web.

Query languages and access methods are important tools to harness web data. However, the assumptions made in this research did not reflect the characteristics of web data and may jeopardize the application of its contributions in practice. For instance, the authors assumed that the web was composed by well-formed HTML or XML documents (formatted according to the specification), or plain texts. Pages dynamically generated were not considered within the scope of the work and peculiar characteristics of web data, such as duplication, were also not addressed. Several experimental studies contradict these assumptions. Around 30% of the documents on the web are not HTML (Heydon & Najork, 1999), the number of XML files published on the web is not significative (Abiteboul *et al.*, 2000), most of the web pages are not well-formed (Woodruff *et al.*, 1996), there are

much more dynamically generated web pages than static ones (Handschuh *et al.*, 2003) and duplication is pervasive on the web as it was shown in Section 2.1.3.3.

2.3.5 Search engines

Search engines were the first systems to implement large-scale web warehouses. A research paper describes the original architecture of the Google search engine when it was still an academic project (Brin & Page, 1998). Notably, after the commercial success of this search engine, several papers continued to describe its functioning. The architecture of the 15 000 commodity class PCs that compose the Google search engine was described by Barroso *et al.* (2003) as well as the software components specially developed to fulfill the needs of a large-scale search engine (Ghemawat *et al.*, 2003). Dean & Ghemawat (2004) presented the Google's MapReduce programming model created to simplify the data processing on large clusters of machines. A programmer just needs to write two functions: the map function that processes a key/value pair to generate a set of intermediate key/value pairs and; the reduce function that merges all intermediate values associated with the same intermediate key. The run-time system partitions the input data, schedules the program execution across the cluster, guarantees fault tolerance and manages inter-machine communication enabling programmers without experience in distributed systems to execute tasks in a distributed fashion (Pike *et al.*, 2005).

Hadoop is an Apache Software Foundation project that implemented an open-source framework for running applications based on the MapReduce programming model (The Apache Software Foundation, 2006). Hadoop also provides a fault-tolerant distributed file system to store data across several machines. It was successfully tested on a cluster of 620 storage nodes. Hadoop was originally part of the Nutch project that created open source web-search software (Cafarella & Cutting, 2004). Later, it became an independent project.

Lifantsev & Chiueh (2003) presented the design, architecture, implementation and performance measurements of the Yuntis search engine prototype. Yuntis was developed as an academic project for three years. The authors used Yuntis to

evaluate a voting model for assessing quality and relevance of pages (Lifantsev, 2000).

Search engines can be divided in two classes according to the data they harness:

Broad. Harvest the largest amount of pages or the most popular ones on the web within a limited time interval, independently from the subjects addressed in them;

Topical. Harvest information relevant to a specific topic from the web, usually with the help of a classification algorithm, to filter irrelevant contents (Chakrabarti *et al.*, 1999; Qin *et al.*, 2004). A topical search engine may be designed to fulfill the requirements of a given community of web users, such as a national community (Almeida & Almeida, 2004).

Independently from their scope, search engines face similar design problems that must be addressed according to the characteristics of the portion of the web that is indexed, such as avoiding duplicated data from being harvested and presented as different search results (Henzinger *et al.*, 2002; Patterson, 2004).

2.3.6 Web archives

A web archive stores and enables access to web data for historical purposes. These data must be periodically processed to ensure its preservation and accessibility.

The Internet Archive was the pioneer web archive system and it has been executing broad crawls of the web since 1996 (Kahle, 2002). Currently, there are 16 countries with well-established national web archiving programs (National Library of Australia, 2006).

The National Library of Australia founded its web archive initiative in 1996 (Phillips, 2003). It developed the PANDAS (PANDORA Digital Archiving System) software to periodically archive Australian online publications, selected by librarians for their historical value.

The British Library leads a consortium that is investigating the issues of web archival (Consortium, 2006). The project aims to collect and archive 6 000

selected sites from the United Kingdom during two years using the PANDAS software. The sites have been stored, catalogued and checked for completeness.

The MINERVA (Mapping the INternet Electronic Resources Virtual Archive) Web Archiving Project was created by the Library of the Congress of the USA and archives specific publications available on the web that are related to important events, such as an election (The Library of Congress, 2006).

In December 2004, the Danish parliament passed a new legal deposit law that calls for the harvesting of the Danish part of the web for the purpose of preserving cultural heritage and two libraries became responsible for the development of the Netarkivet web archive (Christensen, 2005).

The legal deposit of contents in France will be divided among the Institut National de l'Audiovisuel (INA) and the National Library of France (BnF). Drugeon (2005) presented a detailed description of the system developed to crawl and archive specific sites related to media and audiovisual. The authors estimated that INA will be responsible for the periodic archive of 10 000 to 15 000 sites. The BnF will be responsible for the archive of online writings and newspapers and preliminary work in cooperation with a French national research institute (INRIA) has begun (Abiteboul *et al.*, 2002).

The National Library of Norway had a 3-year project called Paradigma (2001-2004) to find the technology, methods and organization for the collection and preservation of electronic documents (Albertsen, 2003). The main objective of this project was to provide access to the archived data to the library's users.

The NEDLIB project (1998-2000) grouped national libraries from several countries and had the purpose of developing harvesting software specifically for the collection of web resources for an European deposit library (Hakala, 2001). The Austrian National Library together with the Department of Software Technology at the Technical University of Vienna, initiated the AOLA project (Austrian On-Line Archive) (Rauber *et al.*, 2002). The goal of this project is to build an archive by periodically harvesting the Austrian web. The national libraries of Finland, Iceland, Denmark, Norway and Sweden participate in the Nordic Web Archive (NWA) project (Hallgrímsson & Bang, 2003). The purpose of this project is to develop an open-source software tool set that enables the archive and access to web collections.

There were researchers that contributed with the study of systems to be used in web archiving. Crespo & Garcia-Molina (1998) proposed a multi-layer architecture formed by a federation of independent, heterogeneous and collaborative sites that ensured the preservation of documents and relationships among them, through a replication schema. Later, this architecture was used in the Stanford Archival Repository Project where simulators were included to evaluate the effectiveness of an archive over a long time span (Cooper *et al.*, 2002). Burkard (2002) presented a web archival system based on a Peer-to-Peer architecture called Herodotus. You & Karamanolis (2004) evaluated of compression file techniques to efficiently archive web data.

2.4 Conclusions

This chapter presented previous works on web characterization, crawling and Web Warehousing systems. Research in these fields remains active because the web is permanently evolving. New models are required to reflect the current state of the web, crawlers have to select and harvest larger amounts of data and efficient Web Warehouses are required, because they became essential tools to extract knowledge from web data.

Web warehouses are usually loaded with a portion of the web relevant to a community of users. Several web characterization studies showed that the web is composed of distinct portions with peculiar characteristics. It is important to accurately define the boundaries of these portions and model them, so that the design of a WWh can reflect the characteristics of the data it will store. The methodology used to sample the web influences the derived characterizations. Hence, the samples used to model a portion of the web must be gathered using a methodology that emulates the extraction stage of the web data integration process. As most web warehouses use crawlers to extract information from the web, this is the most adequate sampling method.

Previous works focused on architectural aspects of web crawlers and warehouses. There are high-performance crawlers that enable the quick refreshment of the data kept in a WWh. Web Warehousing projects produced complete systems able to harvest, store and provide access to large amounts of web data.

However, most research in these domains has assumed that the web is uniform, an assumption not supported by the results obtained in web characterization studies.

This thesis discusses how the characteristics of a portion of the web could influence the design of a WWh. The next two Chapters present a model for a national web. They discuss how the crawling process can bias a web characterization and, on its turn, how the web characteristics can influence the design of a WWh.

Chapter 3

Characterizing the structural properties of a national web

The web can be characterized from multiple perspectives using numerous metrics. This is a challenging task, mainly because of its heterogeneity, large dimension and permanent evolution (Leung *et al.*, 2001). Producing a feasible general characterization is hard, and some statistics derived from the analysis of the global web may not hold as one scales down to more restricted domains. The web has portions with specific characteristics that, given their small presence, do not become visible in a general web characterization. However, these portions can be of interest to large communities, such as those representing national or cultural groups. Additionally, characterizing a small partition of the web is quite accessible and can be done with great accuracy.

In general, a Web Warehouse (WWh) is populated with contents relevant to a given community of users corresponding to a portion of the web. As a result, a characterization of the portion of the web to be warehoused is crucial to efficiently design and manage a WWh. The results presented in the previous chapter showed that the web is not uniform and there are portions of it with peculiar characteristics.

The Portuguese Web, broadly defined as the set of contents of cultural and sociological interest to the people of Portugal, was used in this thesis, as a case study. It has been extensively characterized and crawled to feed a WWh between 2003 and 2006.

This chapter presents a detailed characterization of the Portuguese Web. As the methodologies used to gather web samples can influence a web characterization, crawling was used to sample the web, because it is the most commonly used web data extraction process. However, the crawler configuration can also influence a web characterization. This chapter details the adopted crawling policy and shows how the crawling and data analysis processes can strongly influence other statistics. The statistics themselves are interesting to anyone who manipulates these data or will compare them with future characterizations. In addition, the identification of meaningful statistics for a community web characterization and the methods used to gather and interpret the collected data could be useful to a wider audience those interested in the Portuguese Web in particular.

This chapter is organized as follows: Section 3.1 discusses selection criteria for delimiting the boundaries of the Portuguese Web as a crawling policy. The following section describes the experimental setup used to derive the web characterization. Section 3.3 presents the obtained model of the Portuguese Web, detailing the characteristics of its sites, contents and link structure. Finally, Section 3.4 presents the main conclusions derived from the experience of characterizing the structural properties of the Portuguese Web.

3.1 Identifying the boundaries of a national web

The identification of a community web is not always obvious, despite of available methods that can be used to identify its sites (Flake *et al.*, 2000). A precise definition of which contents should constitute a community web is in general hard to obtain and is conditioned by the rules and resources used.

A country's web is interesting to communities of users that are interested in information with a national scope. For instance, when users look for the "National Library" site, they want to find the site of the library of their country. Empirically, a national web is the set of contents that contain information related to a given country. However, this definition is subjective and therefore hard to be translated into machine-understandable heuristics.

The country-code Top Level Domains (ccTLD) are delegated to national managers to fulfil the local requirements. Thus, sites with domain names under a

ccTLD should be included in a national web. However, due to the strict legal requirements and costs for domain registrations imposed in some countries for domain registrations, many sites containing information related to a national web are registered under general purpose Top Level Domains (gTLD), such as .COM. To overcome this problem, a national web could also include contents hosted on IP addresses that are physically assigned to a given country (Baeza-Yates *et al.*, 2005). However, the physical location of an IP address is provided by databases, such as the RIPE Network Management Database (RARE, 1992), that sometimes provide erroneous locations for IP addresses. Moreover, due to the global nature of the Internet, publishers tend to use the hosting services that provide better prices, independently from the physical location of their servers. Therefore, sites with gTLDs hosted on foreign IP addresses are not part of a national web according to this definition. Some languages, such as Swedish, are spoken in a single country in the world. In these cases, a national web can be identified as the set of contents written in a given language (Albertsen, 2003). However, other languages, such as English or Portuguese, are spoken in several different countries and cannot be considered a defining criterion for a national web. The WHOIS databases contains contact information about the owners of domain names or IP addresses (Harrenstien *et al.*, 1985). Therefore, they could be used to identify sites owned by people of a given country. However, there are domain owners that provide subdomains to host foreign sites, such as `blogspot.com`, in these cases the contents of the sites are not related to the domain owners.

One could consider a national web as the union of the sites identified by all the presented heuristics. Probably, this approach would a achieve a good coverage of a national web. On the other hand, it would include contents that users would not consider as belonging to a given national web. The automatic identification of national webs is interesting to relatively large communities of users. However, it is not straightforward to achieve.

3.1.1 Definition of the Portuguese Web

The Portuguese Web is broadly defined as the set of contents containing information related to Portugal or of major interest to the Portuguese people. As defining

rule, a content was considered as part of the Portuguese Web if it satisfied one of the following conditions:

Condition 1: hosted on a site under the .PT domain;

Condition 2: hosted on a site under the .COM, .NET, .ORG or .TV domains, written in Portuguese and with at least one incoming link originating in a page hosted under the .PT domain.

This definition aims to be easily set as a crawling policy and to guarantee that the crawler obtains the best coverage of the Portuguese Web.

Condition 1 goal is to include the sites that form what was considered the core of the Portuguese Web. A list of the most popular sites, accessed from the homes of a panel of Portuguese users, during 2002 and 2003, showed that 49.5% of the sites were hosted under the .PT domain (Marktest, 2003).

Condition 2 goal is to include the increasing number of Portuguese sites that are registered outside the .PT domain (Zook, 2000). Previous work showed that the link structure of the web can be used to define communities (Flake *et al.*, 2000; Gibson *et al.*, 1998). Condition 2 assumes that the probability of a site hosted outside the .PT domain belonging to the Portuguese Web community decreases as the number of hops in the web graph to the core increases. So, in order to restrict the inclusion of sites outside the .PT domain to those with the highest probability of being part of the Portuguese Web, the number of hops was limited to 1. Condition 2 imposes that only contents directly linked from a site hosted under the .PT domain are part of the Portuguese community web. However, Brazilian contents linked from the .PT domain and hosted under the allowed domains, such as .COM will still be considered as part of the Portuguese Web.

3.1.2 Finding contents outside the ccTLD

A definition of a national community web has an implicit geographical context. An experiment was conducted with the purpose of comparing the coverage of the Portuguese community web outside the .PT domain (Condition 2) with three alternative heuristics that used tools to assign geographic context data for sites.

DNS LOC. Geographic information for the sites could be obtained by querying a special DNS record (LOC) that carries location information about hosts, networks, and subnets (Davis *et al.*, 1996). A site was considered as part of the Portuguese Web if the DNS LOC located it in Portugal;

Geographical tools. Two commercial tools, Ip2location (Hexa Software Development Center, 2003) and Maxmind (Maxmind LLC, 2003) have been used to extract a geographical location for a site. A site was considered as part of the Portuguese Web if the tool returned that the site was located in Portugal. For two submissions of the same site, Maxmind returned different results. Except for this situation, both tools presented the same results, which suggests that they are based on the same data;

WHOIS database. This heuristic consisted on accessing a WHOIS database to identify the Portuguese sites. For each site, the contact address for the correspondent domain registrant was obtained. If this address was located in Portugal, the site was considered as part of the Portuguese Web.

The experimental data set was composed by the following list of 25 Portuguese sites hosted outside the .PT domain informally suggested by a group of Portuguese users in 2003. The suggested sites were humanly examined to verify if their contents were related to the Portuguese community. All the sites were written in Portuguese and referred to distinct subjects, such as sports, humor or radio.

The definition of the Portuguese Web described in the previous section, based on language identification and the link structure of the web, was validated by checking if the sites had at least one incoming link from a site hosted under the .PT domain. The search engines Google (Google, 2003) and AllTheWeb (Overture Services Inc., 2003) were used to identify pages that link to the sites.

Table 3.1 presents the results given by the four heuristics to identify Portuguese contents outside the .PT domain.

None of the sites had an associated DNS LOC record.

The geographical tools identified only 44% of the Portuguese sites, although they always returned an answer to the location requests. 76% of the Portuguese sites were identified through the registrant information.

Heuristic	% sites identified
DNS LOC	0%
Geographical tools	44%
WHOIS registrant address	76%
Language and links	82%

Table 3.1: Comparison between alternative definitions of the Portuguese Web for a sample of 25 sites.

The WHOIS database did not contain the information regarding the requested domain for 24% of the sites. For some of these cases, the registrant information was found on another WHOIS server. As there is not a central WHOIS database, the registrants information is distributed over the several registrars, which causes inconsistencies among WHOIS databases. The registrant address revealed to be a precise method to identify Portuguese sites outside the .PT domain. All the sites in the list which had a WHOIS record available were correctly identified. Therefore, the WHOIS databases could be the solution to the problem of distinguishing Brazilian sites from Portuguese sites outside the .PT domain. However, most of the WHOIS databases are not publicly available or explicitly forbid their access by automated programs, which collides with the purpose of having a definition of the Portuguese Web that can be implemented as a crawling policy. There are several record formats in use, which also makes it difficult to automatically process WHOIS records. Additionally, some of the companies that provide hosting services put distinct sites identified by sub-domains under the same domain and the WHOIS registries only keep information about second-level domains. This it is a serious restriction considering, for instance, all the Portuguese Blogs hosted under `blogspot.com`. Hence, this heuristic excludes from the Portuguese Web many Portuguese sites hosted outside Portugal.

82% of the suggested sites would be included in the Portuguese Web using the initially proposed definition of the Portuguese Web: they were written in the Portuguese language and had at least one link from a site hosted under a .PT domain.

3.2 Experimental setup

This section describes the crawler configuration and the data set used to derive a characterization of the structural properties of the Portuguese Web.

3.2.1 Crawler configuration

Crawlers are configured according to the purpose of the data they gather. The crawler was configured to get the most textual contents as possible from the Portuguese Web, under the minimum set of constraints that ensure an acceptable performance, considering the resources available and the need to make it robust against the anomalies found on the web.

A content was considered to be valid if it was part of the Portuguese Web, as defined in the previous section. In addition, the following crawler conditions had to be met:

Multiple text types. The crawler harvested not just pages, but also contents of common MIME application types that it could convert to text. Accepted MIME types are: text/html, text/richtext, text/tab-separated-values, text/plain, text/rtf, application/pdf, application/rtf, application/x-shockwave-flash, application/x-tex, application/msword, application/vnd.ms-excel, application/excel, application/mspowerpoint, application/powerpoint and application/vnd.ms-powerpoint;

URL depths less than 6. The crawler followed at most five links in breadth-first search order, from the seed of the site until it reached the referenced content. When crawling a site, any link found to a different site was set as a seed to that site. This way, any page with a link originated on a .PT domain would be visited, including Portuguese subsites hosted on foreign countries. Consider for instance, the site `www.yahoo.com` and its Portuguese subsite `www.yahoo.com/users/myPortugueseSite/`. If the crawler had visited only the seed `www.yahoo.com`, it would had identified that the site was not part of the Portuguese Web, and exited without finding the Portuguese subsite;

Contents downloaded in less than one minute. This prevents very slow web servers from blocking the progress of the crawl;

Contents size under 2 MB. This prevents the download of very large files available on the web, such as database dumps.

3.2.1.1 Spider trap biasing and mitigation

A crawler trap is a set of URLs that cause a crawler to traverse a site indefinitely. A crawler trap is noticed due to the large number of contents discovered in the site (Heydon & Najork, 1999). Most of the traps are unintentional, being caused mainly by session identifiers embedded in the URLs, or poorly designed HTTP web applications that dynamically generate an infinite number of URLs, which in turn reference a small set of contents. This raises the issue of how should these contents be considered in a characterization. They should not be excluded because they are available online and represent part of the web. However, they cannot bias a characterization with their "infinite" presence. The solution adopted was to set the crawler as a very patient web surfer as a compromise. The crawler was set to visit a maximum of 8 000 URLs per site, to prevent the crawling of infinite sites.

After seeing 50 duplicates, the crawler gives up on following links for a site, keeping all the information crawled until then. This limitation intends to avoid spider traps that always return the exact same content. If the trap generates slightly different contents, it will be identified when the site reaches the maximum number of contents allowed. A criterion that identifies contents with distinct URLs and contents as being similar enough to be considered the same is highly subjective. Plus, the computation of partial similarity between contents is too expensive to be applied in the identification of spider traps during the crawling process.

An intentional trap could be created using DNS wildcarding (Mockapetris, 1987) to resolve any possible host name within a domain to the same IP address, generating an infinite number of host aliases and giving crawlers the illusion that each site serves only a small number of pages. In order to mitigate this situation, the crawler avoids harvesting host aliases by identifying them through

State	Nr. URLs	%
200 (OK)	3 235 140	83.9
302 (Temporary redirect)	193 870	5.0
404 (Not found)	132 834	3.4
TimedOut	45 486	1.2
301 (Moved permanently)	39 920	1.0
ExcludedByREP	35 596	0.9
500 (Internal server error)	33 247	0.9
NotAllowedType	25 976	0.7
403 (Forbidden)	18 598	0.5
UnknownHost	17 842	0.5
SizeTooBig	17 453	0.5
ConversionError	13 986	0.4
Other	23 244	0.6
Total	3 856 436	100.0

Table 3.2: Status codes of the visited URLs.

a pre-computed list gathered from a previous crawl. If two sites were hosted on the same IP address and had the same home page, they were considered host aliases. The host aliases list was derived using online information before starting the crawl, so that it could be as accurate as possible.

3.2.2 Data set

The sample of the Portuguese Web was crawled between the 1st of April and the 15th of May, 2003. The crawler was bootstrapped with a set of 112 146 seeds gathered from:

- Previous crawls performed for the tumba! search engine;
- Site names derived from a list domains registered under the .PT domain obtained from the official registry service. This list included all the second-level domains under .PT plus, some third level domains administrated by the same registry service, such as COM.PT;
- Sites submitted by Portuguese users to be included in the tumba! search engine.

All these sources of seeds were used to enable the harvesting of *orphan sites*, also known as *islands* (Baeza-Yates *et al.*, 2005). These sites do not receive links from external sites and therefore are hard to be reached through the automatic following of links in web pages.

The crawler visited a total of 146 076 sites, processed over 3.8 million URLs and downloaded 78 GB of data[1].

Table 3.2 presents the statistics of the download status of crawled URLs. Almost 84% of the requests resulted in a successful download and only 3.4% resulted in a 404 (File Not Found) response code, which indicates that most of the seeds were valid and that broken links are not as frequent in this web as reported in other studies (Najork & Heydon, 2001; Spinellis, 2003). There were over 6% of redirections and the crawler failed to fetch and parse a content within one minute in 1.2% of the requests. The Robots Exclusion Protocol prevented the crawler from downloading 0.9% of the URLs, and about the same number of URLs resulted in an Internal Server Error (500). The number of contents with a not allowed MIME type (0.7%) is underestimated, because extracted links that had names hinting that the referenced content did not belong to one of the allowed types (e.g. files with a .JPEG extension) were not harvested. The UnknownHost error (0.5%) is caused by URLs referencing site names that no longer have an associated IP address. Only 0.5% of the referenced files had a size bigger than 2 MB and the conversion to text was not possible in 0.4% of the cases. The remaining situations (0.6%) included other HTTP response codes, unidentified errors, socket and connection errors; each of these represents less than 0.1% of the total number of downloaded contents.

3.3 A model of the Portuguese Web

This section presents the statistics derived from the analysis of the data set. It characterizes the main metrics that influence the design of a WWh.

[1]The information gathered in this crawl is available for research purposes at http://xldb.fc.ul.pt/linguateca/WPT_03.html.

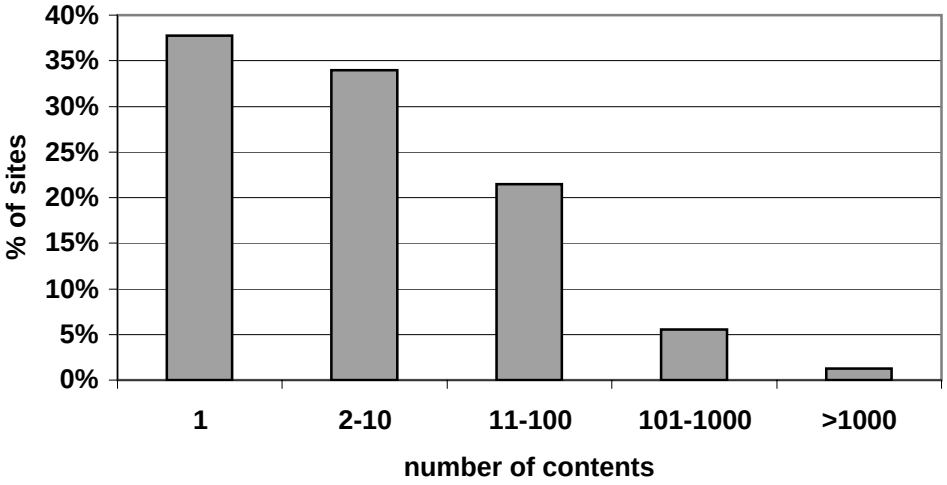


Figure 3.1: Distribution of contents per site.

3.3.1 Site characteristics

A site groups contents that provide related inter-information or are hosted at the same physical location. The characteristics of sites can be explored to extract and store data efficiently in a WWh. As the definition of site may vary, this section describes the site sizes considering that each of them is identified by a site name or an IP address. Then, it describes the distribution of sites among TLDs. Finally, this section presents statistics on the web server software used on the Portuguese sites.

3.3.1.1 Site names

A Portuguese site hosts on average 70 contents, but the size distribution is very skewed, as shown in Figure 3.1. The number of sites having a single content was surprisingly high (38%). A visit to a random sample of these sites revealed that most of them warned readers that the sites were under construction or moved to a different location. There were also a few cases where the home page was written using scripting languages preventing the crawler from extracting links to find other contents within the site. A typical site had less than 101 contents

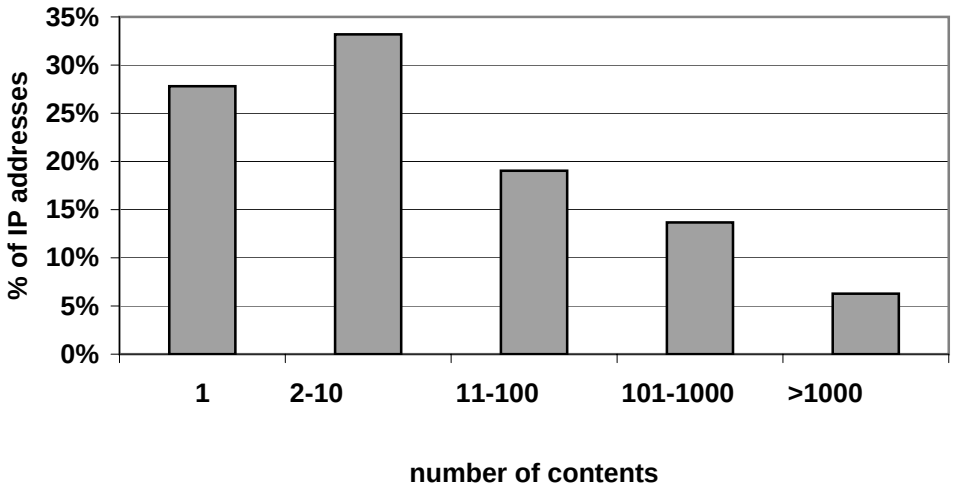


Figure 3.2: Distribution of contents per IP address.

(93%); 6% had between 101 and 1 000 contents and only 1% of the sites had more than 1 000 contents.

Only 577 sites (0.4%) hosted more than the maximum number of 8 000 contents. Most of these sites were huge database dumps published online as dynamically generated pages. The constraint on the maximum site size reduced the number of unnecessary downloads and increased the robustness of the crawler, at the cost of excluding less than 1% of the sites.

3.3.1.2 IP addresses

The distribution of contents per IP address is more uniform than for site names (see Figure 3.2). The percentage of IP addresses that host just one content is 28%, IP addresses that host 2 to 10 contents represent 33%, those which host between 11 and 1 000 contents represent 33% and only 6% host more than 1 000 contents.

Table 3.3 shows that over 32% of the IP addresses host more than one site. Each IP address hosts an average of 6.78 sites. Silva *et al.* (2002) compared results from two crawls of the .PT domain performed in 2001 and 2002, and observed that the number of sites per IP address grew from an average of 3.78 to 4.57 sites

Nr. sites per IP	Nr. IP addresses	%
1	4 643	67.7
2-10	1 931	28.2
11-100	247	3.6
101-1 000	30	0.4
>1 000	5	0.1
Total	6 856	100.0

Table 3.3: Distribution of the number of sites hosted per IP address.

per IP address. The result presented in this thesis suggests that the number of sites hosted per IP address continues to grow. There are five IP addresses that host more than 1 000 sites. These five IP addresses are from portals that offer their clients a virtual host under the portal domain name.

Virtual hosts are very popular on the Portuguese Web: 82% of all sites are virtual hosts. It is important to distinguish host aliases from distinct virtual hosts. The first occur when multiple names refer to the same site, for instance `http://xldb.fc.ul.pt` and `http://xldb.di.fc.ul.pt`. Distinct virtual hosts are distinct sites hosted on the same machine, such as `http://xldb.di.fc.ul.pt` and `http://lasige.di.fc.ul.pt`. An analysis of the data set revealed that 8.5% of the virtual hosts were host aliases.

3.3.1.3 Domain distribution

A site is considered part of the Portuguese Web if it hosts at least one content is considered part of the Portuguese Web. 46 457 (32%) of the 146 076 sites crawled were identified as Portuguese sites. 84.2% of these, were under the .PT domain, 12.5% were under the .COM, 2.5% were under the .NET domain and just 0.8% sites were under the .ORG (see Figure 3.3). 60% of the site names started with "WWW".

3.3.1.4 Web servers

172 distinct HTTP web servers have been identified through the analysis of the Server HTTP header field (Fielding *et al.*, 1999): Figure 3.4 presents their distribution. The Portuguese sites are mainly hosted in Apache (57%) and Microsoft

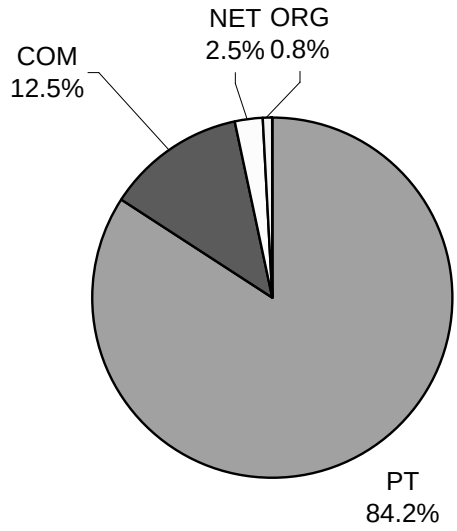


Figure 3.3: Distribution of sites per top-level domain.

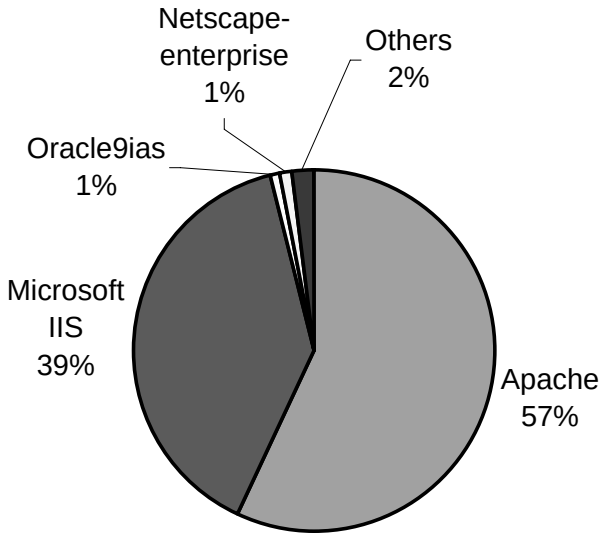


Figure 3.4: Distribution of web servers.

Figure 3.5: Distribution of URL lengths.

IIS (39%) web servers. The next two web servers (Netscape-enterprise and Oracle9ias) represent just 1% each and the remaining just 2%. Statistics on the global web present a similar percentage of Apache web servers (62.5%), but a considerably smaller percentage of Microsoft IIS servers (27.4%) (Netcraft Ltd., 2004). On the other hand, the distribution of web server software on the Portuguese Web contrasts with the one obtained by Boldi *et al.* (2002a) for the African web, in which there is a dominance of Microsoft IIS over Apache (56.1% against 37.9%).

Some security experts encourage webmasters not to provide the Server HTTP field or to provide wrong answers in order to mislead possible attackers. From my experience, I believe that these recommendations are usually not followed by the Portuguese webmasters. However, if they become popular, it will be very difficult to correctly identify the distribution of web server software.

3.3.2 Content characteristics

This section presents measurements regarding the URL length, media type, size, language and meta-data of contents.

3.3.2.1 URL length

A WWh must have an efficient data structure that maps the URLs into the stored contents, which must be designed according to the length of the URLs. However,

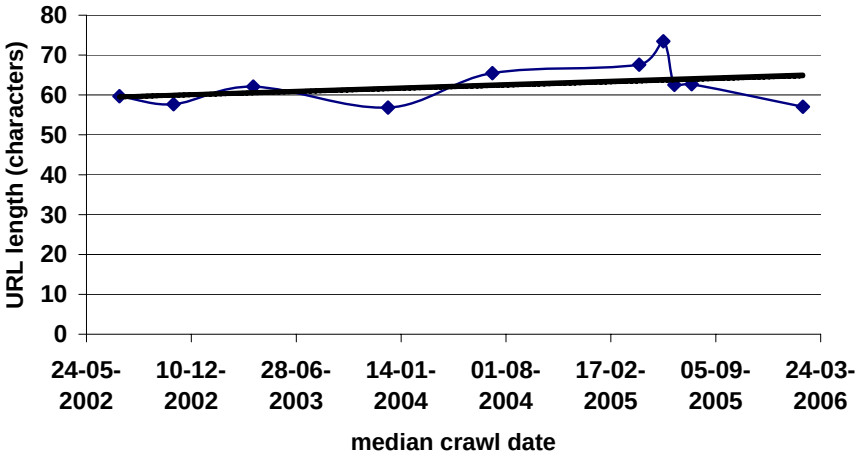


Figure 3.6: Evolution of URL lengths.

the URLs' length is in practice unlimited. Figure 3.5 shows the distribution of URL lengths (not considering the initial seven characters of the protocol definition). The Portuguese Web contained valid URLs with lengths varying from 5 to 1 368 characters. Most of the contents are referenced by an URL with length between 20 and 100 characters, with an average value of 62 and a median of 54. An analysis of the URLs revealed that 82% contained parameters suggesting that the referenced content had been dynamically generated. The growing popularity of these contents could originate a growth of the URL lengths. However, Figure 3.6 shows that the length of the URLs of the Portuguese Web has remained quite stable in the past years (2002–2006), presenting an average size of 62.5 characters per URL.

3.3.2.2 Last-Modified dates

The HTTP header field Last-Modified provides the date of the last modification of a content. However, as shown in Figure 3.7, most of the contents (53.5%) returned an unknown value for this field. Plus, Mogul (1999b) showed that even the returned values are many times inaccurate due to incorrectly set web server clocks (among other situations). An analysis of the URLs with unknown values revealed that 82% of them had embedded parameters. I believe that most of

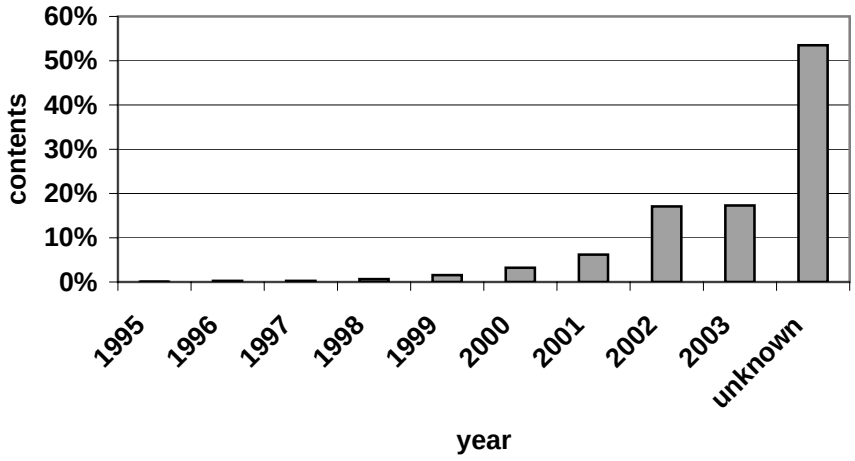


Figure 3.7: Distribution of Last-Modified dates.

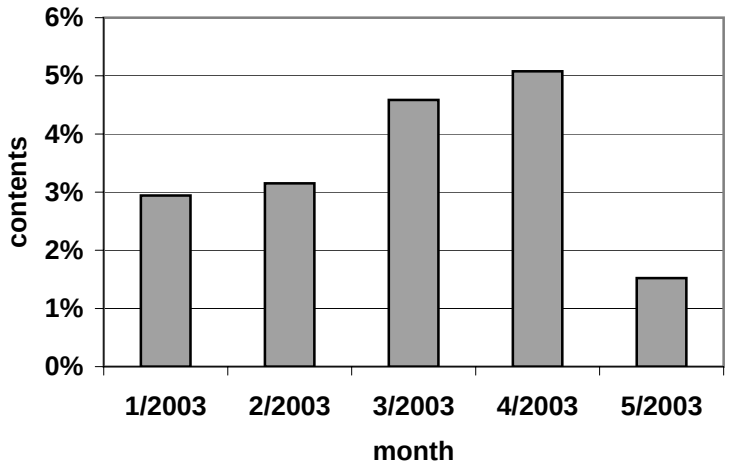


Figure 3.8: Distribution of Last-Modified dates in the last four months before the crawl.

MIME type	Nr. of contents	%
text/html	3 104 771	95.9
application/pdf	62 141	1.9
text/plain	33 091	1.0
application/x-shockwave-flash	17 598	0.5
application/msword	14 014	0.4
powerpoint	2 085	0.1
excel	915	0.0
application/x-tex	222	0.0
text/rtf	194	0.0
application/rtf	66	0.0
text/tab-separated-values	41	0.0
text/richtext	2	0.0
Total	3 235 140	100.0

Table 3.4: Number of contents and prevalence on the web for each MIME type collected.

them were recently modified (see Figure 3.8), since mechanisms to dynamically generate contents are usually used to reference short life contents, such as news. The obtained results show that the Last-Modified header field is a weak metric for evaluating changes and evolution of web contents.

3.3.2.3 Media type and size

The rightmost column of Table 3.4 shows the distribution of contents per MIME type. All the MIME types used for Microsoft Powerpoint files are grouped under the name *powerpoint* and all the Microsoft Excel files under the name *excel*. The predominant text format is text/html, present in over 95% of the collected contents, followed by application/pdf with just 1.9%.

The HTTP header field Content-Length was analyzed to determine the size of the contents, but 33% of the requests returned an unknown value. Hence, the downloaded contents were analyzed and the unknown sizes were replaced by their actual sizes. The differences on the average sizes between the results were insignificant, except for text/html where the size grew from 12.2 KB to 20.5 KB. In Table 3.5, the second and third columns show the average sizes of contents and corresponding extracted texts (without any formatting tags), and the fourth

MIME type	Avg. content size (KB)	Avg. text size(KB)	% text
powerpoint	1 054.9	7.0	0.7
text/rtf	475.6	1.2	0.3
application/pdf	207.4	13.6	6.6
application/rtf	121.3	4.7	3.9
application/msword	118.6	9.9	8.3
excel	50.4	21.9	43.4
application/x-shockwave-flash	43.9	0.3	0.7
text/html	20.5	2.5	12.2
text/richtext	16.3	16.2	99.2
application/x-tex	16.1	14.7	91.2
text/plain	10.5	7.8	74
text/tab-separated-values	3.9	3.8	97.5

Table 3.5: Average size, extracted text size and percentage of extracted text.

column presents the ratio between the length of the extracted text and content size. The smaller contents tend to provide more text than the larger ones. A curious fact is how text/plain contents represent 74% of text. An analysis of some of these contents revealed that some web servers return text/plain, when the file type of the content is not recognized. Therefore, some PowerPoint Presentation files (.PPS) or Java Archives (.JAR) were incorrectly processed as text/plain, resulting in poor extraction of text from these files.

Figure 3.9 shows the general distribution of content sizes. Most contents have sizes between 4 and 64 KB. The mean size of a content is 32.4 KB and the mean size of the extracted texts is 2.8 KB. The total size of the contents was 78.4 GB, while the total size of extracted texts was just 8.8 GB (11%). Hence, a WWh that just keeps the extracted texts has much less demanding storage requirements than one that stores the original contents.

The previous chapter referenced several works that state that the size of web contents tends to increase. Figure 3.10 shows that the evolution of content sizes on the Portuguese Web also follows this trend. In three years, the average size of a content almost doubled. The growth of content size with time matches a linear function with an R-squared value of *0.9163*. This model is useful to estimate the storage capacity required for a WWh.

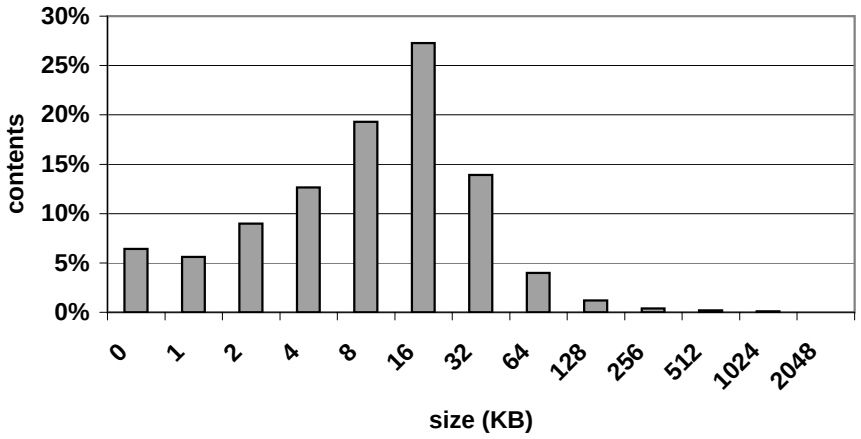


Figure 3.9: Distribution of content sizes.

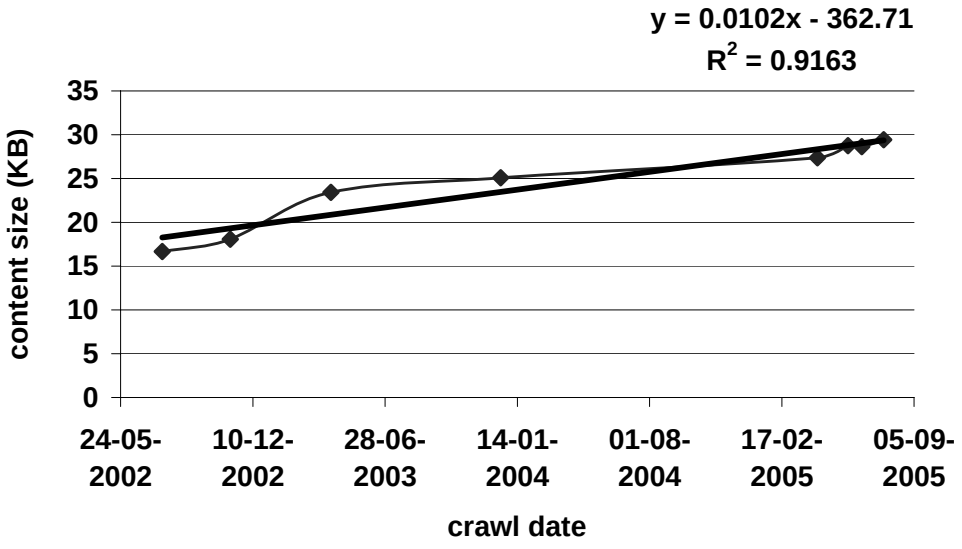


Figure 3.10: Evolution of average content size.

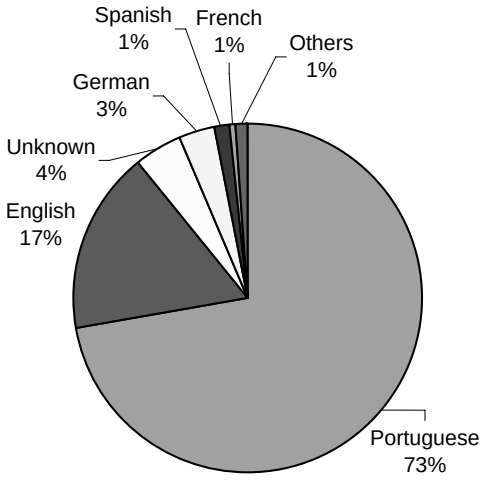


Figure 3.11: Distribution of languages under the .PT domain.

3.3.2.4 Language

The language of the contents was identified based on an language detector that implements an n-gram algorithm (Martins & Silva, 2005a). Figure 3.11 shows the distribution of languages on the contents written in all languages hosted under the .PT domain: 73% of the contents were written in Portuguese, 17% in English, 3% in German, 1% in Spanish, 1% in French and 1% in other languages. According to O'Neill *et al.* (2003), on the global web 72% of the pages are written in English and only 2% are written in Portuguese. Identifying the language of a content is sometimes a hard task, because there are contents with short text or written in several languages. The used language detector could not the language in 4% of the contents.

3.3.2.5 Meta-tags

There are two widely used meta-tags supported by HTML: *description* and *keywords* (W3C, 1999). The description meta-tag provides a description of the page's information and the meta-tag keywords provides a set of keywords that search engines may present as a result of a search. However, just 17% of the Portuguese

pages had the meta-tag description and, among these, the usage of this meta-tag does not seem to be correct. There were only 44 000 distinct description texts for 555 000 description meta-tags. This means that 92% of the texts of the descriptions were repeated elsewhere. There are several causes for this situation:

- The meta tag value is a default text inserted by a publishing tool;
- The publisher repeated the same text in all the pages of its site, although they are different;
- There are duplicates on the web.

The keywords meta-tag is present in 18% of the pages. A deeper analysis revealed that 91% of the pages that have the description meta-tag also had the keywords meta-tag. O'Neill *et al.* (2003) showed that the usage of meta-tags on the global web has being increasing. In 2002, 70% of the pages included meta-data. Although the presented results on the Portuguese Web focus only on two meta-tags, they suggest that the usage of meta-tags on the Portuguese Web is much less frequent than on the global web.

The titles of the pages are not very descriptive either. There were over 600 000 distinct titles for 3.1 million pages. The main reason for this observation is that the title of the home page is used as the title for all the pages in the site.

3.3.3 Web structure

This section presents measurements on the duplication, link structure and popularity of sites and contents on the Portuguese Web.

3.3.3.1 Duplication

Each content was identified by its MD5 digest to detect duplication (Rivest, 1992). There were 2 734 942 distinct contents and 15.5% of the URLs referenced duplicates. Kelly & Mogul (2002) identified only 5% of duplicates when analyzing a client trace from WebTV, but they used a different sampling methodology in their experience that may have be responsible for the difference between the results.

Number of duplicates	Number of Contents	% of contents
0	2 462 490	90.0
1	205 882	7.5
2	33 468	1.2
3	12 814	0.5
4	6 086	0.2
5	5 272	0.2
6-10	6 453	0.2
11-100	2 318	0.1
101-1 000	154	0.0
>1 000	5	0.0
Total	2734942	100.0

Table 3.6: Distribution of contents with duplicates.

Table 3.6 presents the duplication distribution. Most of the contents (90%) are unique and 7.5% had exactly one duplicate. Contents replicated more than 1 000 times are very rare. However, they were the cause of 13 146 downloads for just five distinct contents. These five instances were all caused by mal-functioning web servers, which always returned the same error page for all the requests. These situations are pathological for crawlers and also tend to bias the collection statistics. Unfortunately, the measures adopted against these traps were not completely successful. When the crawler identified the trap due to the high number of duplicates within the site, it stopped inserting new links but it already had inserted numerous URLs to crawl.

The obtained statistics indicate that 42% of the duplicates were hosted on the same site; 60% of the duplicates were hosted on a different site; 2% of the duplicates were hosted both in the same site and in another site.

3.3.3.2 Link structure

This analysis focused on links between distinct sites. The links to URLs that evidenced that the referenced content was not of one of the accepted types were excluded. For instance, URLs where the file part has a .jpg extension were not considered.

Most of the pages (95%), did not link to another Portuguese site (Figure 3.12). On average, a Portuguese page has 0.23 links to contents on another site. This

	Nr. links	URL
1	3 540	cpan.dei.uc.pt/modules/00modlist.long.html
2	2 425	ftp.ist.utl.pt/pub/rfc/
3	2 309	homepage.oninet.pt/095mad/bookmarks_on_mypage.html
4	1 632	www.fis.uc.pt/bbsoft/bbhtm/mnusbib3.htm
5	1 621	cpan.dei.uc.pt/authors/00whois.html
6	1 532	www.fba.ul.pt/links4.html
7	1 458	boa.oa.pt/bbsoft2/bbhtm/mnusbib3.htm
8	1 346	www.esec-canecas.rcts.pt/Educacao/Escolas.htm
9	1 282	pisco.cii.fc.ul.pt/nobre/hyt/bookmarks.html
10	1 181	www.fpce.uc.pt/pessoais/rpaixao/9.htm

Table 3.7: The 10 contents with highest number of outgoing links.

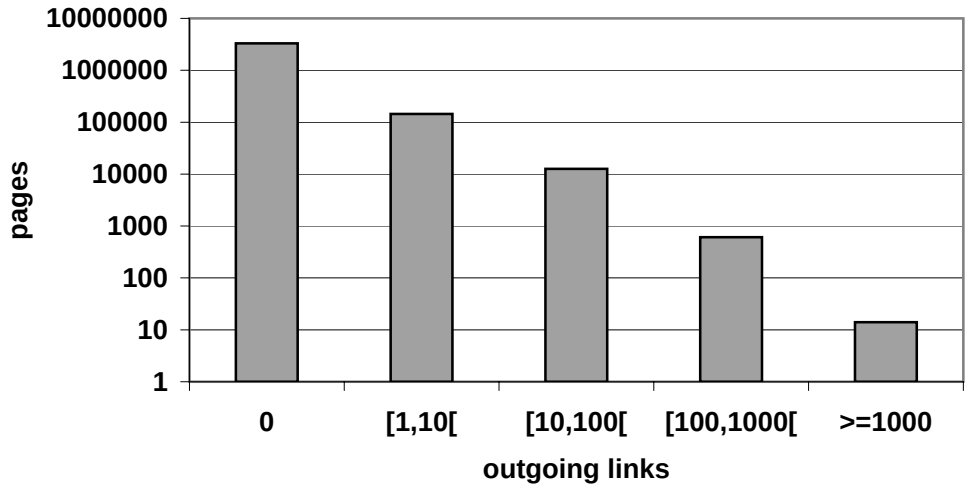


Figure 3.12: Distribution of the number of outgoing links per page.

Rank	Nr. of incoming links	Content URL
1	6 862	www.fccn.pt/
2	754	clanhosted.clix.pt
3	688	www.sapo.pt
4	606	www.publico.pt
5	522	www.infocid.pt
6	448	paginasbrancas.pt
7	423	www.dn.pt
8	413	www.sapo.pt/
9	361	security.vianetworks.pt
10	350	www.uminho.pt

Table 3.8: The 10 URLs with highest number of incoming links.

is not surprising, since most links in pages are navigational (Koehler, 2002). Nonetheless, there are also pages rich in outgoing links (see Table 3.7). The obtained results showed that 66% of the links in Portuguese pages did not point to another Portuguese site. This measure was made under the assumption that all the URLs hosted outside the .PT domain for which the language could not be determined were considered as being outside the Portuguese Web. 40% of the outgoing links pointed to home pages.

Figure 3.13 shows that 3 189 710 Portuguese contents (89%), were not referenced by a link originated in another Portuguese site. As observed on the global web, most links tend to point to a small set of pages (Kleinberg, 1999).

3.3.3.3 Content popularity

The importance of a content can be determined through the analysis of the web graph. In order to achieve a meaningful ranking of the relative importance of contents, links to duplicates and HTTP redirects were handled differently:

- Links to duplicates cause the splitting of the number of links to a content among the several URLs that refer it. In the presence of duplicated contents, the content with the smallest URL was elected as the common reference. Then, the links to duplicates were removed from the web graph and the correspondent links were re-targeted to the URL used as common reference;

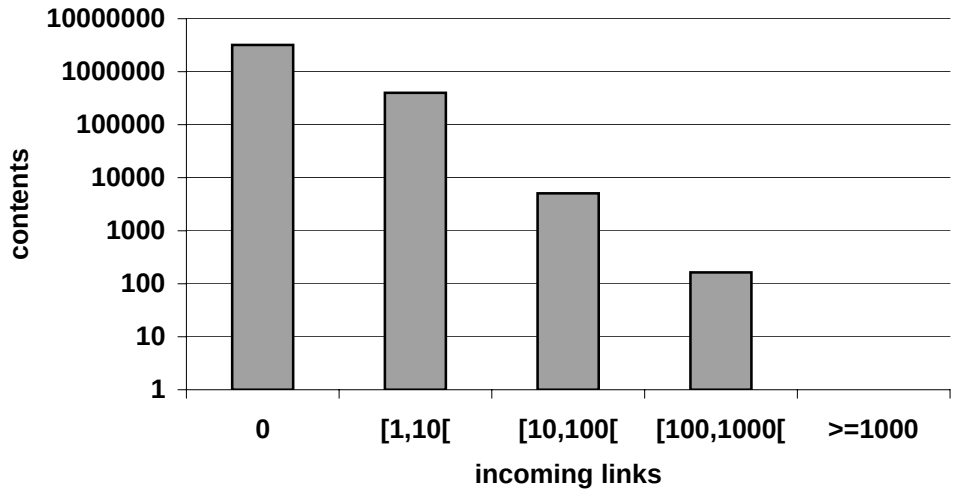


Figure 3.13: Distribution of the number of incoming links per content.

- HTTP redirects are almost invisible to web surfers. Involuntarily, publishers link to the URL of the redirect instead of the URL of the content. This causes a split in the number of links between the redirects and the content. Each redirect was followed until a non-redirect URL was found. Then, the redirect nodes were replaced in the graph by the correspondent non-redirect URLs.

Table 3.8 presents the URLs of the 10 contents that received most incoming links after the above modifications were applied.

Despite the efforts to eliminate pathological situations in the analyzed graph, there were still anomalies on the most ranked content lists. In positions 3 and 8 of Table 3.8, the number of incoming links was spread among two different URLs, although they both refer to the same content. The problem was that the two URLs were identified by their string representation and between the crawl of the first and the second URL, the content referenced by them changed. Sometimes the change on the content is very small. In the second example, the change was just a link to an advertisement.

	Nr. incoming links	Site
1	7 109	www.fccn.pt
2	1 881	br.weather.com
3	1 617	images.clix.pt
4	1 601	www.sapo.pt
5	1 481	www.clinicaviva.pt
6	794	www.depp.msst.gov.pt
7	777	www.infocid.pt
8	721	www.fct.mct.pt
9	652	ultimahora.publico.pt
10	615	www.miau.pt

Table 3.9: The 10 sites that received most incoming links.

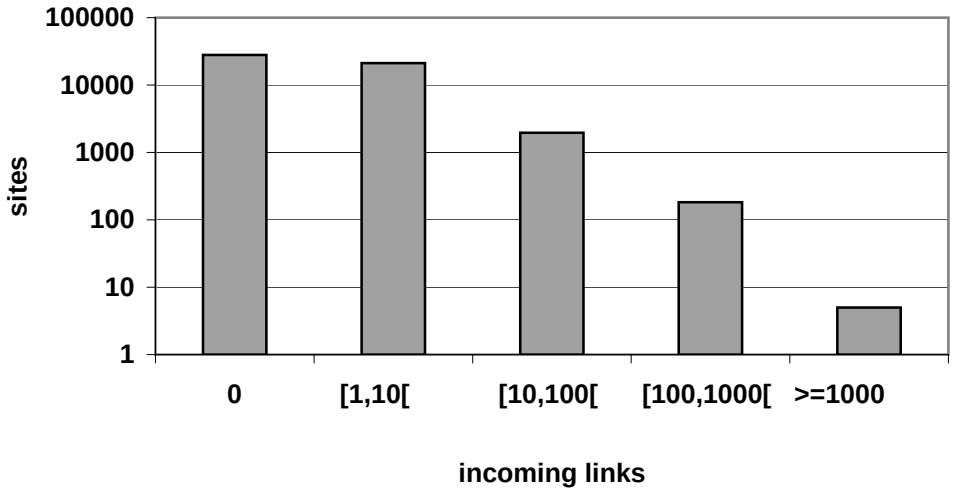


Figure 3.14: Distribution of the number of incoming links per site.

	Nr. users	Site
1	779 000	www.sapo.pt
2	580 000	www.microsoft.com
3	560 000	pesquisa.sapo.pt
4	548 000	loginnet.passport.com
5	540 000	www.clix.pt
6	538 000	www.google.pt
7	480 000	www.geocities.com
8	477 000	login.passport.net
9	471 000	www.terravista.pt
10	463 000	www.iol.pt
11	408 000	windowsupdate.microsoft.com
12	405 000	v4.windowsupdate.microsoft.com
13	380 000	www.msn.com
14	311 000	pesquisa.clix.pt
15	290 000	ww2.hpg.ig.com.br
16	247 000	webmail.iol.pt
17	244 000	www.mytmn.pt
18	227 000	webmail.sapo.pt
19	224 000	www.aeiou.pt
20	223 000	www.google.com
21	219 000	www.cidadebcp.pt
22	215 000	planeta.clix.pt
23	203 000	www.yahoo.com
24	202 000	webmail.clix.pt
25	193 000	caixadirecta.cgd.pt
26	191 000	netcabo.sapo.pt
27	189 000	dossieriraque.clix.pt
28	185 000	tsf.sapo.pt
29	185 000	www.cgd.pt
30	182 000	login.passport.com
31	181 000	www.msn.com.br
32	180 000	bandalarga.netcabo.pt
33	177 000	www.dgci.gov.pt
34	173 000	www.abola.pt
35	171 000	auth.clix.pt
36	165 000	pwp.netcabo.pt
37	159 000	www.tvi.iol.pt
38	156 000	netbi.sapo.pt
39	156 000	www.record.pt
40	156 000	download.com.com

Table 3.10: The 40 most accessed sites by Portuguese users between 2002 and 2003 (source: Marktest LDA).

3.3.3.4 Site popularity

The importance of a site can be derived from the total number of incoming links that it receives from other sites (Wu & Aberer, 2004). An highly ranked site might not host highly ranked contents. For instance, some online newspapers receive a large number of incoming links to many distinct news pages, but as news are interesting and are in many cases available for only a short period of time, they never get to be highly ranked contents.

Broder *et al.* (2000) analyzed the graph structure of the web through two large crawls of 200 million pages each. They considered each page as a node and each hypertext link as an edge on the graph. They found that 91% of the pages were reachable from one another by following either forward or backward links after computing an algorithm that finds weak components in the graph.

The results obtained for the Portuguese Web were obtained through a different methodology, because only the links between distinct sites were considered and weak components were not detected in the graph. Each Portuguese site is a node and each link between contents on two different sites an edge. 73% of the sites were connected to another site, by following links in both directions. This result contrasts with the one obtained by Broder *et al.* (2000), showing that the connectivity of the graph may decrease on a smaller portion of the web, such as the Portuguese Web.

Figure 3.14 presents the distribution of the number of incoming links per site. Considering the graph directed, with links followed only in their real direction, only 45% of the sites were reachable from another site, which leaves a majority of sites (55%) that are never linked (orphan sites). These sites were found by the crawler because they were part of the seeds.

Table 3.9 presents the 10 Portuguese sites that received most incoming links. These results were compared with a list of 495 selected sites, accessed from the homes of a panel of Portuguese users, during the period when the crawl was performed (Marktest, 2003). 50% of all the sites accessed by the panel of Portuguese users were part of the Portuguese Web. There is correlation of 0.527 between the number of users and links to the Portuguese sites. This shows that the most linked sites are also usually more visited by the users of this community web.

Table 3.10 presents the 40 sites that received most distinct users. The majority (27) of these popular sites are hosted under the .PT domain. There was a high number of accesses to sites that are automatically contacted by tools. For instance, when users type URLs of sites that are not found, Internet Explorer automatically redirects their requests to auto.search.msn.com by default. These sites appear as overrated in usage statistics.

At first sight, it is surprising that the site www.fccn.pt, which occupies the first position in Table 3.9, is not present in the list of the 495 sites accessed by the users. A deeper analysis revealed that 96% of the links to the FCCN (National Foundation for Scientific Computing) site were originated on sites hosted under the .RCTS.PT domain and almost all of them (99%) pointed to the FCCN home page (www.fccn.pt/). The RCTS network (Network for Science, Technology and Society) is also managed by FCCN. It is composed by over 11 000 sites from several public institutions, specially schools, hosted under the .RCTS.PT domain. FCCN automatically generated a site on the RCTS network for every school in the country, initially composed by a single page containing its address, e-mail and, a link to www.fccn.pt/. The content of these sites was supposed to be replaced by contents produced by the schools, but in most cases that didn't happen. As a result, the default site prevailed, generating a high number of links to the FCCN site from these sites.

3.4 Conclusions

This chapter presented a study on the characterization of the sites, contents and link structure of the Portuguese Web. It proposed a selection criteria that covers this web with high precision and is simultaneously easy to configure as a crawling policy.

This chapter proposed solutions for the situations on the web that may bias web statistics. This shows that a web characterization depends on the used crawling technology and configuration. The interpretation of statistics of data gathered from a national web is beyond a mathematical analysis. It requires knowledge about web technology and social reality of a national community.

Studies like this are interesting to others who need to characterize community webs and may help in the design of software systems that operate over the web. Web archivers can better estimate necessary resources and delimit portions interesting for archival. Web proxies can be more accurately configured by administrators. Crawlers can be improved through suitable architectures and configurations. Finally, web search engines that index data from optimized crawlers can improve their coverage of the web leading to better search results.

A question that was not completely answered is how different is the Portuguese Web from the global web. The obtained results suggest that the Portuguese Web has similarities and differences with other portions of the web. However, these differences may have been caused by the distinct research methodologies used in previous works, which often are not detailed enough. Moreover, because the web is permanently evolving, the differences may result from samples being gathered in different periods of time. One way to answer this question would be to crawl the Portuguese Web and the general web simultaneously using the same crawler configuration. Unfortunately, the resources to execute such an experiment were not available during the period of this research.

Although the characterization of the Portuguese Web, may not be representative of other web portions, it is representative of the data that would populate a WWh of the Portuguese Web because the extraction stage of the web data integration process was reproduced on the sampling methodology.

One snapshot is not enough to model all the characteristics of the web. Some of them must be analyzed through periodic sampling. The next chapter discusses characteristics of the Portuguese Web derived from several snapshots gathered for three years.

Chapter 4

Web data persistence

The information available on the web is permanently changing. Despite the ephemeral nature of web data, there is persistent information that prevails for long periods of time. Models of web data persistence are essential tools for the design of efficient Web Warehouses that repeatedly collect and process web data.

The lack of models and up-to-date characterizations of the web frequently postpone important design decisions for a late development stage. For instance, a Web Warehouse (WWh) could be designed to use a delta storage mechanism in order to save on storage space (Gomes *et al.*, 2006b). However, its efficiency would be jeopardized in practice because delta storage mechanisms are built on the assumption of persistency of object identifiers and do not cope with web contents identified by short life URLs.

This chapter models the persistence of information on the web using two metrics: the persistence of URLs and the persistence of contents. These metrics cannot be modelled through the analysis of a single snapshot of the web. So, several snapshots of a national web (the Portuguese Web) gathered for three years were studied. Besides these models, the study provides updated statistics on technological and structural characteristics of the web, and a characterization of persistent information.

This chapter is organized as follows: the next section describes the experimental setup used in this research work. Section 4.2 models the persistence of URLs and Section 4.3 the persistence of contents. Section 4.4 analyzes the re-

Crawl id.	Median date	Size (GB)	Nr. of URLs (millions)	Nr. of sites
1	2002-11-06	44	1.2	19 721
2	2003-04-07	129	3.5	51 208
3	2003-12-20	120	3.3	66 370
4	2004-07-06	170	4.4	75 367
5	2005-04-12	259	9.4	83 925
6	2005-05-28	212	7.3	81 294
7	2005-06-18	288	10	94 393
8	2005-07-21	299	10.2	106 841

Table 4.1: Statistics of the crawls in the data set.

lation between these two metrics and Section 4.5 draws the conclusions of this evaluation.

4.1 Experimental setup

Web data persistence was modelled through the analysis of eight crawls of the Portuguese Web gathered between 2002 and 2005. Table 4.1 presents the median date of harvesting of the pages, the total size of the downloaded contents, the number of URLs and sites successfully visited.

Each new harvest of a crawl was seeded with the home pages of the sites successfully harvested in the previous one. The crawler iteratively downloaded and followed links to URLs, visiting at most once each one of them. Ideally, the crawls should be successively larger, tracking web growth. However, Crawl 6 was stopped before it was finished due to hardware problems. The pairs of crawls that did not present an increasing number of contents were excluded from the analysis.

Robustness measures against hazardous situations for harvesting, such as spider traps, were imposed. The crawler harvested at most 5 000 URLs per site, following links from the seeds until a maximum depth of five in a breadth-first mode. The content sizes were limited to 2 MB and had to be downloaded within one minute. The length of the URLs was limited to a maximum of 200 characters. The crawls included contents from several media types convertible to text.

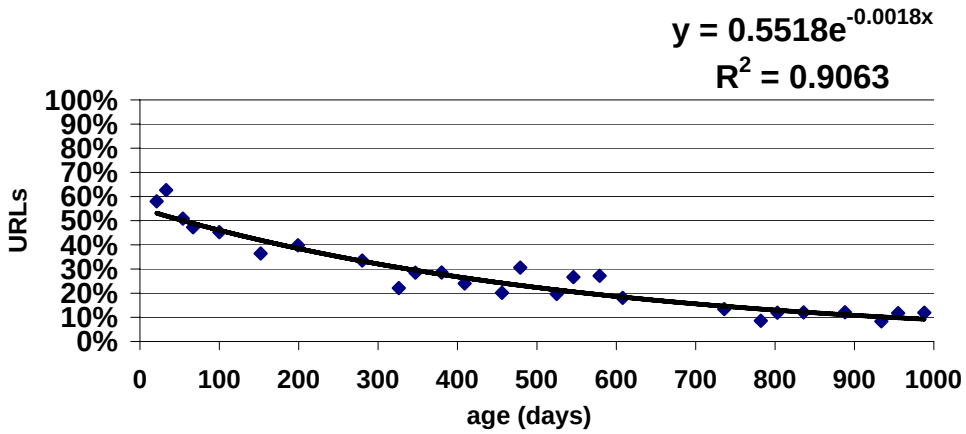


Figure 4.1: Lifetime of URLs.

97% of the contents were HTML pages, which is not surprising since this is the dominant textual format on the web (Heydon & Najork, 1999).

4.2 URL persistence

The URLs are the identifiers of the resources available on the web and the basis of its structure. Hence, a WWh must be designed to manage URLs efficiently. A model for predicting the lifetime of URLs enables the definition of data structures and algorithms to manage them at an early stage of the system's development.

4.2.1 Lifetime of URLs

There are several situations that lead to the bulk disappearance of URLs: webmasters migrate their servers to different technological platforms, entire sites are shut down and session identifiers generate new URLs for each visit.

An URL was considered persistent if the referenced content was successfully downloaded in two or more crawls, independently from content changes. The URLs that were not linked from any page could not be found by the crawler and would hardly be found by a web user through navigation. Thus, they were considered dead.

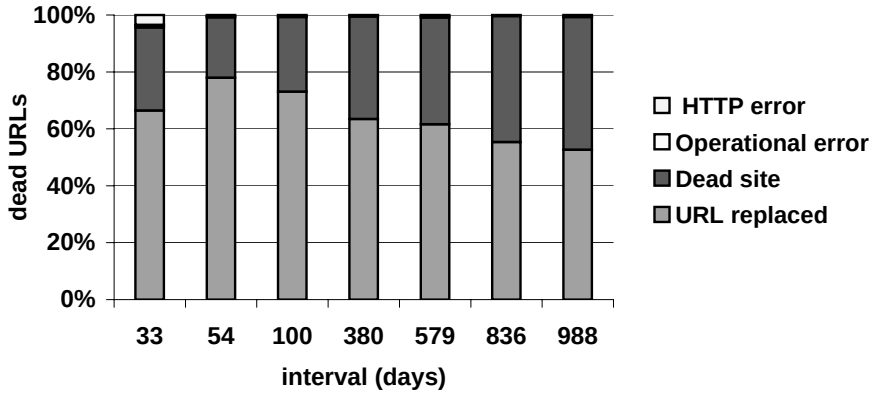


Figure 4.2: Reasons for URL death.

Figure 4.1 shows the relation between the percentage of persistent URLs and their age. The age for a persistent URL found in each pair of crawls is given by the difference in days between the median dates of the crawls. For instance, Crawl 1 was executed in November 2002 and Crawl 3 was executed in December 2003. Hence, the URLs of Crawl 1 that persisted until Crawl 3 were 409 days old and 24% of these URLs persisted between the two crawls. Most URLs have short lives and the death rate is higher in the first months. However, a minority of URLs persists for long periods of time. The lifetime of URLs follows a exponential function trend line with an R-squared value of 0.9063. The function estimates the probability of an URL being available given its age. The half-life of URLs is the time that it takes to 50% of the URLs in a data set to die. The obtained results showed that the half-life of URLs is 60 days.

A previously proposed model to estimate the frequency of change of pages assumed that URLs persisted in time as identifers (Cho & Garcia-Molina, 2003). The model for estimating URL persistence presented in this section complements that work by estimating the time span under which the assumption is valid.

4.2.1.1 URL death

An URL was considered dead if it was not referencing a content in the last crawl (8^{th}), but was successfully harvested previously. A site was considered dead if it

did not provide at least one content.

Figure 4.2 presents the main reasons found for URL death. The *xx* axis represents the time elapsed in days between the pairs of crawls analyzed. Considering an interval of 54 days between crawls. For 78% of the dead URLs, the corresponding site was alive but did not link to them. This suggests that the URLs were replaced by new ones. For 21% of the dead URLs, the corresponding sites were also found dead. The percentage of URL deaths due to site's disappearance increased with time.

While harvesting, operational problems like network failures may occur. On average, only 0.4% of the URLs were considered dead due to these problems, most of them because the referenced content could not be downloaded within one minute. URL unavailability identified through HTTP errors represents on average 0.8% of the causes of URL death, but these errors become more visible in shorter intervals, 3.5% of the URLs in Crawl 7 presented HTTP errors in Crawl 8 (33 days of interval). The most common HTTP errors were *File Not Found* (404), *Internal Server Error* (500) and permanent or temporary redirections (301, 302). Notice that the crawler also visited the target URLs of the redirections. Spinellis (2003) studied the reasons of death among the URLs cited from research articles and witnessed similar results.

The obtained results suggest that the main causes of URL death are the frequent replacement of URLs and site death, independently from the source of citation.

4.2.1.2 Lifetime of sites

The lifetime of sites was also studied. For each crawl, the percentage of sites that were still alive in subsequent crawls was identified. The age of a site is the difference between the dates of the visits to obtain the crawls. Figure 4.3 shows that over 90% of the sites younger than 100 days were alive, but this percentage decreased to 30-40% among those older than 700 days.

The half-life of sites is 556 days, which is significantly larger than the half-life of URLs. Hence, a WWh can be designed to reuse information about a site although their URLs may disappear. For instance, consider a site that migrates

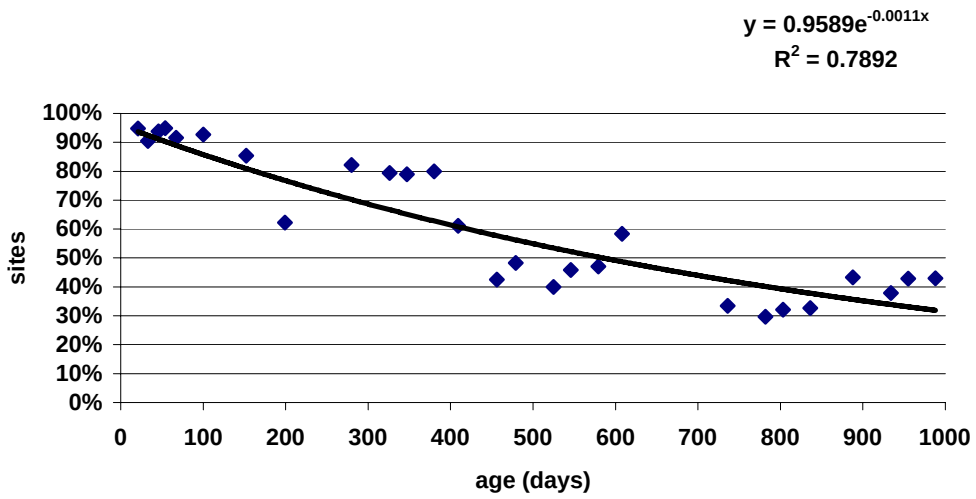


Figure 4.3: Lifetime of sites.

to a new content management system, causing the replacement of most of its URLs. The information kept in a WWh about the site, such as the description of its content, does not need to be updated, although most of the previous URLs of the site no longer reference contents.

4.2.2 Characteristics of persistent URLs

The characteristics of an URL can predict its persistence. Crawl 8 was used as baseline to characterize persistent URLs in the evaluated data set. The URLs in the baseline that persisted from previous crawls were identified and their feature distributions were compared. The age of an URL is the difference in days between the date of the crawl and the date of the baseline (which is 0 days old).

4.2.2.1 Dynamic URLs

URLs containing embedded parameters are commonly generated on-the-fly by the referrer page to contain application specific information (e.g. session identifiers). These URLs are frequently used just once. The URLs containing embedded parameters are *dynamic* and the remaining are *static* (Figure 4.4).

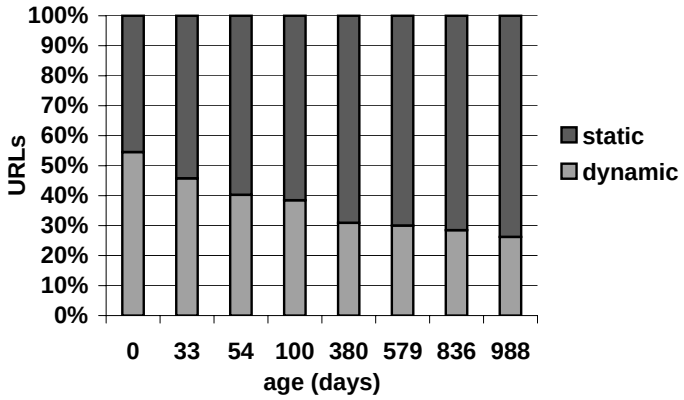


Figure 4.4: Distribution of dynamic URLs.

The URLs were extracted from links in pages, so dynamic URLs resultant from the input of values in forms were not considered. The first column identified with age 0 shows that 55% of the URLs in the baseline were dynamic and 45% were static. The second column presents the distribution of static and dynamic URLs that persisted from Crawl 7 until the baseline. As the date of the baseline was 2005-07-21 and the date of Crawl 7 was 2005-06-18, the persistent URLs have an age of 33 days. The presence of dynamic URLs decreases smoothly as they grow older: 46% of the URLs 33 days old were dynamic, but this percentage decreased to 26% among URLs 988 days old.

The obtained results show that static URLs are more persistent than dynamic URLs, although there are dynamic URLs that persist for years.

4.2.2.2 URL length

Figure 4.5 presents the relation between the length and the persistence of URLs. It shows that URLs shorter than 50 characters are more persistent than longer ones. This observation is consistent with the results presented on the previous Section, because dynamic URLs were longer (average 77.1 characters) and less persistent than static URLs (average 49.2 characters). An analysis of a sample of these URLs revealed that very long URLs tend to be used in poorly designed sites that are quickly remodelled or deactivated.

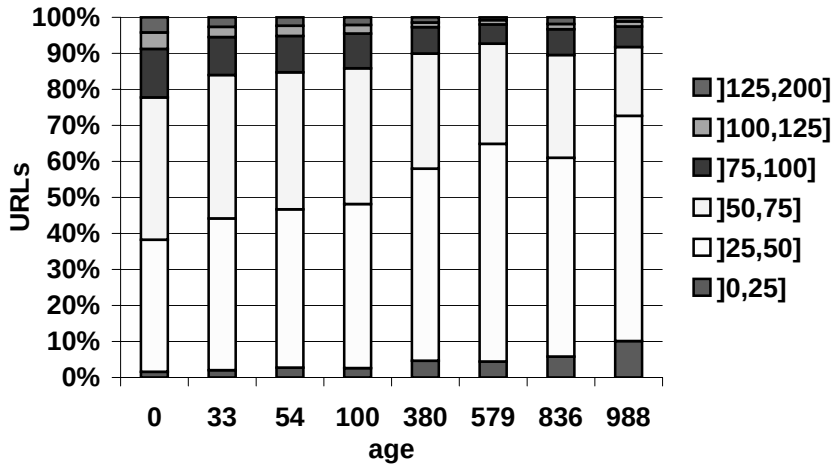


Figure 4.5: Distribution of URL length (number of characters).

4.2.2.3 Depth

The depth of an URL is the minimum number of links followed from the home page of the site to the URL. The URLs at lower depths are usually the most visited. One may hypothesize that they should be more persistent because broken links are easier to detect. However, Figure 4.6 describes the distribution of the URLs per depth and shows that depth did not influence URL persistence.

An analysis of these persistent URLs revealed that they can be found at different levels of depth according to the structure of the site. There are sites presenting a deep tree structure, while others have a shallow and wide structure. So, an URL with $depth = 3$ may be deep in one site but not in another.

4.2.2.4 Links

Authors use links to reference information related to their publications. The number of links that an URL receives from external sites represents a metric of importance, while links internal to the site are navigational.

Figure 4.7 describes the distribution of the URLs that received at least one link from another site. 98.5% of the URLs in the baseline did not receive any link. However, the presence of linked URLs among persistent URLs slightly increased

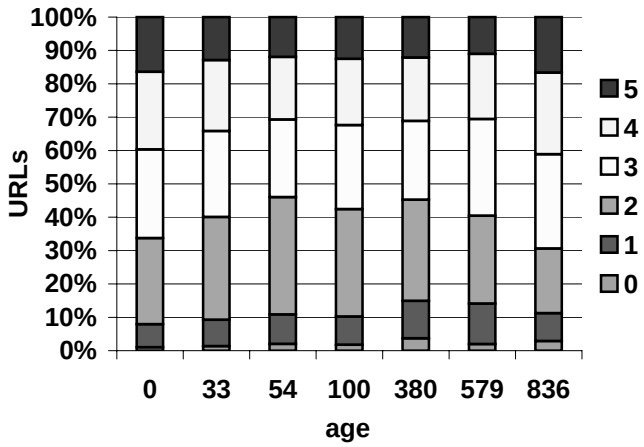


Figure 4.6: Distribution of URL depths.

with time. It raised from 1.5% among URLs aged 33 days to 9.6% among URLs 988 days old. There are two possible explanations for this fact. First, persistent URLs are more likely to accumulate links during their lifetime. Second, the number of links to an URL increases its measure of popularity in search engines and the owners of popular URLs take special care in preserving them because of their commercial value.

4.3 Lifetime of contents

A WWh needs to periodically update the information collected from the web. A model that determines the lifetime of a content enables measuring the freshness of the information kept and the schedule of refreshment operations.

Fetterly *et al.* (2003) observed a set of pages for 11 weeks and observed that the age of a content is a good predictor of its future persistence. This section presents a study on content persistence for a longer period of time.

The definition of a boundary for deciding if a content has changed enough to imply its refreshment from the web is highly subjective. It was assumed that any change in a page generates a new content to study the persistence of contents on the web. Contents were identified by comparing their fingerprints

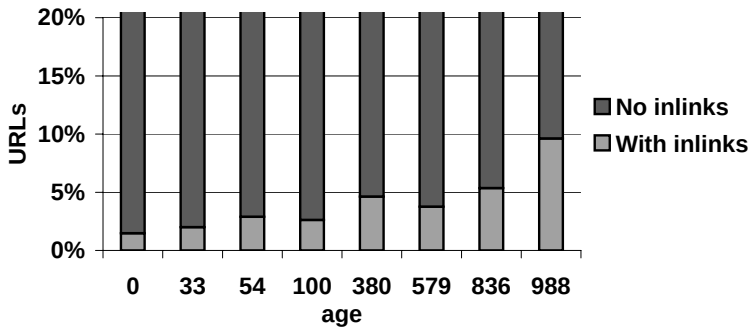


Figure 4.7: Distribution of linked URLs.

between crawls, independently from the URLs that referenced them. For each crawl, the percentage of contents that were still available on the following ones was computed.

Figure 4.8 summarizes the percentages of persistent contents according to their age. Just 34% of the contents 33 days old persisted, but 13% of the contents lived approximately one year. The lifetime of contents matches a logarithmic function with an R-squared value of 0.8475. This function enables the estimation of the contents percentage that remain unchanged within a collection of pages given its age. The half-life of contents is the time that it takes to 50% of the contents in a data set to change or disappear. The obtained results suggest that the half-life of contents is two days. Most contents have short lives, but there is a small percentage that persists for long periods of time.

4.3.1 Characteristics of persistent contents

The characteristics of contents influence their persistence. Modelling persistent contents enables the design of web warehouses, which may be tuned to the characteristics of the data collections processed. For instance, a model on the persistence of web data can be used to tune the refresh policy according to the lifetime and characteristics of the cached information.

This section presents an analysis of five features that characterize persistent contents. Crawl 8 was the baseline to derive feature distributions. Each of these

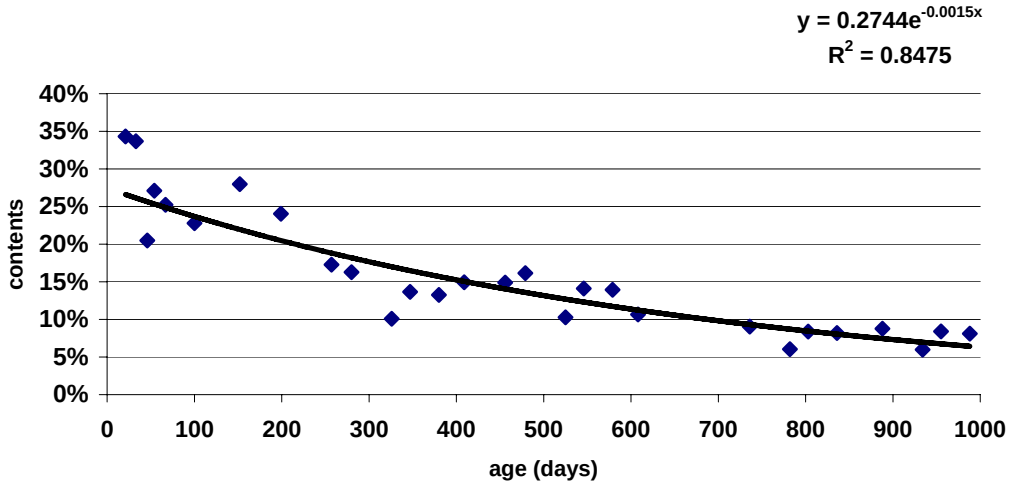


Figure 4.8: Lifetime of contents.

features will now be dissected.

4.3.1.1 Dynamic contents

Dynamic contents are generated on-the-fly when a web server receives a request. They became popular because they enable the efficient management of instance data in databases, independently from publishing formats (e.g. online shops).

A WWh can benefit from applying different storage policies for static and dynamically generated contents gathered from the web. The URLs containing embedded parameters referenced dynamic contents and the remaining were static.

Figure 4.9 shows that 55% of the contents in the baseline, harvested in July, 2005, were dynamically generated, a figure superior to the 34% witnessed by Castillo (2004) in May, 2004. The presence of dynamic contents decreased to 32% among contents 33 days old and to less than 9% among contents older than 579 days. The conclusion derived from the results is that in the long term the static contents are more persistent than dynamic contents.

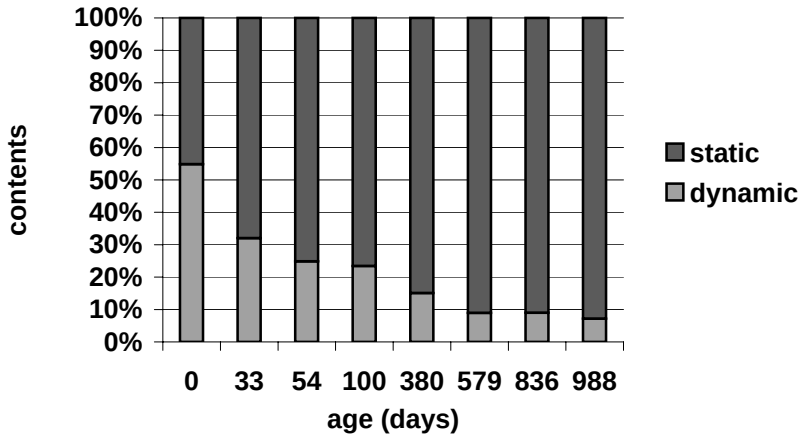


Figure 4.9: Distribution of dynamically generated contents.

Author	Date of the sample	%Unknown
Douglis *et al.* (1997)	1997-?	21%
Brewington & Cybenko (2000)	1999-03	35%
Bent *et al.* (2004)	2003-07	44%
Baseline	2005-07	64%

Table 4.2: Evolution of the presence of unknown dates in the Last-Modified header field.

4.3.1.2 Last-Modified date

The Last-Modified date allows to detect if a content has changed since a previous visit without having to download it. This meta-data can be used to implement refresh policies in web warehouses. For instance, cache entries can be invalidated based on this information. Webmasters are encouraged to disable the Last-Modified field for pages that change frequently (The Apache Software Foundation, 2004). So, the simple presence of this information for a content can be an indicator of its persistence.

Figure 4.10 shows that the contents with an associated Last-modified date are significantly more persistent than those with an unknown date of last modification. Web servers returned unknown values for 64% of the contents in the

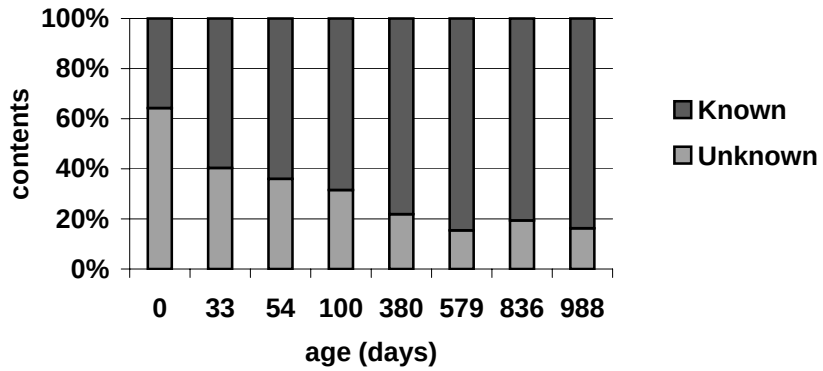


Figure 4.10: Distribution of contents with known Last-Modified dates.

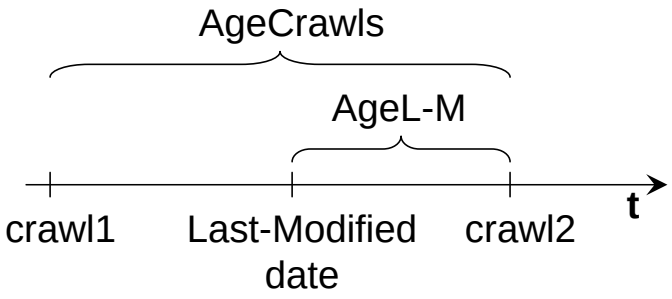


Figure 4.11: Content age calculated through Last-Modified and crawl dates.

baseline. Table 4.2 presents the results obtained in previous works and shows that the usage of the Last-modified header field tends to decrease.

Web warehouses should not rely blindly on the Last-Modified date because it can provide erroneous values. The web server's clock may not be correctly set or the file date may have been updated with no associated change to its content. The ages of the contents derived from the Last-Modified header field were compared with the ages calculated from the dates of harvest to measure the presence of Last-Modified dates that underestimated the longevity of contents on the web (see Figure 4.11).

The line connecting the squares in Figure 4.12 shows that the number of contents older than the Last-Modified date increased with age. One reason for this

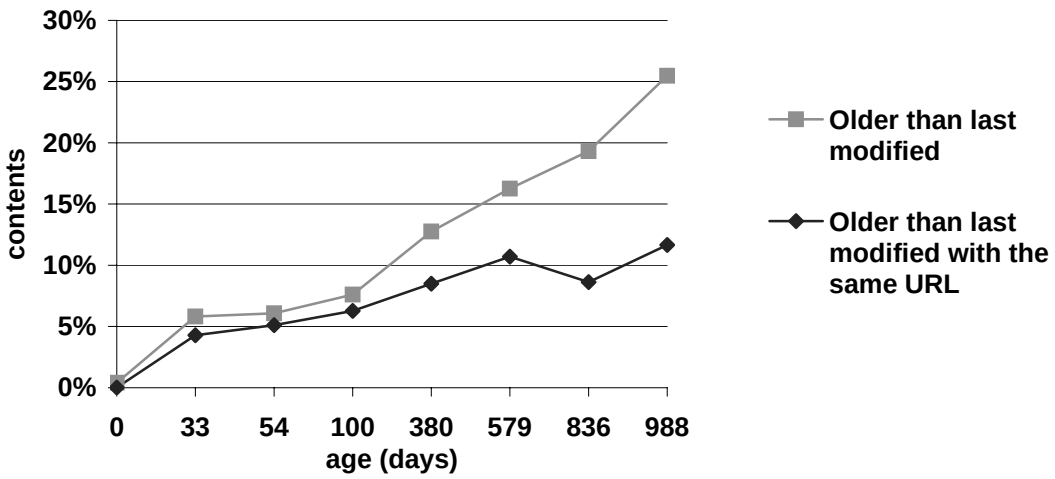


Figure 4.12: Contents that present underestimated ages due to erroneous Last-Modified dates.

result might be that older contents are more liable to experience site reorganizations that move them to different locations and update timestamps without causing changes in the contents. These operations commonly cause changes in the URLs. Hence, the ages were recomputed for the contents that maintained the same URL (line with triangles) and the number of erroneous Last-Modified dates dropped significantly for contents older than 100 days.

The obtained results show that contents with an associated Last-Modified date are more persistent. The presence of inaccurate Last-Modified dates increases among elder contents, but it is less visible among contents that maintain the same URL. Hence, a WWh should be designed assuming that the contents with a known Last-Modified date are more persistent, but the supplied date is not a reliable source of information for the elder contents.

4.3.1.3 Content length

Figure 4.13 presents the size distribution of contents. The presence of small contents increased with age, 27% of the contents in the baseline were smaller than 10 KB, but this percentage increased to 74% among the contents 988 days old.

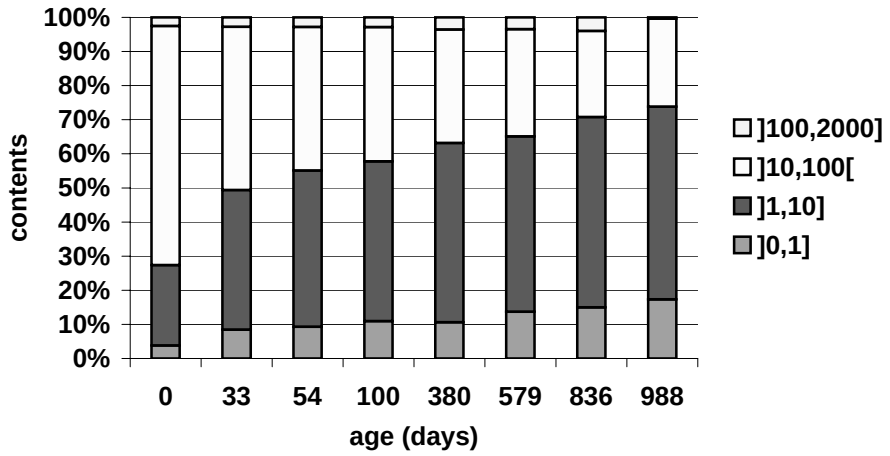


Figure 4.13: Distribution of content size (KB).

The obtained results show that small contents are more persistent than bigger ones. This conclusion is consistent with the observation by Fetterly *et al.* (2003) that large pages change more often than smaller ones.

4.3.1.4 Depth

The contents kept at low depths are the most reachable. Empirically, they should change often to include new advertisements or navigational links within the site. The contents kept deep in the sites are frequently archives of persistent information.

Figure 4.14 shows that the depth distribution is maintained regardless of the contents' age. Some sites change their contents frequently (e.g. online auctions), while others keep them unchanged (e.g. online digital libraries), regardless of depth. Thus, the obtained results show that depth is not a predictor of content persistence.

4.3.1.5 Site size

The size of a site is the number of contents that it hosts. One may argue that only large sites, such as digital archives, maintain contents online for long periods

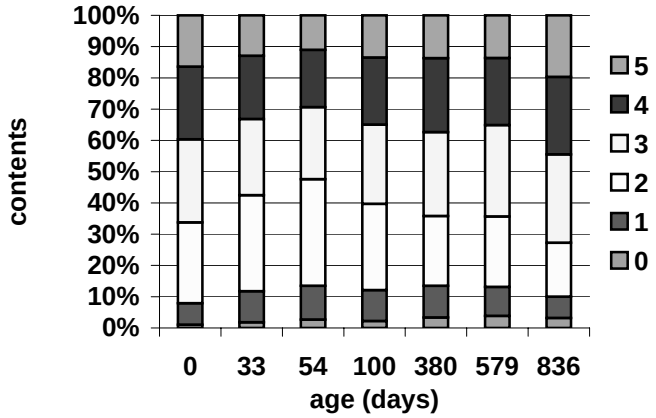


Figure 4.14: Distribution of content depth.

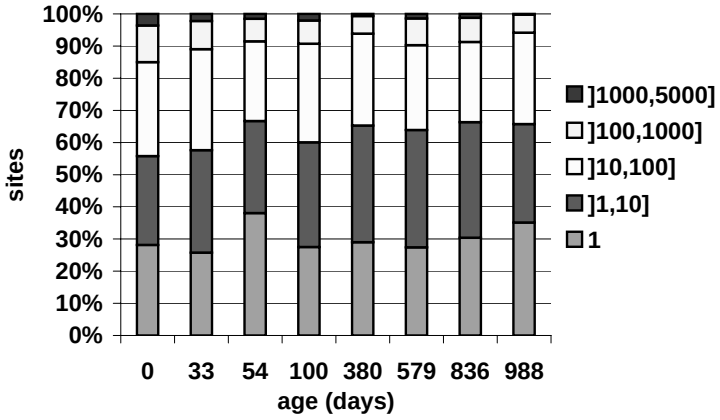


Figure 4.15: Distribution of site size.

of time. In this case, there would be a prevalence of large sites among persistent contents.

Figure 4.15 describes the distribution of the site sizes. The sites that hosted a single content were mainly home pages of sites under construction that were never finished. The percentage of sites that hold more than 100 persistent contents tends to slightly decrease with time but the general distribution of the site sizes does not significantly change. The conclusion is that the distribution of the number of persistent contents per site is similar to the one found on a snapshot of the web.

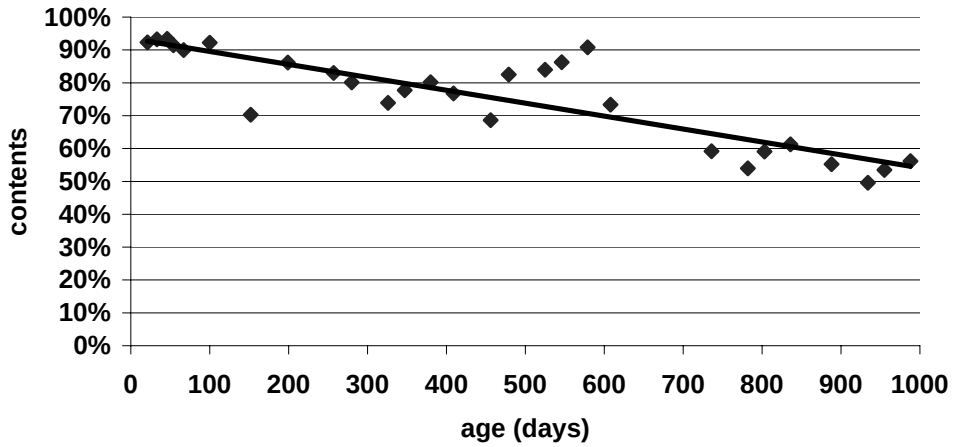


Figure 4.16: Persistent contents that maintained the same URL.

4.4 Relation between URL and content persistence

Previous work on the study of the evolution of the web focused on the change of contents under the same URL, assuming that the unavailability of an URL implied the death of the referenced content (Cho & Garcia-Molina, 2003; Fetterly *et al.*, 2003). However, a change of a site's name modifies all the correspondent URLs without implying changes on the referenced contents. Lawrence *et al.* (2000) witnessed that for almost all invalid URLs found in academic articles it was possible to locate the information in an alternate location. Figure 4.16 quantifies the presence of persistent contents that maintained the same URL. Over 90% of the persistent contents younger than 100 days maintained the same URL. However, this relation tends to decrease as contents grow older, on average only 58% of the contents older than 700 days maintained the same URL. These results show that the assumption that the death of an URL implies the death of the referenced content is inadequate in long-term analysis.

The permanent change of contents on the web may lead to believe that most URLs reference several different contents during their lifetime.

Figure 4.17 depicts the relation between persistent URLs and persistent contents. 55% of the URLs 33 days old referenced the same content during their

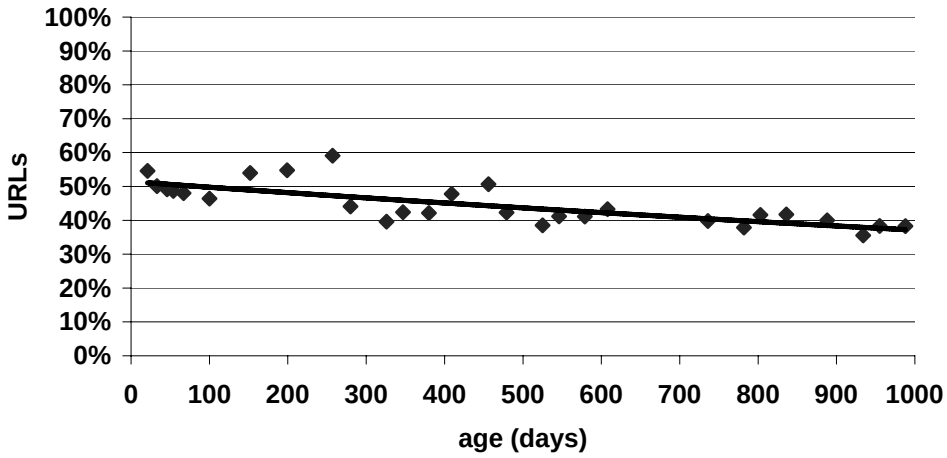


Figure 4.17: Persistent URLs that maintained the content.

lifetime. This percentage does not vary much as URLs grow older. The results show that persistent URLs tend to reference persistent contents.

4.5 Conclusions

Web data models help on important design decisions in the initial phases of WWh implementation projects. This chapter presented models for the persistence of information on the web, analyzing the lifetime of URLs and contents, and the characteristics of web data that influence them. The persistence of information on the web was studied through the analysis of a set of 51 million pages harvested from a national community web. These data differ from those in previous studies, as it was built from exhaustive harvests of a portion of the web during several years, regardless of page importance or the selection bias of documents kept by topic-specific digital libraries.

I believe that a collection generated by exhaustive harvests of a national community web is representative of the persistence of information on the general web, because it includes a broad scope of sites that represent distinct genres (e.g. blogs, news, commercial sites). However, a national community web may differ from the general web in other aspects, such as language usage.

Author	Age	URL persistence	Model estimation
Koehler (2002)	1.9 years	50%	17%
Cho & Garcia-Molina (2000a)	1 month	70%	60%
Fetterly *et al.* (2003)	2.8 months	88%	47%
Ntoulas *et al.* (2004)	1 year	20%	26%
Digital Libs.			
Spinellis (2003)	1 year	80%	26%
Markwell & Brooks (2003)	4.7 years	50%	5%
Lawrence *et al.* (2000)	1 year	77%	26%

Table 4.3: Comparison of URL persistence with the estimation derived from the obtained model.

Author	Age	Content persistence	Model estimation
Brewington & Cybenko (2000)	100 days	50%	23%
Cho & Garcia-Molina (2000a)	1 day	77%	55%
Fetterly *et al.* (2003)	7 days	65%	41%
Ntoulas *et al.* (2004)	1 year	10%	15%

Table 4.4: Comparison of content persistence with the estimation derived from the obtained model.

The lifetime of URLs and contents follows a exponential distribution. Most URLs have short lives and the death rate is higher in the first months but there is a minority that persists for long periods of time. The obtained half-life of URLs was two months and the main causes of death were the replacement of URLs and the deactivation of sites. Persistent URLs are static, short and tend to be linked from other sites and depth did not influence URL persistence.

Table 4.3 shows that the obtained results contrast with previous work and evidence a quicker decay of URLs. However, they strengthen conclusion from Koehler (2002) that once a collection has aged sufficiently, it becomes more durable in time.

The half-life of contents is just two days. The comparison of the obtained results with previous works suggests that the lifetime of contents is decreasing (see Table 4.4). Typically, persistent contents are not dynamically generated,

are small and have an associated Last-Modified date. However, the presence of contents with known Last-Modified dates has been decreasing in the past years, which is consistent with the conclusion that the lifetime of contents is decreasing. Inaccurate Last-Modified dates increased among elder contents but they were less visible among those that maintained the same URL. Persistent contents were not related to depth and were not particularly distributed among sites. About half of the persistent URLs referenced the same content during their lifetime.

The next chapter discusses how the derived models for the Portuguese Web influenced the design of the WWh that stores it.

Chapter 5

Designing a Web Warehouse

The Web is a powerful source of information, but its potential can only be harnessed with applications specialized in aiding web users. However, most of these applications cannot retrieve information from the web on-the-fly, because it takes too long to download the data. Pre-fetching the required information and storing it in a Web Warehouse (WWh) is a possible solution. This approach enables the reuse of the stored data by several applications and users.

A WWh architecture must be adaptable so it may closely track the evolution of the web, supporting distinct selection criteria and gathering methods. Meta-data must ensure the correct interpretation and preservation of the stored data. The storage space must accommodate the collected data and it should be accessible to humans and machines, supporting complementary access methods to fulfill the requirements of distinct usage contexts. These access methods should provide views on past states of the stored data to enable historical analysis.

This chapter discusses the design of a WWh prototype named Webhouse. The information harvested from the Portuguese Web and described on the previous Chapters was integrated in Webhouse while it was being developed to study the influence of web characteristics in the design of a WWh. Figure 5.1 presents the high-level architecture of Webhouse. There are two main components: the *Viúva Negra* crawler (VN) is responsible for extracting and loading contents into the *Versus repository*. The Versus repository provides high performance structured access to meta-data and extensible storage space to keep contents.

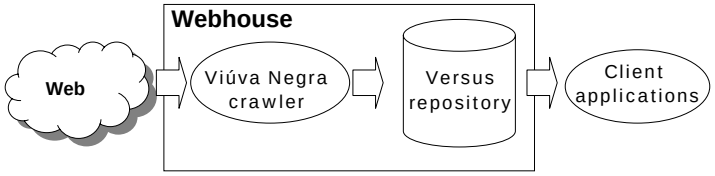


Figure 5.1: Webhouse architecture.

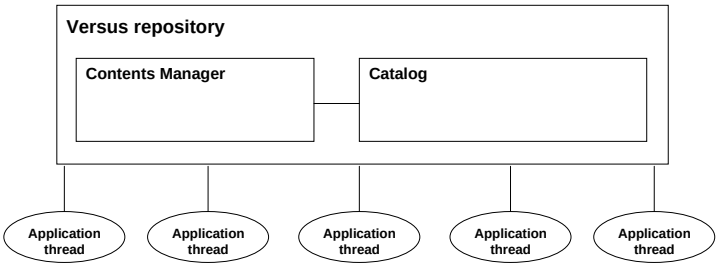


Figure 5.2: Versus architecture.

The focus of the chapter is on the design of the Webhouse prototype, discussing the extraction, loading and management of web data. It is organized as follows: Section 5.1 describes the Versus repository. It discusses the management of contents, addressing in particular the problem of duplication among web collections. It also presents the data and operational models that support the system, enabling the quick development of distributed client applications. Section 5.2 describes the design of the VN crawler. Section 5.3 discusses how some situations on the web can degrade the performance of a WWh. Finally, Section 5.4 presents the main conclusions derived from the development and operation of Webhouse.

5.1 The Versus repository

Figure 5.2 represents the Versus repository architecture. It is composed by the *Content Manager* and the *Catalog*. The Content Manager provides storage space for the contents (Gomes *et al.*, 2006b). The Catalog provides high performance access to structured meta-data. It keeps information about each content, such

as the date when it was collected or the reference to the location where it was stored in the Content Manager.

5.1.1 Content Manager

Web warehousing involves a large amount of data and claims for storage systems able to address the specific characteristics of web collections. The duplication of contents is prevalent in web collections. It is difficult to avoid downloading duplicates during the crawl of a large set of contents, because they are commonly referenced by distinct and apparently unrelated URLs (Bharat & Broder, 1999; Kelly & Mogul, 2002; Mogul, 1999a). Plus, the contents kept by a WWh have an additional number of duplicates, because it is built incrementally and many contents remain unchanged over time, being repeatedly stored (see Chapter 4). The elimination of duplicates at storage level after a crawl is terminated can be executed in batch by finding and deleting duplicates. However, disk I/O operations are expensive. First, every content is written to disk during the crawl, then read for comparison with others to identify duplicates and finally deleted, if it is a duplicate. Considering that a WWh may hold billions of contents and new ones are frequently being loaded, this approach can be prohibitively time consuming. The identification of duplicates is an appealing feature to identify replicas or save on storage space, but it must not jeopardize the WWh performance.

This section describes the design of the Versus Content Manager. It discusses the elimination of exact and partial duplicates and presents the mechanism implemented to detect if a content is a duplicate before it is stored on disk.

5.1.1.1 Elimination of partial duplicates in a WWh

Delta storage or encoding, is a technique used to save space that consists on storing only the difference from a previous version of a content (MacDonald, 1999). There are pages that suffer only minor changes, such as the number of visitors received or the current date. Delta storage enables storing only the part of the content that has changed, eliminating partial duplicates. Versioning systems, such as CVS (Berliner, 1990), assume that objects change in time maintaining a descendence tree under an unique identifier (file name). However, duplication of

pages referenced by different identifiers (URLs) is prevalent on the web and using delta encoding to store them causes continuous duplication in independent trees. Plus, each tree holds few versions of a content, because URLs are highly transient (see Chapter 4). A WWh may need to preserve contents for long periods of time.

Delta storage also raises preservation issues, because the retrieval of a content kept in delta format must be rebuilt by traversing the descendence tree. These algorithms are software dependent and a corruption on the tree may turn the subsequent versions of the content inaccessible. Moreover, delta storage algorithms can only process a limited set of content formats. A delta storage algorithm that processes an HTML file may not be effective when applied to a Macromedia Flash movie, so it is highly dependent on technological changes of the formats used for web publishing. Delta storage imposes expensive comparisons to detect partial duplications between a content in the descendence tree and the content to store. These comparisons executed while contents are being intensively loaded could originate a serious bottleneck in a WWh.

The elimination of partial duplicates to save storage space was analyzed in two studies by You & Karamanolis (2004) and Denehy & Hsu (2003). In the first study the authors compared intra-file compression against inter-file compression using delta encoding and data chunking. These techniques are used to save storage space within a set of contents by eliminating partial duplicates. However, they require additional meta-data to reconstruct the original contents, which may become a preservation problem if the meta-data is lost. The authors also evaluated several techniques with distinct data sets and concluded that none of them presented optimal results for all data sets. For the data set containing pages, compressing each page independently causes just an increase of 4% over the best result presented (using delta encoding).

In the second study, Denehy & Hsu (2003) evaluated different methods for detecting partial duplicates and proposed a system that stores unique blocks of data with a configurable replication level. The evaluations were mainly ran over an email corpus. Although emails contain attachments of formats commonly found on the web, inferring that the best method (sliding window) would be as efficient in a web corpus would be arguable.

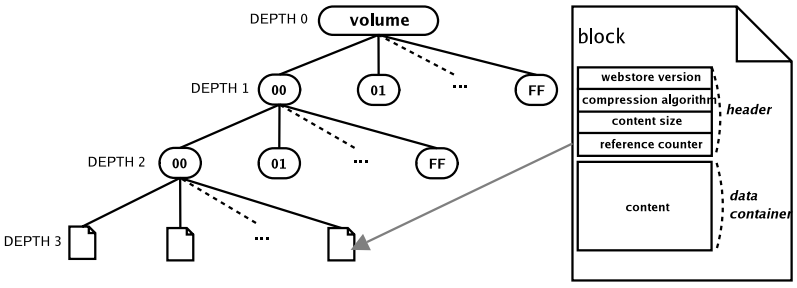


Figure 5.3: Storage structure of a volume: a tree holding blocks on the leafs.

Both studies suggested that eliminating partial duplicates in distributed systems would raise new problems. However, web warehouses require distributed storage systems to cope with the large amounts of web data.

In conclusion, eliminating partial duplicates in a WWh repository is not recommended because the mechanisms for eliminating them at the storage level cannot be efficiently applied in large-scale distributed storage systems. Nevertheless, eliminating partial duplicates could be useful to save space while storing small collections of contents harvested from the web within short intervals of time. Given the heterogeneity of the content formats found on the web, a WWh repository should also support independent compression algorithms according to the content's format.

5.1.1.2 Data model

The data model of the Webhouse Content Manager relies on three main classes: *instance*, *volume* and *block*. The instance class provides a centralized view of a storage space composed of volumes containing blocks. Each block keeps a content and related operational meta-data. The *signature* is the number obtained from applying a fingerprinting algorithm to the content. A *contentkey* contains the signature of the content and the volume where it was stored. A *block* holds an unique content within the volume. It is composed by a *header* and a *data container* (see Figure 5.3). The data container keeps the content. The header contains information about the software version, the content's original size in

bytes and a reference counter that keeps track of the difference between the storage and delete requests performed on the content, allowing independent clients to share the same instance without interfering with each other. The header also specifies the algorithm used to compress the content, enabling the coexistence of several compression types within the volume and the application of suitable algorithms according to the content's format. The storage structure of a volume is a tree containing blocks on its leafs. Figure 5.3 illustrates a storage structure with $depth = 3$. The nodes within each level of depth are identified by numbers represented in hexadecimal format from 0 to FF. The tree depth can change within the volumes that compose an instance, according to the storage capacity of the node.

5.1.1.3 An algorithm for eliminating duplicates

The location of a block within the volume tree is obtained by applying a function called *sig2location* to the content's signature. Assuming that the signature of a content is unique, two contents have the same location within a volume if they are duplicates. Consider a volume tree with depth n and a signature with m bytes of length. Sig2location uses the *(n - 1)* most significant bytes in the signature to identify the path to follow in the volume tree. The i^{th} byte of the signature identifies the tree node with depth i. The remaining bytes of the signature (*m-n-1*) identify the block name on the leaf of the tree. For instance, considering a volume tree with depth *3*, the block holding a content with signature *ADEE2232AF3A4355* would be found in the tree by following the nodes AD, EE and leaf 2232AF3A4355.

The detection of duplicates is performed during the storage of each content, ensuring that each distinct content is stored in a single block within the instance. When a client requests the storage of a content, the system performs a sequence of tasks:

1. Generate a signature s for the content;

2. Apply sig2location to the signature and obtain the location l of the corresponding block;

3. Search for a block in location l within the n volumes that compose the instance, multicasting requests to the volumes;

4. If a block is found in volume, the content is considered to be a duplicate and its reference counter is incremented. Otherwise, the content is stored in a new block with location l in the volume identified by $s \ mod \ n$;

5. Finally, a *contentkey* referencing the block is returned to the client.

The mod-based policy used to determine the volume where to store a content divides the load equally among the volumes. However, clients can define the volume where to store each content implementing alternative load-balancing policies. This way, in the presence of heterogeneous nodes, clients can impose higher workloads on those with higher throughput.

The elimination of duplicates at storage level is transparent to the clients because they just order the storage of a content and receive the correspondent contentkey, independently from if the content was identified as a duplicate.

A client *retrieves* a content by supplying the corresponding contentkey. The Content Manager decomposes the contentkey, identifies the signature of the content and the volume that hosts the correspondent block. The location of the block in the volume is obtained by applying sig2location to the signature. Finally, the content stored in the block is decompressed using the algorithm specified in the block's header, and the content is returned to the client.

The *delete* operation is also invoked with a contentkey as argument. The location of the block is executed by following the same process as for the retrieve operation. If the reference counter contained in the header of the block is set to one, the block is deleted. Otherwise, the reference counter is decremented. Since the location of the content is determined by the contentkey, the volume where the content is stored is directly accessed, both for the retrieve and delete operations. Therefore, the performance of these operations is independent from the number of volumes that compose an instance.

5.1.1.4 Fake duplicates

Theoretically, if two contents have the same signature they are duplicates. However, fingerprinting algorithms present a small probability of *collision* that causes the generation of the same signature for two different contents (Rabin, 1979). Relying exclusively on the comparison of signatures to detect duplicates within a large collection of contents, could cause some contents to be wrongly identified as duplicates and not stored. These situations are called *fake duplicates*.

The probability of occurrence of collisions depends on the fingerprinting algorithm, but it is extremely small in most cases (less than one in one million). On the other hand, a WWh holds millions of contents so a fake duplicate may occur. The remote possibility of loosing a content is acceptable for most people, but it could be disquieting for a librarian in charge of preserving an unique content of great historical importance. I believe that the probability of loosing a content due to a disk error or bug on the underlying software (e.g imported software libraries or hardware drivers) is higher than the probability of fingerprint collisions. Nevertheless, the Content Manager supports three modes for the store operation to fulfill the requirements of applications that may need absolute certainty that fake duplicates do not occur: *force-new*, *regular* and *compare*.

When using the force-new mode, the elimination of duplicates is switched off and a new block is created to store each content. This semantic is useful if one knows that the collection does not contain duplicates.

The *regular* mode (default) detects a collision if two contents have the same signature, but different sizes. In this case, an overflow block is created to keep the content. However, the success of this heuristic depends on the distribution of the content sizes and collisions will not be detected among contents with the same size. The distribution of sizes for a sample of 3.2 million Portuguese pages revealed that the probability of two random pages having the most frequent size is 4.15 x 10^{-6}. Assuming that the probability of two pages having the same size and the probability of fingerprint collision between them are independent events, the obtained results indicate that the comparison of sizes can substantially reduce the occurrence of fake duplicates without requiring longer fingerprint signatures or additional meta-data.

The *compare* mode relies on size and bytewise comparison of the contents to detect collisions. If two contents have the same signature but different sizes, or have the same size but are not byte equal, a collision is detected. Fake duplicates never occur when using this store mode.

5.1.1.5 Content Manager architecture

The Versus Content Manager presents a distributed architecture composed by a set of autonomous nodes that provide disk space with no central point of coordination, extending storage capacity by adding new nodes without imposing major changes in the system. Although network file systems also provide distributed access to data, they are executed at operating system kernel level and require administrative privileges to be installed and operated (Rodeh & Teperman, 2003). On its turn, the Content Manager is platform independent and runs at application level without imposing changes in the configuration of the underlying operating system. Peer-to-peer file systems, such as Oceanstore (Rhea *et al.*, 2003), are designed to manage a large and highly variable set of nodes with small storage capacity, distributed over wide-area networks (typically the Internet). This raises specific problems and imposes complex intra-node communication protocols that guarantee properties such as security, anonymity or fault tolerance that unnecessarily limit throughput on controlled networks. An experiment performed by the authors of Oceanstore showed that it is on average 8.3 times slower than NFS (Callaghan *et al.*, 1995) on a local-area network (LAN).

The Versus Content Manager has a different scope. It assumes that nodes have large storage capacity, are located within the same LAN, and changes on the configuration of the set of storage nodes are unfrequent. The collections managed by the Content Manager contain millions of contents with identical probabilities of being read; therefore, caching mechanisms are not effective. The design assumes that all the nodes have similar distance costs to the clients and there are no malicious clients. The system tolerates faults of storage nodes, but it does not provide automatic recovery and full availability of the contents kept in the faulty nodes. However, the Content Manager provides methods that enable the implementation of replication policies within an instance.

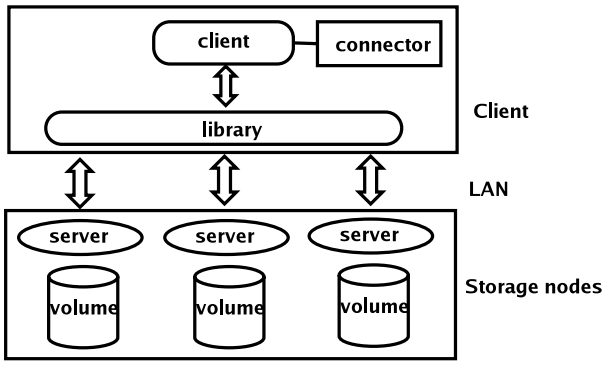


Figure 5.4: Architecture of the Versus Content Manager.

Figure 5.4 depicts the architecture of the Content Manager. An instance is composed by a thin middleware *library*, the *connector* object and the *volume servers*. Clients access an instance through a connector object that keeps references to the volumes that compose the instance. A change in the composition of the instance, such as the addition of a new volume, implies an update of the connector. Each volume server manages the requests and executes the correspondent low-level operations to access the contents. The contents are transmitted between the library and the servers in a compressed format to reduce network traffic and data processing on the server.

5.1.1.6 Implementation

The storage structure of a volume was implemented as a directory tree over the filesystem where the blocks are files residing at the leaf directories. The block header is written in ASCII format so that it can be easily interpreted, enabling access to the content kept in the block independently from the Webstore software. A 64-bit implementation of Rabin's fingerprinting algorithm was used to generate the content signatures (Rabin, 1979). The Content Manager supports Zlib as the built-in compression method, but other compression algorithms can be included. This way, contents can be compressed using adequate algorithms and accommodate new formats. The connector object was implemented as an XML file. The library and the volume servers were written in Java using JDK

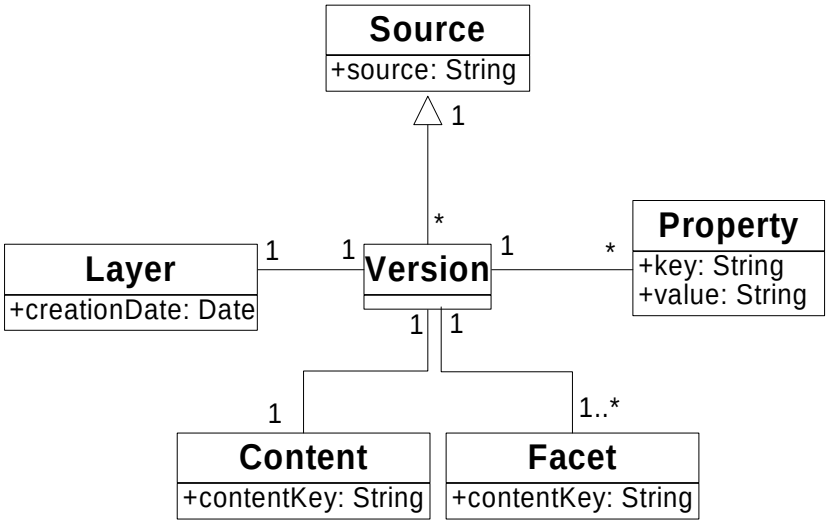


Figure 5.5: Versus Content Manager data model.

1.4.2 (6 132 lines of code). The communication between them is through Berkeley sockets. Currently, clients access volume servers in sequence, (a multicast protocol is not yet implemented). Volume servers are multi-threaded, launching a thread for handling each request. Volume servers guarantee that each block is accessed in exclusive mode through internal block access lists.

5.1.2 Catalog

This section describes the data model that supports the Catalog and the operational model that enables parallel loading and access to the data stored in the Versus repository.

5.1.2.1 Data model

Figure 5.5 presents the UML class model of the Catalog. This model is generic to enable its usage for long periods of time independently from the evolution of web formats. Plus, it also enables the usage of the repository in contexts different from Web Warehousing. For instance, it can be applied to manage meta-data on the research articles kept in a Digital Library. However, it is assumed that the contents are inserted into the repository in bulk loads.

The *Source* class identifies the origin of the content, for example an URL on the web. Each *Version* represents a snapshot of the information gathered from a Source. The Versions that were stored in the repository in the same bulk load are aggregated in *Layers*. A Layer represents the time interval from its creation until the creation of the next one. This way, time is represented in a discrete fashion within the repository, facilitating the identification of contents that need to be presented together, such as a page and the embedded images. The *Property* class holds property lists containing meta-data related to a Version. The use of property lists instead of a static meta-data model, enables the incremental annotation of contents with meta-data items when required in the future. The *Content* and *Facet* classes reference contents stored in the Content Manager. The former references the contents in their original format and the latter alternative representations. For instance, a Content is an HTML page that has a Facet that provides the text contained in it. Facets can also provide storage for current representations of contents retrieved earlier in obsolete formats. Versus supports merging the Content, Facets and meta-data of a Version into a single Facet in a semi-structured format (XML), so that each content stored in the Content Manager can be independently accessed from the Catalog. There are contents that are related. For instance, pages have links to other pages. The *Reference* class enables the storage of associations of Versions that are related to each other.

5.1.2.2 Operational model

A Versus client application is composed by a set of threads that process data in parallel. Each application thread does its specific data processing and Versus is responsible for managing and synchronizing them. The operational model of Versus was inspired on the model proposed by Campos (2003). It is composed by three workspaces with different features that keep the contents meta-data:

Archive (AW). Stores meta-data permanently. It keeps version history for the contents to enable the reconstruction of their earlier views. The AW is an append-only storage, the data stored cannot be updated or deleted;

Group (GW). Keeps a temporary view of the meta-data shared by all application threads. It enables the synchronization among the application threads and data cleaning before the archival of new data;

Private (PW). Provides local storage and fast access to data by application threads. Private workspaces are independent from each other and reside on the application threads addressing space.

An application thread can be classified in three categories according to the task it executes:

Loader. Generates or gathers data from an information source and loads it into Versus;

Processor. Accesses data stored in Versus, processes it and loads the resulting data;

Reader. Accesses data stored in Versus and does not change it, neither generates new information.

The data stored is partitioned to enable parallel processing. A Working Unit (WU) is a data container used to transfer partitions of meta-data across the workspaces. The Working Units are transferred from one workspace to another via *check-out* and *check-in* operations (Katz, 1990). When a thread checks-out a WU, the corresponding meta-data is locked in the source workspace and it is copied to the destination workspace. When the thread finishes the processing of the WU, it integrates the resulting data into the source workspace (check-in). The threads that compose an application share the same partitioning function. There are two classes of Working Units:

Strict. Contain exclusively the Versions that belong to the Working Unit. Strict Working Units should be used by applications that that do not need to create new Versions;

Extended. The Extended Working Units may also contain Versions that do not belong to the WU, named the *External Versions*.

The check-outs are executed in the same way for Strict and Extended Working Units. The partitioning function is applied to the source workspace to determine which Versions belong to the WU, then they are copied to the destiny workspace.

A conflict arises when a thread tries to check-in a Version that was previously inserted by another thread. Conflicts never occur during the check-in of Strict Working Units, because the thread acquired exclusive access to all the Versions belonging to the WU when it executed the check-out operation. Thus, the check-in can be performed very efficiently, transferring Versions in batch without any further testing. However, conflicts may occur during the check-in of the External Versions contained in the Extended Working Units. If an External Version conflicts an existing one, the conflict is automatically resolved under a First-come First-served policy: the meta-data and content correspondent to the External Version are discarded. The advantage of using Extended Working Units is that application threads can work in a completely independent fashion until the check-in, even if they have to create Versions belonging to other Working Units.

The partitioning function used to create Working Units is defined according to the characteristics of data to fulfil the requirements of the applications. For instance, if each thread of an application needs to process a page plus the pages it links to, the WU must contain these data. However it must be disjunctive: one Version belongs to one partition at most. On the other hand, the partitioning function must create Working Units with a size adequate to the resources available to support Versus. Ideally, the Working Units should contain a fixed amount of meta-data, but this may not be compatible with the requirements of the applications. Hence, the partitioning function must be carefully chosen to avoid the creation of partitions with very skewed sizes. Abnormally large Working Units can exhaust the resources available to support a thread's PW (e.g. memory) and originate very long check-in and check-out operations. On the other hand, extremely small Working Units can overload Versus with synchronization operations, compromising the performance benefits of using parallel processing, because synchronization tasks may take longer than the processing of the Working Units.

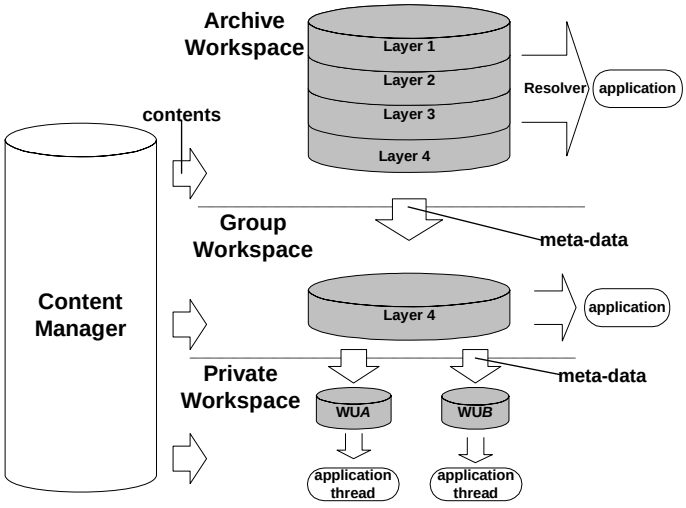


Figure 5.6: Accessing information stored in Versus.

Access

Figure 5.6 depicts the access to information stored on the Catalog. Each workspace provides different access levels to the meta-data. The Archive Workspace provides centralized access to all the archived Versions. The applications define the timespace of the Versions they want to access through a *Resolver* object. For instance, an application can use a Resolver that chooses the last Version archived from each Source. The Group Workspace also provides centralized access to the stored information but it holds at most one Version from each Source. It does not provide historical views on the data. The Private Workspaces are hosted on the application threads and keep one WU at a time enabling parallel processing. The Archive and Group Workspace should be hosted on powerful machines while the Private Workspaces can be hosted on commodity servers. The workflow of a parallel application that accesses data stored in Versus is the following:

1. The application checks-out the required meta-data from the AW to the GW;

2. Several application threads are launched in parallel. Each one of them starts its own PW and iteratively checks-out one WU at a time, processes it and

executes the check-in into the GW. The contents cannot be updated after they are set and any change on a content must be stored as new Facet;

3. When there are no unprocessed Working Units the new data kept in the GW is checked-in into the AW.

The contents are not transferred in the check-out operations, they are retrieved on-demand from the Content Manager. There are two assumptions behind this design decision. The first is that the contents belonging to a WU represent an amount of data much larger than the corresponding meta-data. To enable quick transfers of contents during the check-outs, it would be necessary to use faster network links between the workspaces. Plus, the application threads would require more storage space to keep the contents locally.

The second assumption is that application threads do not need to process all the contents belonging to a WU or they may be exclusively interested on analyzing the meta-data. Thus, checking-out the contents in advance would be a waste of resources.

Load

Figure 5.7 depicts the loading of data into Versus. The meta-data is loaded by the application threads into the Private Workspaces, while the contents are stored in the Content Manager that eliminates duplicates at storage level. However, if a content is identified as a duplicate, the corresponding meta-data, such as URL, is still stored in the Catalog PW. This way, the Webhouse clients can later access the warehoused collection independently from the elimination of duplicates mechanism. The work flow of a parallel application that loads information into Versus is the following:

1. The application creates a new empty layer on the GW;

2. Several parallel threads are launched. Each one of them starts its own PW and iteratively checks-out one empty WU, loads it with meta-data extracted from the Sources and executes the check-in into the GW;

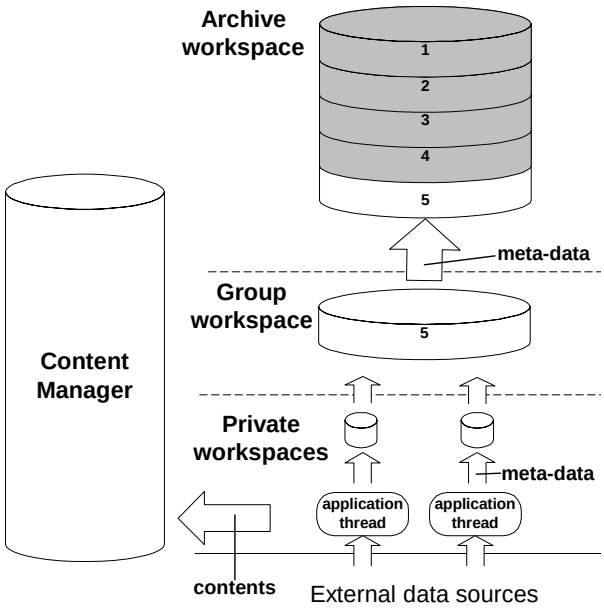


Figure 5.7: Loading data into Versus.

3. When there are no unprocessed Working Units the new data kept in the GW is checked-in into the AW.

The references to the new contents loaded into the Content Manager are kept as meta-data in the PW. If an application thread fails before the check-in, for instance due to a power failure, the references to the contents would be lost, originating orphan contents that could not be later accessed. Versus provides recovery methods to restart the processing of a Working Unit and remove the orphan contents if an application thread fails. Versus also supports direct loads to the Group or Archive Workspaces but they should be used for small amounts of data because parallel loading is not supported.

5.1.2.3 Implementation

The Catalog was mainly implemented using the Java environment and relational database management systems (DBMS). The AW and GW were implemented using Oracle 9i DBMS (Oracle Corporation, 2004). The advanced administration features of this DBMS, such as partitioning or query optimizations, enable

the configuration of the system to be used in the context of Web Warehousing, addressing efficiently the processing of large amounts of data. The use of the SQL language for data manipulation enabled the reuse of the code in the three kinds of workspaces, although each one also had particular data structures and optimization profiles.

The PW used HyperSonicSQL DBMS (HyperSonicSQL). It is written in Java and can be configured to run in three modes:

Memory. The DBMS runs inside the client application. The data is kept exclusively in-memory. If there is failure of the client application the data is lost;

File. The DBMS runs inside the client application but the data is stored in files;

Client/server. The DBMS and client applications run independently and communicate through a network connection using JDBC. The data can be kept in memory or in files.

The PW was implemented using the *memory-mode* because it provide faster responses and it was assumed that failures of the clients were not frequent. The choice of HyperSonicSQL for implementing the PW imposed that each application thread could not host more than one PW at a time because HyperSonicSQL only supports one database instance within each Java Virtual Machine. The Catalog clients access the information in the workspaces using a application programming interface (API) written in Java using JDK 1.4.2 (4 526 lines of code). This API accesses the databases using the Java Database Connectivity interface (JDBC) and hides the underlying technologies from the client applications. The Catalog also enables the submission of direct commands to the database management systems to take advantage from their advanced features.

5.2 The VN crawler

A WWh crawls data from the web to extract new information. The permanent evolution of the web and the upcoming of new usage contexts demands continuous

Partitioning strategy	DNS caching	Use keep-alive connections	Avoid server overloading	Reuse site meta-data	Independency
IP	++	++	++	+	–
Site	+	+	+	++	++
Page	–	–	–	–	++

Table 5.1: Comparison of the partitioning strategies.

research in crawling systems. Kahle (2002), the founder of the Internet Archive, revealed that their commercial crawler is rewritten every 12–18 months to reflect changes in the structure of the web. Although a crawler is conceptually simple, its development is expensive and time consuming, because most problems arise when the crawler leaves the experimental environment and begins harvesting the web.

This section discusses crawling. First, it presents the requirements for a crawler and discusses how the web characteristics influence the choice of a suitable partitioning function for the URL space. Then, it presents a flexible and robust crawler architecture that copes with distinct usage contexts. Finally, it provides a detailed description of hazardous situations to crawling and discusses solutions to mitigate their effects.

5.2.1 Partitioning strategies

A suitable partitioning function that divides the URL space across the set of Crawling Processes that compose a distributed crawler must be chosen according to the characteristics of the portion of the web being harvested. Otherwise, the requirements for a crawler may not be fulfilled. Three partitioning strategies were analyzed: IP, site and page partitioning (see Chapter 2). The number of URLs contained in a partition should be ideally constant to facilitate load balancing. The page partitioning is the most adequate according to this criterion. The IP partitioning tends to create some extremely large partitions due to servers that host thousands of sites, such as Geocities (`www.geocities.com`) or Blogger (`www.blogger.com`). The site partitioning is more likely to create partitions containing a single URL, due to sites under construction or presenting an error message. Thus, the efficiency of the IP and site partitioning is more dependent on

the characteristics of the portion than page partitioning. Table 5.1 summarizes the relative merits of each strategy, which are characterized by the following determinants:

DNS caching. A CP executes a DNS lookup to map the site name contained in an URL into an IP address, establishes a TCP connection to the correspondent web server and then downloads the content. The DNS lookups are responsible for 33% of the time spent to download a content (Habib & Abrams, 2000). Hence, caching a DNS response and using it to download several contents from the same site optimizes crawling. A CP does not execute any DNS lookup during the crawl when harvesting an IP partition, because all the URLs are hosted on the IP address that identifies the partition. A site partition requires one DNS lookup to be harvested because all its URLs have the same site name. A page partition contains URLs from several different sites, so a CP would not benefit from caching DNS responses;

Use of keep-alive connections. Establishing a TCP connection to a web server takes on average 23% of the time spent to download a content (Habib & Abrams, 2000). However, HTTP keep-alive connections enable the download of several contents reusing the same TCP connection to a server (Fielding *et al.*, 1999). A page partition contains URLs hosted on different servers, so a CP does not benefit from using keep-alive connections. On the other hand, with IP partitioning an entire server can be crawled through one single keep-alive connection. When a crawler uses site partitioning, a single keep-alive connection can be used to crawl a site. However, the same web server may be configured to host several virtual hosts. Then, each site will be crawled through a new connection;

Server overloading. In general, a crawler should respect a minimum interval of time between consecutive requests to the same web server to avoid overloading it. This is called the courtesy pause. Page partitioning is not suitable to guarantee courtesy pauses, because the URLs of a server are spread among

several partitions. Thus, if no further synchronization mechanism is available, the Crawling Processes may crawl the same server simultaneously, disrespecting the courtesy pause. The page partitioning requires that each CP keeps track of the requests executed by the other ones to respect the courtesy pause. With IP partitioning, it is easier to respect the courtesy pause because each CP harvests exclusively the URLs of a web server and simply needs to register the time of the last executed request to respect it. A crawler using site partitioning respects at first sight a minimum interval of time between requests to the same site but, a server containing virtual hosts may be overloaded with requests from Crawling Processes that harvest its sites in parallel. On the other hand, a web server containing virtual hosts should be designed to support parallel visits to its sites performed by human users. Hence, it should not become overloaded with the parallel visits executed by the Crawling Processes;

Reuse of site meta-data. Sites contain meta-data, such as the Robots Exclusion file, that influences crawling. The page partitioning strategy is not suitable to reuse the site's meta-data because the URLs of a site are spread across several partitions. With the IP partitioning, the site's meta-data can be reused, but it requires additional data structures to keep the correspondence between the sites and the meta-data. Notice however that this data structure can grow considerably, because there are IP partitions that contain thousands of different sites generated through virtual hosting. On its turn, when a crawler is harvesting a site partition, the site's meta-data is reused and the crawler just needs to manage the meta-data of a single site;

Independency. The site and page partitioning enable the assignment of an URL to a partition independently from external resources. The IP partitioning depends on the DNS servers to retrieve the IP address of an URL and it cannot be applied if the DNS server becomes unavailable. If the site of an URL is relocated to a different IP address during a crawl, two invocations of the function for the same URL would return different partitions. In this

case, the URLs hosted on the same server would be harvested by different Crawling Processes.

5.2.2 Crawler architecture

This section details the design of the VN crawler. It was designed as a Versus client application to take advantage of the distribution features provided by the Versus repository. VN has a hybrid Frontier, uses site partitioning and dynamic-pull assignment:

Hybrid frontier. Each CP has an associated Local Frontier where it stores the meta-data generated during the crawl of a partition. The meta-data on the seeds and crawled URLs is stored on the Global Frontier. A CP begins the crawl of a new site partition by transferring a seed from the Global to its Local Frontier (*check-out*). Then, the URLs that match the site are harvested by the CP. When the crawl of the partition is finished, the correspondent meta-data is transferred to the Global Frontier (*check-in*). A CP successively checks-out a partition containing a seed, harvests the corresponding information from the web and checks-in the resultant meta-data, until there are no unvisited seeds in the Global Frontier;

Site partitioning. Besides the advantages discussed in the previous Section, three additional reasons lead to the adoption of the site partitioning strategy. First, a CP frequently accesses the Local Frontier to execute the URL-seen test. As Portuguese sites are typically small and links are mostly internal to the sites (see Chapter 3), the Local Frontier can be maintained in memory during the crawl of the site to optimize the execution of the URL-seen test. Second, web servers are designed to support access patterns typical of human browsing. The crawling of one site at a time enables the reproduction of the behavior of browsers, so that the the actions of the crawler do not disturb the normal operation of web servers. Third, site partitioning facilitates the implementation of robust measures against spider traps;

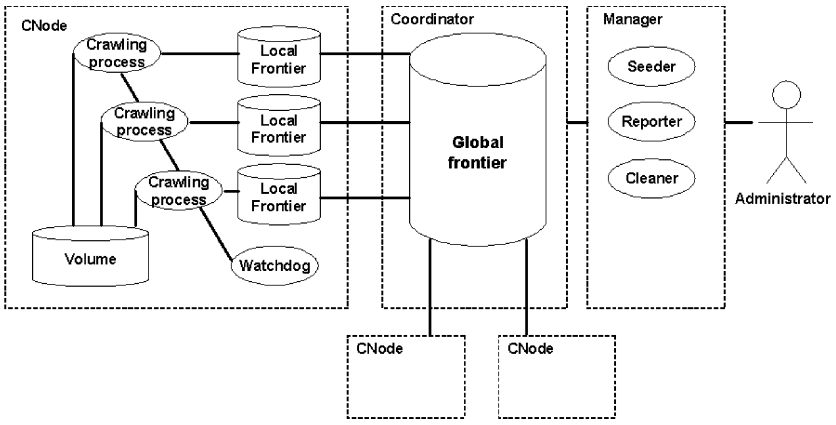


Figure 5.8: VN architecture.

Dynamic-pull assignment. The Global Frontier assigns partitions to Crawling Processes as they pull them. The Global Frontier guarantees that a site is never harvested simultaneously by two Crawling Processes. The Global Frontier identifies each partition with the site's hash and manages three lists: i) sites to crawl; ii) sites being crawled and; iii) sites crawled. When a CP checks-out a partition, it is moved from the first to the second list. The checks-in moves the partition from the second to the third list.

Figure 5.8 describes VN's architecture. It is composed by a Global Frontier, a *Manager* that provides tools to execute administrative tasks and several *Crawling Nodes* (CNodes). The Manager is composed by:

- The *Seeder* that generates seeds to a new crawl from user submissions, DNS listings and home pages of previously crawled sites and inserts them in the Global Frontier;
- The *Reporter* that gets statistics on the state of the system and emails them to a human *Administrator*;
- The *Cleaner* allows to release resources acquired by faulty Crawling Processes.

Each CNode hosts:

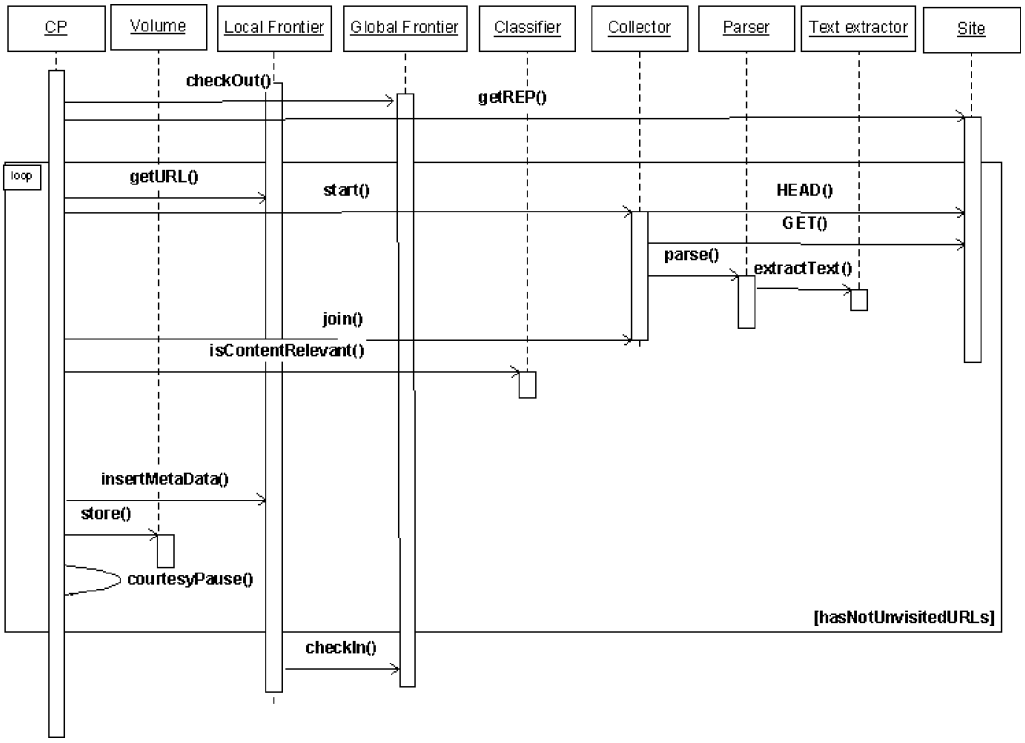


Figure 5.9: Sequence diagram: crawling a site.

- The *Crawling Processes* and the corresponding *Local Frontiers*;
- One Volume that stores the harvested contents;
- One *Watchdog* that restarts the Crawling Processes if they are considered dead (inactive for a given period of time).

The scheduling of the execution of the Crawling Processes within a CNode is delegated to the operating system. It is assumed that when a CP is blocked, for instance while executing IO operations, another CP is executed.

5.2.2.1 Crawling algorithm

Crawlers get a seed to a site and follow the links within it to harvest its contents. They usually impose a depth limit to avoid the harvesting of infinite sites (Baeza-Yates & Castillo, 2004). There are three policies to traverse links within a site (Cothey, 2004):

Best-first. The crawler chooses the most relevant URLs to be crawled first according to a given criteria as, for instance, their PageRank value (Brin & Page, 1998);

Breadth-first. The crawler iteratively harvests all the URLs available at each level of depth within a site;

Depth-first. The crawler iteratively follows all the links from the seed until the maximum depth is achieved.

The best-first policy is aimed to select relatively small sets of high-quality pages to save system resources, such as bandwidth. These pages are usually spread across different sites which could jeopardize the advantages of using site partitioning. However, Najork & Wiener (2001) showed that in practice breadth-first and best-first policies yield high-quality pages. The breadth-first policy presents the advantage of being simpler to implement than best-first and avoids server overloading (Chakrabarti *et al.*, 2002). Crawlers impose a limit on the number of URLs visited within each site as a robustness measure against infinite sites. Assuming that the most relevant pages are located at shallower levels of depth, when using the depth-first approach the maximum number of URLs for a site may be achieved before a significant part of important pages were harvested. Thus, the VN Crawling Processes harvest information from the web visiting one site at a time in a breadth-first mode. Figure 5.9 details the stages of this process. The crawl of a new site begins when the CP *checks-out* a seed. The CP downloads the "robots.txt" file (Robots Exclusion Protocol) and then, iteratively harvests one URL at a time from the site until there are no URLs to visit (*loop*). The CP launches a *Collector* thread that downloads and processes information referenced by an URL. The Collector requests the HTTP headers of the URL to check if the content should be downloaded. For instance, if the content is an MP3 file and the selection criteria defines that only HTML pages should be harvested, the content is not downloaded. Then, the content is parsed to extract various meta-data, such as links to other pages. The extraction and storage of meta-data from the contents during the crawl while they are stored on memory, avoids redundant processing by the Versus client applications that will latter process the web data.

Finally, the Collector returns the content and extracted meta-data to the CP. This information is analyzed by a *Classifier* that checks if the content matches the selection criteria. If the content is considered relevant, it is stored in a *Volume* and the meta-data is inserted in the Local Frontier, otherwise it is discarded. The CP sleeps after crawling each URL to execute a courtesy pause. When the CP finishes visiting the site, it checks-in the partition.

5.2.2.2 Fault management

To face hazardous situations while crawling the web and possible hardware problems on the underlying cluster of machines, VN was designed to tolerate faults at different levels in its components.

A CP launches an independent thread (Collector) to execute sensitive tasks, such as the download, meta-data extraction and parsing of a content. The CP terminates the execution of the Collector after a limited time. Hence, a fault caused by the harvest and processing of a single content does not compromise the crawl of the remaining contents at the site. However, this imposes the overhead of launching a new thread to crawl each URL.

The Crawling Processes are independent from each other and the failure of one of them does not influence the execution of the remaining. However, they depend on the Global Frontier for synchronization. A CP operates independently from the Global Frontier between the check-out and the check-in events. A fault of the Global Frontier causes a gradual degradation of the system because the Crawling Processes will continue harvesting until the check-in is tempted. As a result, there is an interval of time when it is possible to recover the Global Frontier without stopping the crawl. For instance, if a network cable is disconnected from the machine hosting the Global Frontier, the incident can be solved without influencing the progress of the crawl.

The contents can be stored locally on the same machine that hosts the CP or remotely on any other machine that hosts a Volume. Therefore, if the Volume used by a CP fails, for instance because it exhausted its storage capacity, the CP can be quickly set to store contents on remote Volumes, without requiring any data migration.

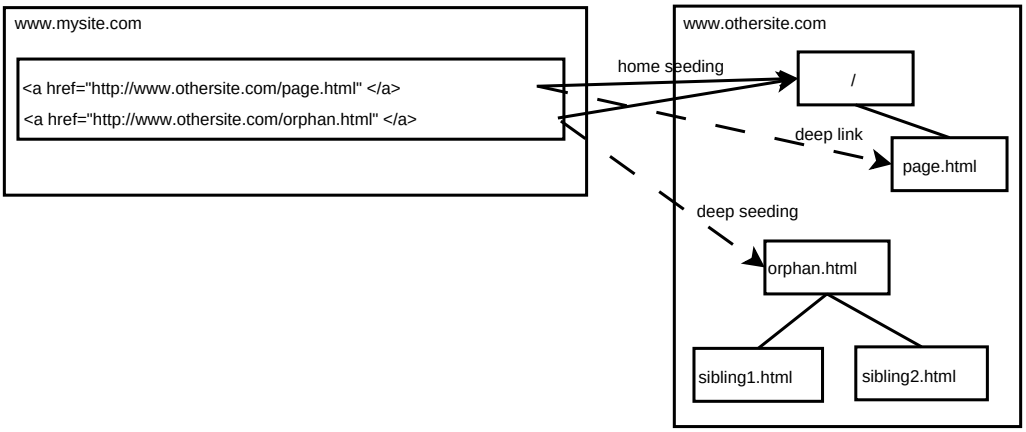


Figure 5.10: Deep vs. home page seeding policies.

5.2.2.3 URL-seen test

The URL-seen test is executed in two steps: first, when the URLs are inserted in the Local Frontier and upon the check-in to the Global Frontier. 81.5% of the links embedded in pages reference URLs internal to its site (Broder *et al.*, 2003). The URL-seen test for internal URLs is done locally because all the seen URLs belonging to the site are covered by the Local Frontier. So, when the CP finishes harvesting the site, it can check-in the internal URLs to the Global Frontier without further testing. However, the URL-seen test for the external URLs must be executed against all the URLs in the Global Frontier during check-in, because the URLs may have been inserted there meanwhile by another CP. Thus, the URL-seen test for external URLs is an expensive operation and the number of external URLs to check-in should be minimized. Nonetheless, the external URLs are important because they are potential seeds to newly found sites. There are three policies for the insertion of external URLs in the Frontier:

Home page. The home page policy assumes that all the contents within a site are accessible through a link path from its home page. Hence, a CP replaces every external URL by its site home page before inserting it in the Local Frontier (see Figure 5.10). The home page policy reduces the number of external URLs to check-in. However, if a CP cannot follow links from the home page, the remaining pages of the site will not be harvested;

Deep link. A deep link references an external URL different than the home page. The deep link policy assumes that there are pages not accessible through a link path from the home page of the site. The CP inserts the external URLs without any change in the Local Frontier to maximize the coverage of the crawl. For instance, in Figure 5.10 the URL `www.othersite.com/orphan.html` is not accessible from the home page of the site but it is linked from the site `www.mysite.com`. However, if the external URL references a content without links, such as a postscript document, the crawl of the site would be limited to this content. Some authors believe they make pages unavailable by removing the internal links to them, forgetting that external pages may maintain links to these pages. The deep link policy enables the crawling of these supposedly unavailable pages and may expose them in search engine results;

Combined. Follows deep links but always visits the home page of the sites. This policy is intended to maximize coverage. Even if a deep link references a content without links, the remaining site accessible through a link path from the home page will be harvested.

VN supports the home page and combined policies. As an optimization, when VN is configured to follow the home page policy, it discards the external URLs hosted on the sites contained in the seeds of the crawl, because they were already inserted in the Global Frontier by the Seeder. Discarding external URLs contained in the initial seed list breaks the deep link and combined policies, because a link may reference a page from a site contained in the initial seed list that is not accessible from the home page.

I believe that in general, the adoption of the combined policy provides a marginal gain of coverage against the home page policy because pages are usually accessible through a link path from their home pages. Hence, the home page policy is suitable for most crawling contexts, while the combined policy should be used when coverage needs to be maximized, such as to execute exhaustive crawls of corporate intranets.

5.2.2.4 Optimizing bandwidth usage

Invalid URLs reference contents that cannot be downloaded. Many URLs extracted from pages are invalid and they degrade the performance of a crawler because it will waste resources trying to harvest them. Thus, visits to invalid URLs should be minimized. They can be pruned using the following strategies:

Discarding malformed URLs. A malformed URL is syntactically incorrect (Berners-Lee *et al.*, 2005). Malformed URLs are most likely caused by a typing errors. For instance, an URL containing white spaces is syntactically incorrect. However, there are web servers that enable the usage of malformed URLs;

Discarding URLs that reference unregistered sites. The site name of an URL must be registered in the DNS. Otherwise, the crawler would not be able to map the domain name into an IP address to establish a connection to the server and download the content. Thus, an URL referencing an unregistered site name is invalid. However, testing if the site names of the URLs are registered before inserting them into the Frontier imposes an additional overhead on the DNS servers.

Duplicates occur when two or more different URLs reference the same content. A crawler should avoid harvesting duplicates to save on processing, bandwidth and storage space. The crawling of duplicates can be avoided through the normalization of URLs:

1. *Case normalization*: the hexadecimal digits within a percent-encoding triplet (e.g., "%3a" versus "%3A") are case-insensitive and therefore should be normalized to use uppercase letters for the digits A-F;

2. *Percent-Encoding Normalization*: decode any percent-encoded octet that corresponds to an unreserved character;

3. *Convert site name to lower case*: the domain names are case insensitive thus, the URLs `www.site.com/` and `WWW.SITE.COM/` reference the same content;

4. *Convert relative to absolute file paths*: For instance, www.site.com/dir/../index.html to www.site.com/index.html;

5. *Remove identification of the HTTP default port 80*: For instance, change www.site.com:80/index.html to www.site.com/index.html;

6. *Add trailing '/' when the path is empty*: The HTTP specification states that if the path name is not present in the URL, it must be given as '/' when used as a request for a resource (Fielding *et al.*, 1999). Hence, the transformation must be done by the client before sending a request. This rule of normalization prevents that URLs, such as www.site.com and www.site.com/, originate duplicates;

7. *Remove trailing anchors*: anchors are used to reference a part of a page (e.g www.site.com/file#anchor). However, the crawling of URLs that differ only on the anchors would result in repeated downloads of the same page;

8. *Add prefix "www." to site names that are second-level domains*: the following section will show that most of the sites named with a second-level domain are also available under the site name with the prefix "www.";

9. *Remove well-known trailing file names*: two URLs that are equal except for a well known trailing file name such as "index.html", "index.htm", "index.shtml", "default.html" or "default.htm", usually reference the same content. The results obtained in experiments crawling the Portuguese Web showed that removing these trailing file names reduced the number of duplicates by 36%. It is technically possible that the URLs with and without the trailing file reference different contents. However, situations of this kind were not found in the experiments. The conclusion is that this heuristic does not reduce the coverage of a crawler noticeably.

A request to an URL may result in a redirect response, (3** HTTP response code), to a different one named the *target URL*. For instance, the requests to URLs like www.somesite.com/dir, where dir is a directory, commonly result in a redirect response (301 Moved Permanently) to the URL www.somesite.com/dir/. Browsers follow the redirects automatically, so they are not detected

by the users. A crawler can also automatically follow a redirect response to download the content referenced by the target URL. However, both the redirect and the correspondent target URLs reference the same content. If they are linked from pages, the same content will be downloaded twice. Automatically following redirects during a crawl increased the number of duplicates by 26%. On the other hand, when a crawler does not follow redirects, it considers that a redirect is equivalent to a link to an URL. The crawler inserts both the redirect and target URLs in the Frontier: the former is marked as a redirect and the target URL is visited to download the content. The number of URLs inserted in the Frontier increases, approximately by 5% (Castillo, 2004; Gomes & Silva, 2005; Heydon & Najork, 1999), but duplicates are avoided.

5.2.2.5 Implementation

The VN web crawler integrates components developed within the XLDB Group and external software. It was mainly written in Java using jdk1.4.2 (3 516 lines of code), but it also includes software components implemented in native code. The Crawling Processes use hash tables to keep the list of duphosts and the DNS cache. The Parser was based on WebCAT, a Java package for extracting and mining meta-data from web documents (Martins & Silva, 2005b). The Classifier used to harvest the Portuguese Web includes a language identifier (Martins & Silva, 2005a). The Robots Exclusion file interpreter was generated using Jlex (Berk & Ananian, 2005). The Seeder and the Cleaner are Java applications. The Reporter and Watchdog were implemented using shell scripts that invoke operating system commands, such as `ps` or `iostat`.

VN was designed as a Versus client application to take advantage of the distribution features provided by the repository. The Local Frontiers of the Crawling Processes were implemented using the Versus Private Workspaces. The Global Frontier was implemented using the Versus Group Workspace. The Content Manager supports the Volumes used to store the harvested contents.

5.3 Coping with hazardous situations

The web is very heterogeneous and there are hazardous situations to crawling that disturb the extraction of web data to be integrated in a WWh. Some of them are malicious, while others are caused by mal-functioning web servers or authors that publish information on the web without realizing that it will be automatically processed. Crawler developers must be aware of these situations to design robust crawlers capable of coping with them.

The description of hazardous situations to crawling is scarce in the scientific literature, because most experiments are based on simulations or short-term crawls that do not enable their identification. Hazardous situations are commonly ignored in academic studies because the scientific hypotheses being tested assume a much simpler model of the web than observed in reality. Hence, the detection of hazardous situations on the web is a recurrent problem that must be addressed by every new system developed to process web data. Moreover, new hazardous situations arise as the web evolves, so their monitoring and identification requires a continuous effort.

This section describes hazardous situations found on the Portuguese Web and discusses solutions to mitigate their effects. First, it presents examples of situations that cause unnecessary downloads and degrade the performance of a crawler. Then, it describes contents that are hard to be automatically processed and frequently prevent crawlers from following their embedded links to other contents. Finally, it discusses heuristics to detect sites with different names that provide the same contents (duphosts), causing the crawl of a large number of duplicates.

5.3.1 Spider traps

Heydon & Najork (1999) defined a spider trap as *an URL or set of URLs that cause a crawler to crawl indefinitely.* In this thesis the definition was relaxed and situations that significantly degrade the performance of a crawler were also considered as spider traps, although they may not originate infinite crawls. Initially, the pages dynamically generated when a server received a request were pointed as

the general cause of spider traps and they were excluded from crawls as a preventive measure (Cho *et al.*, 1998). However, dynamic pages became very popular because they enable the management of information in databases independently from the format used for publication. It was estimated that there are 100 times more dynamic pages than static ones (Handschuh *et al.*, 2003). Thus, preventing a crawler from visiting dynamic pages to avoid spider traps would exclude a large parcel of the web.

Some webmasters create traps to boost the placement of their sites in search engine results (Heydon & Najork, 1999), while others use traps to repel crawlers because they spend the resources of the web servers. Spider traps bring disadvantages to their creators. A trap compromises the navigability within the site and human users get frustrated if they try to browse a spider trap. Plus, search engines have a key role in the promotion of web sites, and they ban sites containing traps from their indexes (Cho & Roy, 2004; Olsen, 2002). There are several examples of spider traps and possible solutions to mitigate their effects:

DNS wildcards. A zone administrator can use a DNS wildcard to synthesize resource records in response to queries that otherwise do not match an existing domain (Mockapetris, 1987). In practice, any site under a domain using a wildcard will have an associated IP address, even if nobody registered it. DNS wildcards are used to make sites more accepting of typographical errors in URLs because they redirect any request to a site under a given domain to a default doorway page (ICANN, 2004). The usage of DNS wildcards is hazardous to crawlers because they enable the generation of an infinite number of site names to crawl under one single domain. Moreover, it is not possible to query a DNS server to detect if a given domain is using wildcards. However, a crawler should be able to know that a given site is reached through DNS wildcarding before harvesting it. To achieve this, one could execute DNS lookups for a set of absurd sites names under a domain and check if they are mapped to the same IP address. If they are, the domain is most likely using a DNS wildcard. This way, a black list of domain names that use DNS wildcards could be compiled and used to prevent crawlers from harvesting them. However, many domains that

use DNS wildcarding also provide valuable sites. For instance, the domain blogspot.com uses DNS wildcarding but also hosts thousands of valuable sites;

Malfunctions and infinite size contents. Malfunctioning sites are the cause of many spider traps. These traps usually generate a large number of URLs that reference a small set of pages containing default error messages. Thus, they are detectable by the abnormally large number of duplicates within the site. For instance, sites that present highly volatile information, such as online stores, generate their pages from information kept in a database. If the database connection breaks, these pages are replaced by default error messages informing that the database is not available. A crawler can mitigate the effects of this kind of traps by not following links within a site when it tops a number of duplicates. A malfunctioning site may start serving contents with a higher latency. This situation would cause that a crawler would take too much time to harvest its contents, delaying the overall progress of the crawl. To prevent this, a crawler should impose a limit on the time to harvest each URL.

There are also infinite size contents, such as online radio transmissions, that cause traps if a crawler tries to download them. A crawler may truncate the content if it exceeds a maximum limit size. Notice that contents in HTML format can be partially accessed if they are truncated, but executable files become unusable;

Session identifiers and cookies. HTTP is a stateless protocol that does not allow tracking of user reading patterns by itself. However, this is often required by site developers, for instance, to build profiles of typical users. A session identifier embedded in the URLs linked from pages allows maintaining state about a sequence of requests from the same user. According to web rules, a session identifier should follow a specific syntax beginning with the string "SID:" (Hallam-Baker & Connolly, 2005). But in practice, the session identifiers are embedded by developers in URLs as any other parameters. Session identifiers have lifetimes to prevent that different users

are identified as the same one. A session identifier replacement causes that the URLs linked from the pages are changed to include the new identifier. If the crawl of a site lasts longer than the lifetime of a session identifier, the crawler could get trapped harvesting the new URLs generated periodically to include the new session identifiers. The replacement of session identifiers also originates duplicates, because the new generated URLs, reference the same contents as previously crawled (Douglis *et al.*, 1997). A crawler may avoid getting trapped by stop following links within the site when a limit number of duplicates is achieved. This heuristic fails if the pages of the site are permanently changing and the new URLs reference distinct contents. In this case, the insertion of new links within the site should be stopped when a limit number of URLs crawled from the site is achieved.

Cookies have been replacing session identifiers embedded in URLs (Bent *et al.*, 2004). A cookie is a piece of data sent by the site that is stored in the client and enables tracking user sessions without URL changes. A crawler able to process cookies is less prone to fall in traps caused by session identifiers embedded in URLs. It was observed that 27% of the Portuguese Web URLs contained well-known session identifier parameter names (phpsessid, sid, sessionid). Interestingly, 95% of them were hosted in sites developed with PHP engines. A visit to some of these sites revealed that the links of the pages were changed by the web server to contain session identifiers when the HTTP client did not accept cookies. VN was enhanced to accept cookies and the percentage of URLs containing session identifiers dropped to 3.5%. This also had the noteworthy effect of reducing the average URL length from 74 to 62 characters, which saved space on the data structures in the Frontiers;

Directory list reordering. Apache web servers generate pages to present lists of files contained in a directory. This feature is used to easily publish files on the web. Figure 5.11 presents a directory list and its embedded links. The directory list contains 4 links to pages that present it reordered by *Name*, *Last-Modified date*, *Size* and *Description*, in ascendent or descendent order. Thus, if a crawler follows all the links embedded in a directory

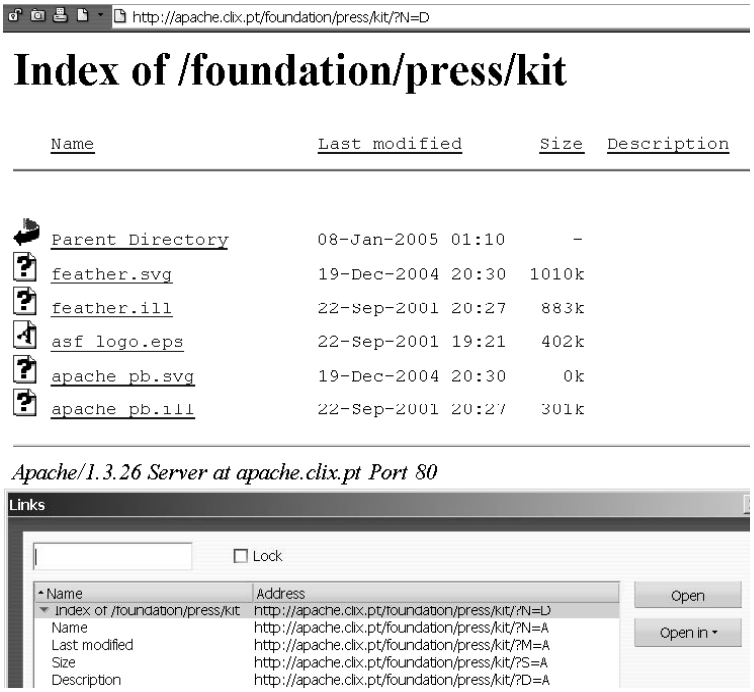


Figure 5.11: Apache directory list page and the linked URLs.

list page, it will harvest the same information referenced by 8 different URLs. Moreover, a directory list enables browsing a file system and may accidentally expose contents that were not meant to be published on the web. A crawler could get trapped if harvesting, for instance, temporary files periodically generated in a directory. A crawler could exclude URLs that reference directory listings to avoid traps but they are frequently used to publish valuable contents, such as open-source software and documentation;

Growing URLs. A spider trap can be set with a symbolic link from a directory `/spider` to the directory `/` and a page `/index.html` that contains a link to the `/spider` directory. Following the links will create an infinite number of URLs (`www.site.com/spider/spider/...`) (Jahn, 2004). Although, this example may seem rather academic, these traps exist on the web. There are also advertisement sites that embedded the history of the URLs followed by an user on the links of their pages. The idea is that when users reach a given page they stop browsing and the URL that referenced the page

contains the history of the URLs previously visited. This information is useful for marketing analysis. The problem is that a crawler never stops "browsing" and it gets trapped following the generated links. Hence, a crawler should impose a limit on the length of the URLs harvested.

5.3.2 Hard to interpret contents

Crawlers interpret the harvested contents to extract valuable data such as links or texts. If a crawler cannot extract the linked URLs from a page, it will not be able to iteratively harvest the web. The page text is important for focused crawlers that use classification algorithms to determine the relevance of the contents, for instance, a focused crawler could be interested in harvesting contents containing a set of words. However, the extraction of data from contents is not straightforward because there are situations on the web that make contents hard to interpret:

Wrong identification of media type. The media type of a content is identified through the HTTP header field Content-Type. HTTP clients choose the adequate software to interpret the content according to its media type. For instance, a content with the Content-Type application/pdf is commonly interpreted by the Adobe Acrobat software. However, sometimes the Content-Type values do not correspond to the real media type of the content (Gomes *et al.*, 2006a) and a HTTP client may not be able to interpret it correctly. An erroneous Content-Type response can be detected through the analysis of the extracted data. A page that does not contain any links raises the suspicion that something went wrong. If the text extracted from a content does not contain words from a dictionary or does not contain white spaces between sequences of characters, the content may have been incorrectly interpreted. If a crawler identifies an erroneous Content-Type response it may try to identify the correct type to enable the correct interpretation of the content. The format of a content is commonly related to the file name extension of the URL that references it. This information can be used to automatically identify the real media type of the content. However, the usage of file name extensions is not mandatory within URLs and the same file name extension may be used to identify more than one format. For example

the extension .rtf identifies contents in the application/rtf and text/richtext media types. The media type could also be guessed through the analysis of the content. For instance, if the content begins with the string <html> and ends with the string </html> it is most likely an HTML page (text/html media type). However, this approach requires specific heuristics to each of the many media types available on the web and identifying the media type of binary files is a complex task;

Malformed pages. A malformed content does not comply with its media type format specification, which may prevent its correct interpretation. Malformed HTML contents are prevalent on the web. One reason for this is that authors commonly validate their pages through visualization on their browsers, which tolerate format errors to enable the presentation of pages to humans without visible errors. As a result, the HTML interpreter used by a crawler should also be tolerant to common syntax errors, such as unmatched tags (Martins & Silva, 2005b; Woodruff *et al.*, 1996);

Cloaking. A cloaking web server provides different contents to crawlers than to other clients. This may be advantageous if the content served is a more crawler-friendly representation of the original. For instance, a web server can serve a Macromedia Shockwave Flash Movie to a browser and an alternative XML representation of the content to a crawler. However, spammers use cloaking to deceive search engines without inconveniencing human visitors.

Cloaking may be also unintentional. There are web servers that, when in the presence of an unrecognized client, return a page informing that the client's browser does not support the technology used in the site and suggest the usage of an alternative browser. A crawler may identify itself as a popular browser to avoid suffering from this cloaking situation. However, this solution violates the principles of politeness and webmasters could confuse the consecutive visits of a crawler with an attack to their sites;

JavaScript-intensive pages. JavaScript is a programming language created to write functions, embedded in HTML pages that enable the generation of

presentations that were not possible using HTML alone. The AJAX (Asynchronous JavaScript And XML) libraries contributed to the widespread usage of this language in pages (Paulson, 2005). It is becoming common to find pages where normal links are JavaScript programs activated through clicking on pictures or selecting options from a drop-down list (Thelwall, 2002).

A JavaScript program may build a link or a text to be accessed through a series of computational steps. However, writing an interpreter to understand what a JavaScript program is doing is extremely complex and computationally heavy. As consequence, the extraction of data from pages written using JavaScript is hard and crawlers usually identify the embedded URLs using pattern matching. For instance, they identify an URL embedded in a JavaScript program if it begins with the string "http://".

5.3.3 Duplicate hosts

Duplicate hosts (duphosts) are sites with different names that simultaneously serve the same content. Technically, duphosts can be created through the replication of contents among several machines, the usage of virtual hosts or the creation of DNS wildcards. There are several situations that originate duphosts:

Mirroring. The same contents are published on several sites to backup data, reduce the load on the original site or to be quickly accessible to some users;

Domain squatting. Domain squatters buy domain names desirable to specific businesses, to make profit on their resale. The requests to these domains are redirected to a site that presents a sale proposal. To protect against squatters, companies also register multiple domain names related to their trade marks and point them to the company's site;

Temporary sites. Web designers buy domains for their customers and point them temporally to the designer's site or to a default "under construction" page. When the customer's site is deployed the domain starts referencing it.

The detection of duphosts within a web data set can be used to improve information retrieval algorithms. For instance, search engines can avoid presenting the same information published in duphosts as different search results. Crawlers should avoid crawling duphosts to save on bandwidth and storage space. Previous works presented algorithms to detect duphosts within a set of contents harvested from the web (Bharat *et al.*, 2000; da Costa Carvalho *et al.*, 2005). However, preventing a crawler from harvesting duphosts is more difficult than detecting them on a static data set, because a list of duphosts extracted from a previously compiled web collection may not reflect the current state of the web. Sites identified as duphosts may have meanwhile disappeared or started presenting distinct contents.

Next, this section presents heuristics to identify duphosts within a data set and describes an experiment to evaluate their application in crawling. The experimental data set was composed by 3.3 million pages crawled from 66 370 sites. The intersection of content fingerprint signatures between sites was compared to derive a list of pairs of duphosts, where the first site is considered a *replica* of the second one, designated as the *original*. The election of the original within a pair of duphosts is arbitrary because they both provide the same contents. Three heuristics were considered to detect if two sites were duphosts:

SameHome. Both sites present equal home pages. The home page describes the content of a site. So, if two sites have the same home page they probably present the same contents. However, there are home pages that permanently change their content, for instance to include advertisements, and two home pages in the data set may be different although the remaining contents of the sites are equal. On the other hand, there are temporary sites within the data set composed by a single transient "under construction" home page, that in a short notice after the data set was built, begin presenting distinct and independent contents;

SameHomeAnd1Doc. Both sites present equal home pages and at least one other equal content. This approach follows the same intuition than the SameHome for the home pages but tries to overcome the problem of transient duphosts composed by a single page;

Heuristic	Invalid IP	% of duphosts	Precision	Relative coverage
SameHome	6.7%	4.8%	68%	6.6
SameHomeAnd1Doc	4.4%	3.9%	92%	2.1
Dups60	4%	3.7%	90%	2.5
Dups80	4.1%	2.2%	92%	2.0
Dups100	4.7%	0.4%	94%	1.0

Table 5.2: Results from the five approaches to detect duphosts.

DupsP. Both sites present a minimum percentage (P) of equal contents and have at least two equal contents. Between the crawl of the duphosts to build the data set, some pages may change, including the home page. This approach assumes that if the majority of the contents are equal between two sites, they are duphosts. A minimum of two equal contents was imposed to reduce the presence of sites under construction.

Five lists of duphosts were extracted following the heuristics SameHome, SameHomeAnd1Doc, and DupsP considering levels of duplication of 60%, 80% and 100%. The proposed heuristics were evaluated by simulating their application on a crawl executed 98 days after the creation of the data set. Table 5.2 summarizes the obtained results. A DNS lookup was executed for each site on the duphosts lists and excluded those that did not have an associated IP address because a crawler would not be able to harvest them. On average 4.8% of the pairs were no longer valid because one of the duphosts did not have an associated IP address (column *Invalid IP*). The 3^{rd} column of Table 5.2 presents percentage of the total number of sites in the data set that were replicas identified through each heuristic after the IP check. On average, 1 841 replicas were identified, which represents 2.8% of the sites found within the data set. In order to measure the precision of each heuristic, 50 random pairs of duphosts were chosen from each list and visited simultaneously to verify if they still presented the same contents (Precision column). The lowest number of pairs detected (Dups100) was used as baseline to compare coverage (Relative coverage column). The SameHome heuristic achieved the maximum relative coverage (*6.6*) but the lowest precision value

Domain level	% duphosts avoid	% sites lost
2nd level	30%	4%
3nd level	17%	16%

Table 5.3: Consequences of the normalization of the usage of the WWW prefix in site names.

(*68%*). When it was imposed that at least one content besides the home page must exist on both sites (SameHomeAnd1Doc), the relative coverage decreased to *2.1* but the precision improved to *92%*. The Dups100 heuristic detected sites that shared all the contents and it achieved the highest precision of *94%*. The remaining 6% of the pairs referenced sites that were no longer online, although they still had an associated IP address. As the threshold of duplication decreased, more replicas were identified, maintaining the precision over *90%*, (see 2^{nd} and 3^{rd} lines of Table 5.2).

The SameHome heuristic is an inexpensive way to detect duphosts because it requires the comparison of just one page per site. However, it is the most prone to identify transient duphosts originated by single-page sites under construction. The detection of duphosts imposes an overhead on the crawler and avoiding the crawl of a duphost containing one single page may not pay-off. The SameHome-And1Doc overcomes this problem at the cost of comparing more contents per site. The number of duphosts decreases as the threshold of duplicates required between sites increases. At the same time, precision is improved. Due to the permanent changes that occur on the web, I believe that the effectiveness of the proposed heuristics to avoid the crawl of duphosts depends on the the age of the data set used to extract the list of duphosts.

The most common reason for duphosts is the existence of site names that just differ on the prefix "www.". 51% of the names of the duphosts detected on the previous experiments differed just on this prefix. It is recommended that World Wide Web site names begin with the prefix "www." (Barr, 1996). So, one way to avoid the crawl of duphosts is to normalize the URLs to visit by appending the "www." prefix when it is not present. However, there are site names that use a

different prefix and this change could generate invalid site names, excluding valid URLs from the crawl.

An experiment was performed to evaluate the application of this heuristic to prevent duphosts. The experimental data set was composed by two lists of second-level domains (e.g. domain.pt) and third-level domains (e.g. subdomain.domain.pt) from the official registries. A list of home page URLs referencing the domain names and the domain names with the prefix "www." was generated and crawled. Table 5.3 presents the obtained results. The normalization heuristic applied to the second-level domains avoided the crawl of 30% of duphosts and 4% of the sites were excluded because they were not available with a name containing the "www." prefix. For the third-level domains, just 17% of the sites were duphosts due to the usage of the "www." prefix and 16% were lost due to the normalization process. The results suggest that the success of appending the prefix "www." to avoid duphosts depends on the domain level of the site name.

5.4 Conclusions

Web warehouses are powerful tools to help users harnessing web data. This chapter presented the architecture of a WWh that addresses all the stages of web data integration, from the extraction of information from the web to its staging for mining applications. The proposed architecture was validated through the development of a prototype named Webhouse. This chapter focused on the extraction of information from the web and its management within a WWh.

Webhouse was designed considering the characteristics of web data. The information harvested from the Portuguese Web was integrated in Webhouse during its development to study the influence of web characteristics in the design of a WWh. The results obtained while modelling the Portuguese Web and previous works showed that duplication is prevalent. Hence, it should be considered in the design of a WWh. However, existing storage techniques, based on the persistence of content identifiers that enable the elimination of partial duplicates, are not adequate to Web Warehouses, because:

- URLs are highly transient;

- Raise problems for the preservation of the stored data;
- Cannot be efficiently applied in large distributed storage systems.

The main components of Webhouse are the Versus repository and the VN crawler. Versus supports versioning, parallel operation and meta-data management. Previous works on web repositories focused mainly on the design of systems that enable high-performance access to large amounts of contents (hundreds of TB). The research performed to design Versus had a different scope, because it aimed to define a flexible system architecture that enables its application to a broader range of usage contexts. Versus supports complex access methods over the meta-data, such as relational algebra operators.

The extraction of web data was studied to design the VN crawler. This chapter presented a novel architecture for a scalable, robust and distributed crawler, an analysis of methods for partitioning the URL space among the processes of a distributer crawler and techniques to save on bandwidth and storage space. There are important architectural choices that must be made, and the decisions are influenced by the crawling application requirements and characteristics of the web. The standard URL normalization process is insufficient to avoid the download of duplicates and can be improved with additional rules adequate to crawling. Several situations on the web hazardous to crawling have been investigated and solutions to mitigate their effects were proposed. These hazardous situations were presented in the context of crawling, but they affect HTTP clients in general. So, the presented solutions can be used to enhance other systems that process web data, such as browsers or proxies.

The next chapter presents the results obtained while validating the components of Webhouse and shows how they were applied in several usage contexts.

Chapter 6

Validation

The Webhouse prototype was designed to study the influence of web characteristics in the design of Web Warehouses. The proposed architecture was iteratively developed and validated following an Engineering approach (Zelkowitz & Wallace, 1998). Several versions of the system were successively released until its design could not be significantly improved. The final version of the system was subject to several experiments to validate its efficiency.

The data used to validate Webhouse was obtained through controlled and observational methods. The controlled methods were used to validate the Webhouse components individually. Replicated experiments measured differences before and after using a new component. Dynamic analysis experiments collected performance results in the production environment of the tumba! search engine. The execution of experiments based on simulations with artificial data was minimized, because the web is hardly reproducible in a controlled environment and the obtained results might not be representative of the reality.

The data collected to validate Webhouse as a complete system was mainly gathered using observational methods. Webhouse was validated using a case study approach during the development of a search engine for the Portuguese Web. Data was obtained to measure the effectiveness of each new version of the system. The final version of Webhouse was used in several projects to collect feedback on its performance (*field study* validation method).

This chapter presents the obtained results. Section 6.1 presents the evaluation results of the VN crawler. Section 6.2 describes the experimental results of the

Config.	Nr. of machines	CPU (GHz)	Mem. (GB)	Disk speed (rpm	Storage (GB)
1	1	2 x P4-2.4	4	SCSI 10 000	5 x 73
2	4	1 x P4-2.4	1.5	IDE 7 200	2 x 180
3	2	2 x P4-2.4	2	IDE 7 200	5 x 250
4	1	2 x P3-1.26	1	SCSI 15 000	2 x 18
5	1	2 x P3-1.26	4	SCSI 10 000	5 x 73

Table 6.1: Hardware used to support crawling experiments.

Versus Content Manager. Section 6.3 describes applications of the Webhouse prototype. Finally, Section 6.4 draws the conclusions of this validation.

6.1 Crawler evaluation

This section presents the results obtained from experiments performed while harvesting the Portuguese Web with the VN crawler. These crawls ran in June and July, 2005, with the purpose of evaluating the crawler's performance. The analysis of the results enabled the detection of bottlenecks, mal-functions and helped on tuning the crawler's configuration according to the characteristics of the harvested portion of the web.

6.1.1 Experimental setup

Table 6.1 summarizes the hardware configurations of the nine machines used to support VN in the experiments. For instance, a machine with Configuration 1 had two Pentium4 processors running at 2.4 GHz, 4 GB of memory and five SCSI disks running at 10 000 rotations per minute, each one with 73 GB of storage space. All the machines use the RedHat Linux operating system with kernel version 2.4. The machine with Configuration 5 hosts the Global Frontier and the remaining host CNodes. The machines were inter-connected through a 100 Mbps Ethernet and accessed the Internet through a 34 Mbps ATM connection shared with other customers of the data center where they were hosted.

VN was configured to use the home page policy and gather contents from several media types convertible to text. It collected statistics for web characterization, stored original contents for archival and extracted texts for indexing. The thresholds that prevent VN from getting trapped in hazardous situations were determined based on the obtained model for the Portuguese Web (see Chapter 3):

- At most 5 000 URLs were crawled per site;
- The URL depth was limited to five;
- The size of the contents was limited to 2 MBs;
- There was a limit number of 10 duplicates per site, 60 seconds to download each content and 200 characters for the URL length;
- The Crawling Processes were considered dead and restarted by the Watchdogs if they remained inactive for more than five minutes;
- A courtesy pause of two seconds between requests to the same site was respected. Users frequently visit subsequent pages with a time gap of one second (Cockburn & McKenzie, 2001) and a page contains on average 20 embedded images that must also be downloaded (Jaimes *et al.*, 2003; Marshak & Levy, 2003; Wills & Mikhailov, 1999). Assuming that a browser executes these downloads sequentially, the time interval between its requests to the server is just 0.05 seconds. Based on these results, the defined courtesy pause for VN imposes less load on the web servers than human browsing.

6.1.2 Performance comparison

This Section presents a performance comparison based on the results published in related work. Table 6.2 summarizes these results. This comparison must be taken with a grain of salt because the experiments were run using different setups and in different periods of time. Results gathered on different periods of time require different interpretations because the web is permanently evolving. For instance, the Googlebot in 1998 harvested 34 pages per second and VN harvested 25 pages per second, in 2005. However the size of web pages has grown and 34 pages in

Crawler name	Number of machines	Internet (Mbps)	Nr. downloads /second	Data KB/s	% dups.	downloads /URLs	Simulation results
Googlebot	5	?	34	200	?	31%	No
Kspider	1	8	15	360	?	92%	No
Mercator	4	100	112	1 682	8.5%	87%	No
Polybot	4	45	77	1 030	13%	85%	No
Ubicrawler	5	116	?	?	?	?	Yes
VN	9	34	25	738	7%	80%	No
WebBase	1	1 000	6 000	111 000	?	?	Yes
Webgather	4	?	7	?	?	?	No

Table 6.2: Performance comparison between crawlers.

1998 corresponded to 200 KB of data, while 25 documents in 2005 corresponded to 768 KB. The usage of simulations that can be reproduced for different crawlers is too restrictive, because they cannot reproduce the upcoming hazardous situations on the web that degrade the crawler's performance in practice. Simulations cannot realistically test the robustness of the crawlers or if their actions are incommodious to web servers. The download rate of a crawler while harvesting the real web tends do be significantly lower than the one obtained on simulations. The developers of the Googlebot estimated that it could execute 100 downloads per second (600 KB/s) but in practice it did not surpass 34 downloads per second. The performance of the Kspider was initially measured by crawling a fixed set of 400 000 URLs and the obtained results suggested that the crawler could download 618 pages per second (6MB/sec.) using four machines. However, the results obtained from a crawl of eight million pages from the Thai web suggest that the download rate would have been of just 232 downloads/sec using four machines. The WebBase crawler achieved an outstanding performance of 6 000 downloads per second. However, the harvested documents were not written to disk and the paper suggests that the pages were harvested from sites accessible through a 1 Gbps LAN.

Nonetheless, VN's performance while crawling the Portuguese web was compared with the results presented in previous studies. Its performance is close to the one presented by other crawlers but it was hosted on a larger number of machines (nine). However, VN had the additional overhead of extracting and storing meta-data during the crawl. The speed of the connection to the Internet must be considered to analyze crawling performance. The 3^{rd}, 4^{th} and 5^{th} columns of

Table 6.2 show that the most performant crawlers used the fastest connections to the Internet. So, this might be also a reason why VN presented a lower download rate than the most performant crawlers.

The performance of a crawler is usually synonymous of its download rate, but there are other features that should be considered. The *%dups.* column of Table 6.2 presents the percentage of duplicates harvested by the crawlers. The results show that the efforts to minimize the download of duplicates, saving on bandwidth and storage space, yield good results in practice. VN crawled the smallest percentage of duplicates (23% of the contents were duplicates in its first release). A crawler should also minimize the number of visits to URLs that do not reference a downloadable content. The *downloads/URLs* column of Table 6.2 presents the ratio between the number of downloads and the URLs visited. VN was configured as a focused crawler of the Portuguese web and discarded contents considered irrelevant. However, the ratio of *downloads/URLs* is close to the one achieved by the remaining crawlers, which did not discard any contents.

6.1.3 Bottlenecks

The Global Frontier is the central point of coordination of VN. So, it is a potential bottleneck that may compromise the scalability of the crawler. The scalability of the VN architecture was tested by measuring the number of downloads and amount of data crawled within 24 hours, with an increasing number of Crawling Processes spread across several machines. A new machine hosting 20 Crawling Processes was added to the cluster daily. Figure 6.1 presents the obtained results. VN scaled until 160 Crawling Processes executing 2 169 831 downloads in one day (63 GB). This shows that the architecture is scalable and the Global Frontier is not a bottleneck within the observed range.

The duration of the main operations executed during the crawl and the load they imposed on the underlying operating system and hardware were also monitored. The objective of these experiments was to detect bottlenecks in the components of VN. The experimental setup was composed by 4 machines with Configuration 2. Each one hosted an increasing number of Crawling Processes. Figure 6.2

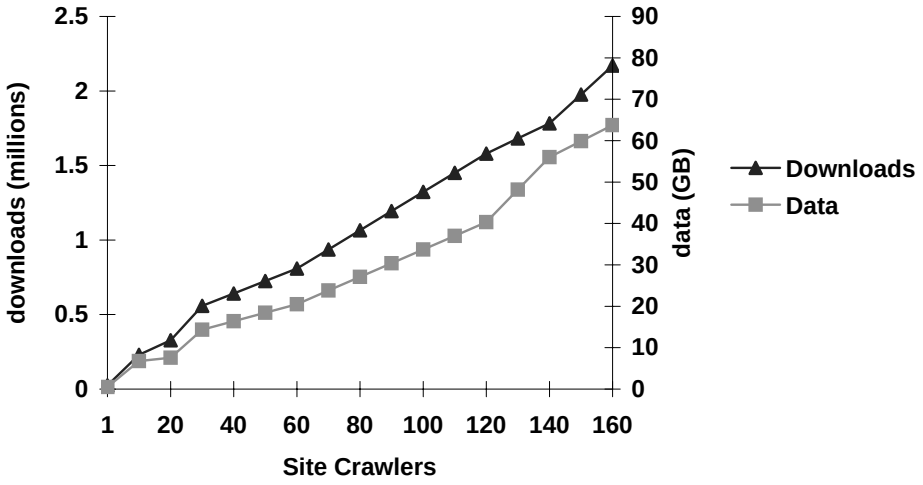


Figure 6.1: Scalability of the download rate with the addition of new CNodes.

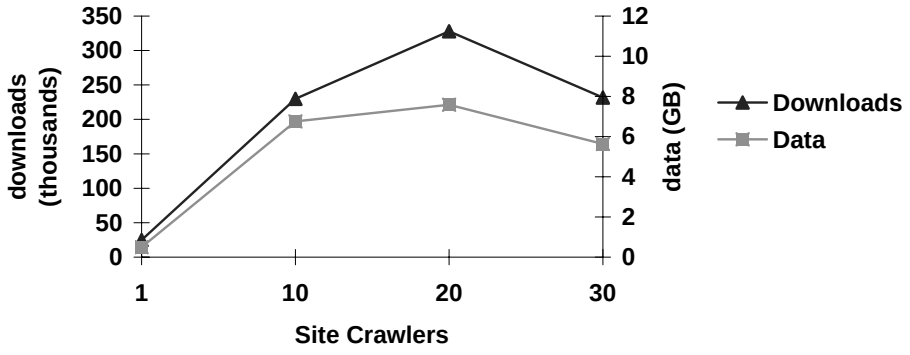


Figure 6.2: Data downloaded vs. Number of Crawling Processes per CNode.

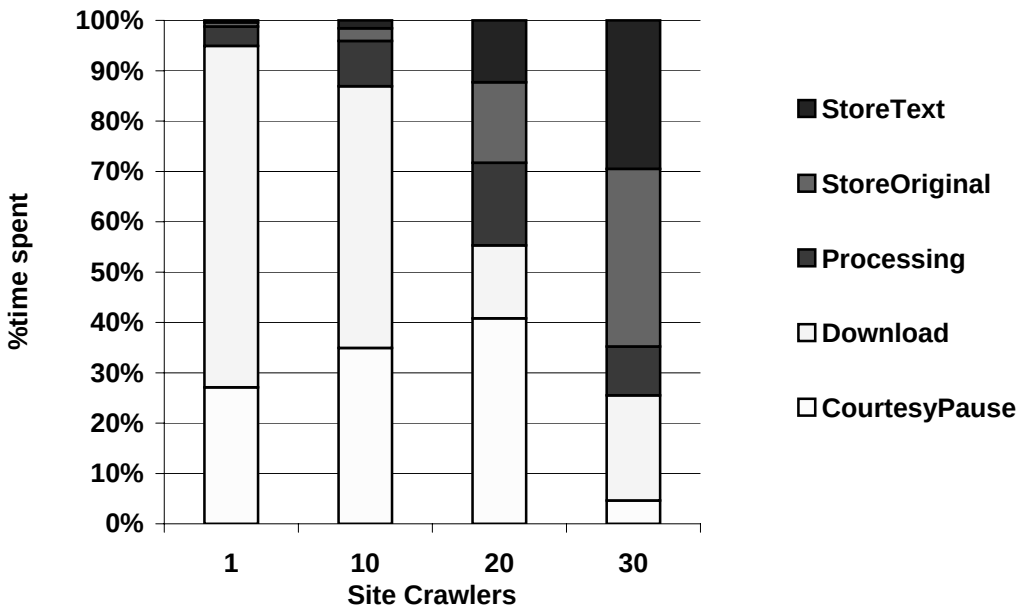


Figure 6.3: Duration of the operations.

presents the amount of data crawled within 24 hours by each set of Crawling Processes. The download rate increased from 1 to 10 Crawling Processes. The system tools indicated that the CPU of the machine was exhausted at this point. However, the number of downloads still increased until 20 Crawling Processes. For 30 Crawling Processes there is a clear performance degradation. The Watchdogs were monitored to detected if dead Crawling Processes considered dead (those inactive for more than five minutes) due to resources starvation caused by the operating system scheduler but no relation between the events was found.

Figure 6.3 shows the average duration of the operations executed by the Crawling Processes. The *StoreOriginal* and *StoreText* series present the time spent to store the contents and the corresponding extracted texts in the Volume. The *Processing* series includes parsing of pages, check-operations, interpretation of the Robots Exclusion Protocol (REP) and meta-data extraction. The *Download* series includes the establishment of a connection to a web server, the download of the header and content. Before executing a request to a site, the CP checks that the configured courtesy pause of two seconds was respected. If it was not, the CP

Number of CPs	Nr. of WR requests p/second (w/s)	Avg. queue length of the requests (avgqu-sz)	Avg. time (ms) for requests to be served (await)	Nr. of sectors written p/sec. (wsec/s)
1	0.5	0.2	36.2	17.2
10	22.6	10.9	410.4	409.4
20	39.7	49.7	811.2	667.7
30	48.0	92.9	1299.8	821.3

Table 6.3: Disk I/O analysis of the Volume using iostat. The name of the column on the iostat report is presented in parenthesis.

sleeps for the remaining time, yielding its execution. The *CourtesyPause* series represents the time elapsed since the CP decided to sleep until it was executed again. The results show that with one CP running on a machine, most of the time is spent on Download operations and executing the CourtesyPause. The percentage of time spent in the storage operations was 1.2%. However, the load of the storage operations increased along with the number of Crawling Processes. For 30 Crawling Processes, the storage operations spent 64.8% of the execution time.

Table 6.3 presents an analysis of the disk accesses. It shows that the disk throughput could not cope with the load imposed by the Crawling Processes. The size of the queue of the requests waiting to be served (3^{rd} column) grew as the number of requests issued to the device increased (2^{nd} column). Surprisingly, it was observed that the time spent in the CourtesyPause increased to 40.8% until 20 Crawling Processes and then dropped to 4.6% for 30 Crawling Processes. The reason for this phenomenon is that when a machine hosts just one CP, the operating system executes the CP immediately after the courtesy pause is reached. However, the time that a CP waited to be executed after performing the courtesy pause increased with the load on the machine because there were other processes scheduled to run before it. For 30 Crawling Processes, the average time for requests to be served was 1.2998 seconds, a value close to the configured courtesy pause of two seconds. Hence, the time elapsed between two requests due to disk latency to the same site frequently achieved two seconds and the Crawling Processes did not need to execute courtesy pauses.

The *Processing* operations use mainly the CPU and access data kept on memory without requiring disk accesses. Hence, they were not significantly affected

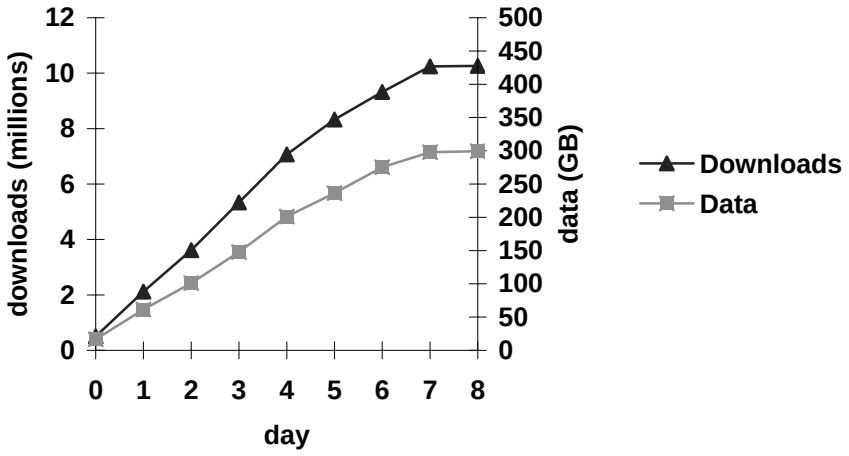


Figure 6.4: Evolution of a Portuguese-web crawl.

by the load imposed on the machines. The obtained results show that disk access became a bottleneck in the crawler before bandwidth was exhausted. The conclusion it that designing a crawler to be deployed on cheap hardware requires additional attention to optimizing disk accesses. I believe that the storage throughput could be improved by using an alternative implementation of the Volumes that would cache larger blocks of data in memory and wrote them sequentially to disk. However, this approach would require the usage of additional memory.

6.1.4 Robustness

The robustness of a crawler is measured by the number of pages downloaded over a large period of time. A crawler may achieve an outstanding download rate in one day and do not crawl a single content on the next, because it crashed due to some unexpected problem. VN was set up to execute a complete crawl of the Portuguese Web to evaluate its robustness. The crawl was bootstrapped with 152 000 seeds generated from a previous crawl. There were 140 Crawling Processes hosted across seven machines.

Figure 6.4 represents the evolution of the crawl. VN collected a total of 10.3 million contents totalling 299.3 GB in eight days. During the crawl several prob-

File extension	Tool	OK	Conversion	Timeout error	Max. size	Type not allowed	404	Other
.ppt, .pps	xlhtml	19%	54%	1%	15%	1%	9%	2%
.xls	xlhtml	19%	60%	13%	1%	0%	6%	1%
.rtf	unrtf	25%	33%	2%	1%	0%	38%	1%
.swf	webcat	36%	53%	1%	0%	5%	4%	1%
.doc	antiword	54%	33%	2%	1%	0%	8%	2%
.ps	ghostscript	59%	6%	25%	3%	0%	5%	2%
.pdf	xpdf	74%	8%	4%	4%	1%	7%	2%
.txt	-	90%	0%	5%	0%	0%	4%	1%
.html, .htm	webcat	94%	0%	0%	0%	0%	4%	2%

Table 6.4: Analysis of text extraction efficiency.

lems occurred on the machines, like operating system crashes, which required reboots or disks that ran out of space. However, the crawl never stopped. VN crawled on average 1.1 million contents per day (37.2 GB) and the peak download rate achieved was 1 734 573 contents (53.2 GB). Figure 6.4 shows that the download rate started to decrease on day 7. The reason for this fact is that VN was configured to crawl first the sites referenced by the seeds and then begin the expansion phase. In this phase, it harvests new sites found through URL extraction to expand the boundaries of the Portuguese Web. The number of pages matching the selection criteria falls sharply once the crawler enters the expansion phase because most of the pages visited outside the .PT domain were not written in the Portuguese language (79%).

The obtained results show that VN presented a steady download rate before the expansion phase, which validates its robustness to hazardous situations on the web as well as operational problems that occurred on the underlying system setup.

6.1.5 Text extraction

VN extracted texts from the harvested contents and classified them to determine if they match the selection criteria (written in the Portuguese language). However, VN could not extract text from the 37% of the contents that were not in the HTML format. The objective of this experiment was to measure the effectiveness of the tools used to extract text from these contents and detect the reasons for their failure.

Table 6.4 describes the file extension of the contents, the tool used to extract text from them, the percentage of contents successfully converted to text and the identified causes of failure. The text extraction process mostly failed for the contents in proprietary formats. Only 19% of the Microsoft Powerpoint presentations (.ppt, .pps) and Microsoft Excel worksheets (.xls) were successfully converted to text. One reason for this is that the owners of the formats frequently release new versions for commercial purposes and the conversion tools used by the VN crawler could not extract text from the most recent versions. The column *timeout* shows that 25% of the postscript contents (.ps) and 13% of the Microsoft Excel worksheets (.xls) could not be converted to text within one minute. The column *max. size* presents the percentage of the contents that were larger than the configured limit of 2 MB. Most of the files were smaller than this limit but 15% of the Powerpoint presentations were larger. These files are large because they usually contain many illustrations. The *type not allowed* column shows that 5% of the files with extension .swf were not identified by the web servers with the expected media type application/x-shockwave-flash, 32% of these files did not return any type on the Content-Type header field, 56% returned the media type application/octet-stream and the remaining 12% presented other media types. The MIME specification states that the media type application/octet-stream is to be used in the case of uninterpreted binary data, in which case the simplest recommended action is to offer to write the information into a file for the user (Freed & Borenstein, 1996b). Hence, this media type does not seem adequate for flash movies, which must be interpreted by a specific software and are written following a proprietary format.

The conclusion is that VN requires more efficient tools for extracting text from contents in proprietary formats. However, there are still contents that cannot be converted to text because their web servers provide erroneous identifications for their media types. When a crawler is setup to execute an exhaustive crawl of contents with different media types, the limit thresholds should be configured according to each media type. For instance, the maximum size allowed for Powerpoint presentations should be higher than for HTML pages.

Parameter	Limit	% Sites	% URLs
Nr. of duplicates	10 duplicates	1.6%	–
Nr. of URLs	5000 URLs	0.8%	–
URL depth	5	11.3%	–
Download time	60 seconds	–	0.4%
Size	2 MB	–	0.1%
URL length	200 characters	–	1%

Table 6.5: Percentage of sites and URLs that exceeded limit values.

6.1.6 Tuning thresholds

A portion of the web presents specific characteristics and a crawler should be tuned to improve its performance according to them. On the previous experiments, VN was configured with limit values for several parameters to avoid hazardous situations. Periodically, these thresholds must be reviewed and updated to reflect the evolution of the web. However, the decision to update requires human reasoning. If the limits were achieved due to hazardous situations they should not be updated.

Table 6.5 describes the limits imposed and the percentage of URLs or sites whose crawl was stopped by reaching one of the limits. The maximum number of duplicates was overcome by 1.6% of the sites. A sample of 10 of these sites was humanly analyzed and they all contained spider traps. The number of sites that contained more than 5 000 URLs was just 0.8%, which confirms previous findings (Brandman *et al.*, 2000). The maximum depth was achieved in 11.3% of the sites. A visit to a sample of these sites revealed that the limit depth was not related to any hazardous situation. Hence, this threshold should be updated. The results obtained by Liu (1998) suggested that the timeout of a web client should not be longer than 10 seconds. Table 6.5 shows that only 0.4% of the Portuguese URLs took longer than 60 seconds to be downloaded and processed to extract meta-data. These results suggest that the performance of the crawler could be improved by reducing the timeout value, without excluding valid contents. Only 0.1% of the contents achieved the maximum size of 2 MBs and just 1% of the URLs were longer than 200 characters.

In a nutshell, the conclusion derived from this experiment is that the limits for most of the parameters were adequate, except for the maximum URL depth, which should be increased on subsequent crawls of the Portuguese Web.

6.2 Versus content management

Web Warehouses require storage systems able to address the specific characteristics of web collections. One peculiar characteristic of these collections is the existence of large amounts of duplicates. The Versus Content Manager was designed to efficiently manage duplicates through a manageable, lightweight and flexible architecture, so that it could be easily integrated in existing systems.

This section presents the results gathered from four experiments ran on the Content Manager against NFS. These replicate its application in several usage contexts. NFS was chosen as baseline, because it is widely known and accessible, enabling the reproducibility of the experiments.

6.2.1 Experimental setup

The experimental setup consisted on Pentium4 at 2.4 GHz machines with Red Hat Linux 9.0. The disks were IDE ATA 100, 7200 rpm, managed by a software RAID 5 controller. Clients in Java were developed for each one of the experiments. The Content Manager clients executed the operations through the invocation of methods available in the API library and the NFS clients used the JDK IO library to execute equivalent operations on a remote directory mounted on the local file system. The data set used in the experiments was composed by 1 000 distinct HTML pages with a total size of 19.2 MB gathered from a previous crawl of the Portuguese Web.

6.2.2 Retrieving

In this experiment, a Versus Content Manager volume server was loaded with the pages from the data set and all the generated contentkeys were kept. Then, these contentkeys were split among the clients that retrieved the corresponding contents from the Content Manager in parallel. In each measurement, a new machine

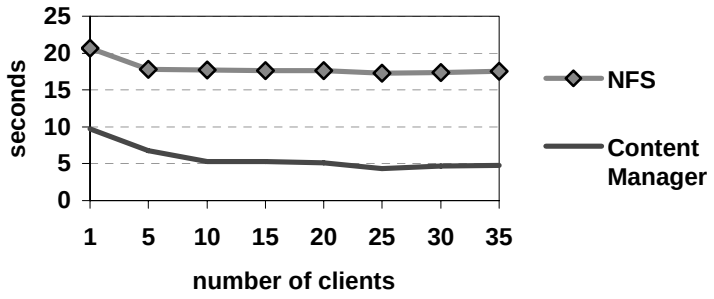


Figure 6.5: NFS read vs. Versus Content Manager retrieve.

hosting five clients was added to stress the server. For the NFS test, an equivalent procedure was followed: the files were stored keeping the corresponding paths and the clients were launched to read them in parallel. Before each measurement, the NFS volumes were re-mounted to prevent caching of the files.

Figure 6.5 presents the total time that the clients took to read all the contents. The results show that the Content Manager is on average 68% faster than NFS for read operations. The total time for executing the task remained constant for more than 10 parallel clients with the Content Manager and more than five clients with NFS, which shows that the NFS server exhausted its response capacity sooner than the Content Manager volume server.

6.2.3 Deleting

The data set was loaded into the Versus Content Manager and NFS volumes and the references to the contents were split among the clients, as described in the previous experiment. First, the clients were launched in parallel to delete the contents from the data set. Then, the data set was loaded twice in the Content Manager to create a situation where all the contents stored were duplicates and the clients were relaunched to delete them.

Figure 6.6 compares the total time that the clients took to delete the contents. The Content Manager is on average 67% faster than NFS when deleting duplicates because only the reference counter is decremented in this case. When deleting the block it is 60% faster than NFS. I believe that the Content Manager outperforms NFS because it uses a lighter protocol of communication and does

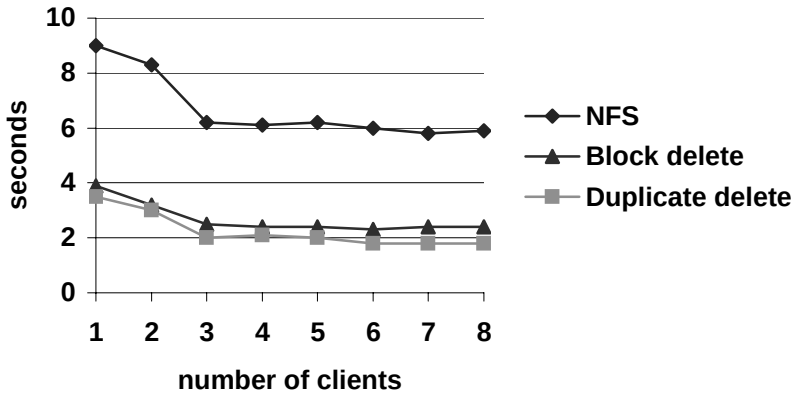


Figure 6.6: NFS remove vs. Versus Content Manager delete.

not have the overhead of maintaining caches of files and associated meta-data, such as permissions. Both the NFS and Content Manager servers reached their maximum throughput with 10 clients.

6.2.4 Storing

The data set was split among the clients that stored it in the Versus Content Manager and NFS. For each set of clients, the data set was stored twice in the Content Manager to compare the times spent creating blocks to store new contents against adding references to duplicates. The NFS volume was exported with the *sync* option, which does not allow the server to reply to requests before the changes made by the request are written to disk (same behavior as the Content Manager). The experiments on NFS had to be restarted several times due to crashes of the Java Virtual Machines that were running the clients. The causes of this problem could not be exactly determined, but as this situation never occurred with the Content Manager clients, I suspect that the crashes were due to an incompatibility between the JDK IO library and NFS.

Figure 6.7 presents the total times spent storing the contents in Content Manager and NFS. As expected, the store operation is faster for storing duplicates than for storing new contents, because they are not transferred from the clients

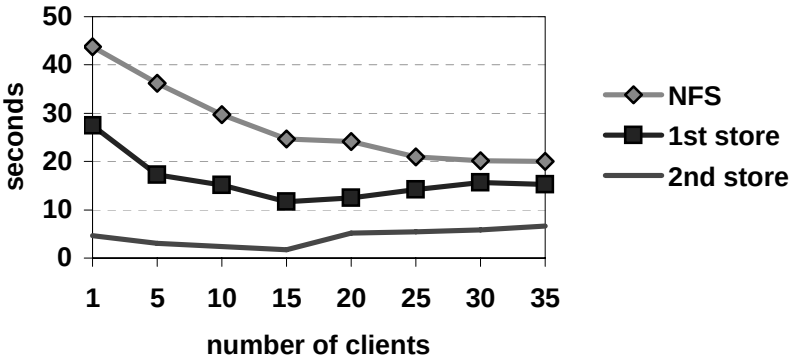


Figure 6.7: NFS save vs. Versus Content Manager regular store.

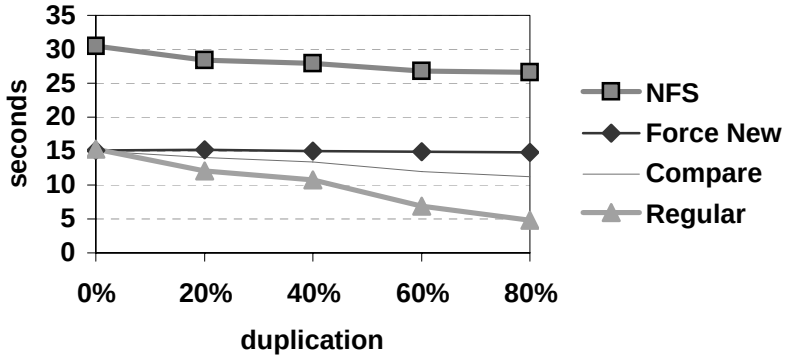


Figure 6.8: Time spent to store the data set presenting increasing levels of duplication.

to the volume servers and the creation of new blocks is not required. For duplicates the Content Manager outperformed NFS by 82%. For the new ones, it outperformed NFS on average by 50%, from 5 to 20 clients. However, it seems to reach its saturation point at 15 clients, while NFS stands 25 clients. The reason for this is that as the Content Manager is faster than NFS, it reached the server's disk throughput peak performance earlier as the number of clients rose.

6.2.5 Semantics of the store operation

The objective of this experiment was to measure the performance of the three different modes available for the store operation: *regular*, *compare* and *force-new*. The level of duplication was gradually increased by 20% within the data set and 10 clients were launched. Each client stored the data set using in turn each one of the three modes.

Figure 6.8 presents the obtained results. The semantics of the force-new mode is similar to the NFS write operation, given that it does not put any effort in eliminating duplicates. However, it took almost half of the time to finish the task. As the level of duplication increased, the compare and regular modes presented the best results. The compare mode is slower than the regular, because it avoids fake duplicates through bytewise comparison, while the regular mode compares only the sizes of the contents. When there was not replication within the data set, all the three modes performed the same way. The conclusion is that the overhead of detecting duplicates within a volume is insignificant. Besides saving disk space, this experiment showed that the proposed mechanism for the elimination of duplicates increases the storage throughput of the system when it manages collections containing duplicates.

6.3 Webhouse applications

This section describes the main applications of Webhouse, covering the features and operation of the tumba! search engine. It also describes how Webhouse was used in several other research experiments, discusses selection criteria to populate a national web archive and describes the use of Webhouse in a web archive prototype.

6.3.1 The tumba! search engine

The lack of context in queries to global search engines causes that users in different communities expect different results for the same query (Silva, 2003). Tumba! is a search engine designed and developed by the XLDB Group of the University of Lisbon (`xldb.fc.ul.pt`). Its main objective is to provide better search results

to the users of the Portuguese Web, by using information sources about Portugal and applying knowledge about the characteristics of the Portuguese Web contents in the design of the system.

In November, 2002 the tumba! search engine was released to the public as a free public service (available at www.tumba.pt). The service has been improved and maintained by graduate students of the research group. Tumba! supports several features besides finding the pages that contain a given term:

Advanced search. Tumba! supports several operators that enable the combination of several words in a query, so that users can submit more expressive queries. It supports exact searches to find pages that contain a specific phrase, site searches that enable the restriction of the search to a given domain and Boolean operators among search terms;

Content properties. Each indexed content has associated meta-data required by the search engine. Some of these meta-data can also be accessed by the tumba! users. For instance, users can access the stored version of a content (cache) and find the pages that link to and from it;

Related pages. This service provides a ranked list of pages, related to a content (Martins, 2004);

Clustering. Some query terms are vague or have several different meanings. The clustering feature groups search results in categories so that users can choose which ones stratify best their information needs (Martins & Silva, 2003);

Geographical searches. There are pages on the web that have information about geographical locations. Tumba! identifies the geographical scope of pages and enables searching within a geographical location, such as, finding the home page of a traditional food restaurant in Lisbon (Silva *et al.*, 2006);

Mobile device interface. A simpler search interface adapted to devices such as WAP phones or PDAs. This feature is specially useful to help users finding pages of services close to their location (Freitas *et al.*, 2006);

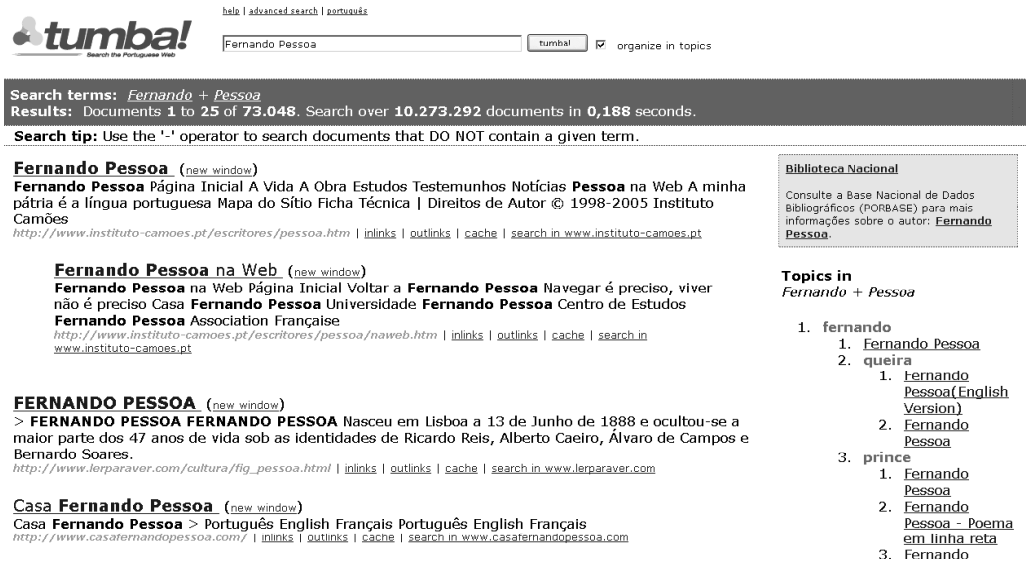


Figure 6.9: Tumba! web interface.

Query spell-checker. Misspelled words are common in queries to search engines. The spell-checker selects the best choice among the possible corrections for a misspelled term and suggests it to the users (Martins & Silva, 2004a);

Connection to external tools. Tumba! is connected to external services that complement its searching features. Users can access free online dictionaries of Portuguese, hear the pronunciation of terms (a cooperation with INESC's L2F group) or find publications of Portuguese authors in a database of the National Library (Borbinha *et al.*, 2003).

Figure 6.9 presents an example a search result provided by tumba!. Users submit the search terms in a web form. They can use advanced search operators to relate several terms and choose if they want to receive clustered results by ticking the check-box next to the form box. The page titles and the snippets of the pages where the search terms occur are presented on the left side of the results page. Bellow each snippet the users can follow links that enable the access to the content properties or related pages. On the right side of the results page there are presented links to the National Library database, geographical and clustered results.

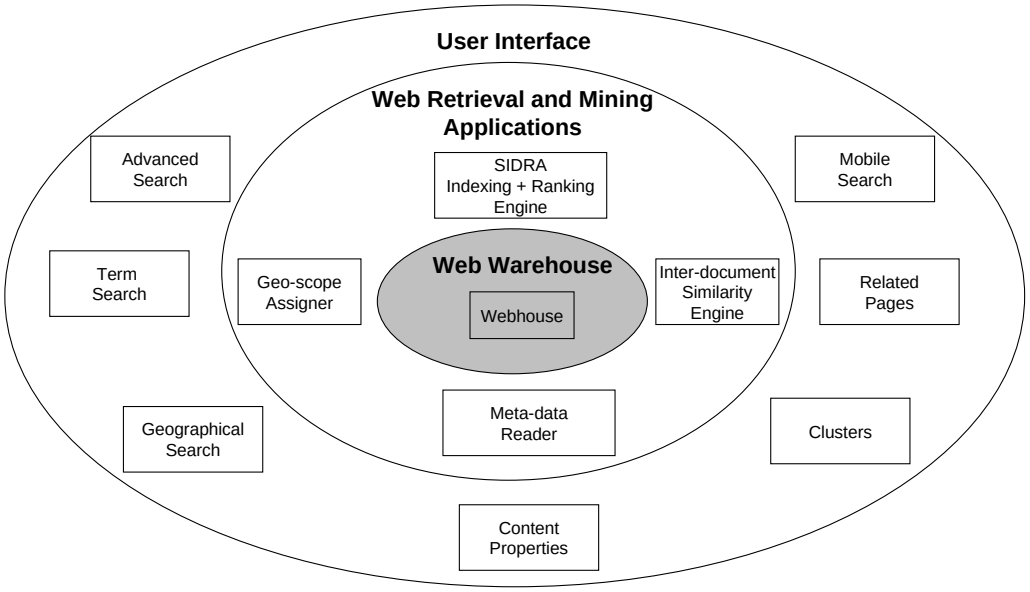


Figure 6.10: Webhouse role in the tumba! search engine.

Figure 6.10 presents the architecture of the tumba! search engine. Webhouse is one of its main components. It is responsible for harvesting the contents from the Portuguese Web, store them along with the correspondent meta-data and, after each crawl is finished, provide access to all the web mining applications that process the stored data. For instance, the SIDRA system accesses the web data to build indexes that enable quick searches (Costa, 2004). The models derived for the Portuguese were also applied in the ranking algorithms to provide better search results. The Geo-scope assigner processes the text extracted from each content to derive its geographical scope.

The tumba! search engine was a valuable test bed for Webhouse validation, because it has several applications with different web data processing requirements. Tumba! ran with success for three years using several releases of Webhouse. On average it provided around 4 000 searches per day and its users came from all the world, but mainly from Portugal and Brazil.

6.3.1.1 Supporting research experiments

The data gathered during the development of Webhouse and the system itself were used as experimental platform to support research in other fields. The WPT03

is a textual corpus compiled from the Portuguese Web using Webhouse (Cardoso *et al.*, 2005c). The main objective of the creation of this corpus is to provide an experimental test collection to be used in Natural Language Processing. The WPT03 package also includes tools to facilitate the automatic processing of the data and a query log from the tumba! search engine compiled during six months. The WPT03 corpus was used in several research studies, contributing to:

- The execution of a statistical study of linguistic metrics of the Portuguese contents (Martins & Silva, 2004b);
- The creation of a database containing lexicon tables that can be used in the construction of gazetteers, finding semantically related elements and implementing word-sense disambiguation techniques based in clustering algorithms (Sarmento, 2006);
- The identification of geographical entities on the Portuguese Web (Santos & Chaves, 2006);
- The evaluation of web partitioning methods based on geographical criteria to distribute URLs in a distributed crawler (Exposto *et al.*, 2005);
- The evaluation of a question answering system (Costa & Sarmento, 2006).

A new corpus containing data gathered with Webhouse in 2005 (WPT05) is scheduled for release in 2007.

The Cross-Language Evaluation Forum (CLEF) initiative aims to support the development and enhancement of digital library applications that support European cross-language retrieval (DELOS, 2006). It has been organizing system evaluation campaigns annually since 2000, creating test-suites that enable system benchmarking and stimulating collaboration between researchers. Each campaign is composed by several tasks directed to evaluate different type of systems. The CLEF organization provides a test collection to the participants and they load it to their systems to perform the designated task. In the end, the results obtained by each system are compared and published. The XLDB Group has been using Webhouse has experimental platform to store and access the test collections of CLEF tasks since 2004:

Ad hoc. This task aims to evaluate system performance on a multilingual collection of plain text news documents but the participants can also choose to participate on monolingual subtasks. The XLDB Group participated in the mono and bilingual tasks to evaluate several components of the tumba! search engine (Cardoso *et al.*, 2004, 2005a, 2006);

WebCLEF. This task uses a test collection built through the crawl of governmental sites in Europe (EuroGOV collection) to evaluate retrieval systems in a web environment. The XLDB Group participated in this task and detected problems with the EuroGOV collection that may have influenced the evaluation results (Santos & Cardoso, 2005);

GeoCLEF. Geographical Information Retrieval systems (GIR) retrieve information considering spatial awareness. GeoCLEF aims to evaluate these systems in a multi-lingual environment. The XLDB Group has participated in this task since its first edition in 2005 (Cardoso *et al.*, 2005b; Martins *et al.*, 2006). Besides providing a experimental platform for this task, Webhouse provided data about the Portuguese Web that contributed to the generation of a geographic knowledge base and the creation of the first geographical ontology for Portugal (Chaves *et al.*, 2005).

6.3.2 The Tomba web archive

Never before in the history of mankind so much information was published. However, it was never so ephemeral. Web documents such as news, blogs or discussion forums are valuable descriptions of our times, but most of them will not last longer than one year (see Chapter 4). If the current contents are not archived, the future generations could witness an information gap in our days. The archival of web data is of interest beyond historical purposes. Web archives are valuable resources for research and could also provide evidence in judicial matters when offensive contents are no longer available online.

The archival of conventional publications has been directly managed by human experts, but this approach cannot be directly transposed to the web, given its size and dynamics. I believe that web archiving must be performed with minimal

human intervention, but this is a technologically complex task. The Internet Archive collects and stores contents from the world-wide web. However, it is difficult for a single organization to archive the web exhaustively while satisfying all needs, because the web is permanently changing and many contents disappear before they can be archived. As a result, several countries are creating their own national archives to ensure the preservation of contents of historical relevance to their cultures (Day, 2003).

Conceptually, a web archive is an instance of a WWh. It harvests and stores large collections of contents gathered from the web and provides access to them. However, a web archive requires specific selection criteria to collect contents of historical relevance that must be preserved for long periods of time. This section discusses strategies for selecting contents for a national web archive and presents a system's architecture based on Webhouse. This architecture was validated through a prototype named Tomba that was loaded with a total of 57 million contents (1.5 TB) gathered from the Portuguese Web during 4 years.

6.3.2.1 Selection criteria for historical relevance

Web archivists define strategies to populate web archives according to the scope of their actions and the resources available. An archive can be populated with contents delivered from publishers or harvested from the web. The delivery of contents published on the web works on a voluntary basis in The Netherlands, but it is a legislative requirement in Sweden (National Library of Australia, 2006). However, the voluntary delivery of contents is not motivating for most publishers, because it requires additional costs without providing any immediate income. On the other hand, it is difficult to legally impose the delivery of contents published on sites hosted on foreign web servers, outside a country's jurisdiction. The absence of standard methods and file formats to support the delivery of contents is also a major drawback, because it inhibits the inclusion of delivery mechanisms in popular publishing tools. Alternatively, a web archive can be populated with contents periodically harvested from the country's web. However, defining the boundaries of a national web is not straightforward and the selection policies are controversial.

The broad selection criteria adopted for the tumba! search engine (see Chapter 3) may not be adequate to populate a national web archive. The main objective of a search engine is to provide relevant and up-to-date results to its users and not to preserve data for historical purposes. A search engine can adopt broad selection criteria that include many irrelevant contents because used ranking mechanisms exclude those contents from search results. A search engine has storage requirements less demanding than a web archive because its web collection is periodically refreshed and the old contents are discarded. On its turn, a web archive builds its web collection incrementally and must adopt a narrower selection criteria to select contents with historical relevance and save storage space.

An experiment was performed to discuss alternative selection criteria for a national web archive. A crawl of 10 million Portuguese contents performed for the tumba! search engine in July, 2005 was used as baseline to compare various selection policies. The analyzed selection criteria were derived from controversial subjects among the web archiving community.

Exclude Blogs. Blogs have been introduced as frequent, chronological publications of personal thoughts on the web. Although the presence of blogs is increasing, most of them are rarely seen and quickly abandoned. According to a survey, "the typical blog is written by a teenage girl who uses it twice a month to update her friends and classmates on happenings on her life" (Perseus Development Corp., 2004), which hardly matches the common requirements of a document with historical relevance. On the other hand, blogs are also used to easily publish and debate any subject, gaining popularity against traditional sites. Blogs that describe the life of citizens from different ages, classes and cultures will be an extremely valuable resource for a description of the current times (Entlich, 2004).

A site was identified as a blog in the baseline if it contained the string "blog" on the site name. The obtained results showed that 15.5% of the contents in the baseline would have been excluded from a national web archive if Blogs were not archived. 67% of the blog contents were hosted under the .COM domain and 33% were hosted on blogs under the .PT domain.

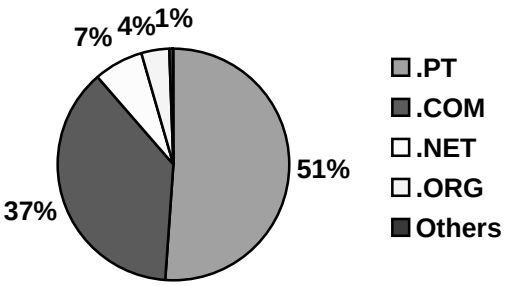


Figure 6.11: Distribution of contents per domain from the Portuguese Web.

Country code Top Level Domains. There are two main classes of top-level domains (TLD): generic (gTLDs) and country code (ccTLDs). The gTLDs were meant to be used by particular classes of organizations, such as .COM for commercial organizations, and are administrated by several institutions world wide. The ccTLDs are delegated to designated managers, who operate them according to local policies adapted to best meet the economic, cultural, linguistic, and legal circumstances of the country. Hence, sites with a domain name under a ccTLD are strong candidates for inclusion in a national web archive. However, this approach excludes the contents related to a country hosted outside the ccTLD. Figure 6.11 presents the distribution of contents from the Portuguese Web per domain and shows that 49% of its contents are hosted outside the ccTLD .PT. There were identified two main reasons found for finding such a large amount of Portuguese contents outside the ccTLD. The first is that the gTLDs are cheaper and faster to register. The second is that most popular blogging sites are hosted under the .COM domain, which increases the number of contents from a national web hosted outside the country code TLD. For instance, Blogspot held 63% of the Portuguese blogs.

Physical location of web servers. The RIPE Network Management Database provides the country where an IP address was firstly allocated or assigned (RARE, 1992). One could assume that the country's web is composed by

MIME type	avg size (KB)	%docs.
text/html	24	**61.2**%
image/jpeg	32	**22.6**%
image/gif	9	**11.4**%
application/pdf	327	1.6%
text/plain	102	0.7%
app'n/x-shockwave-flash	98	0.4%
app'n/x-tar	**1,687**	0.1%
audio/mpeg	**1,340**	0.04%
app'n/x-zip-compressed	**541**	0.1%
app'n/octet-stream	454	0.1%
other	129	1.8%

Table 6.6: Prevalence of media types on the Portuguese Web.

the contents hosted on servers physically located on the country. However, the obtained results showed that only 39.4% of the IP addresses of the baseline were assigned to Portugal.

Select media types A web archive may select the types of the contents it will store depending on the resources available. Preservation strategies must be implemented according to the format of the contents. For instance, preserving contents in proprietary formats may require having to preserve also the tools to interpret them. The cost and complexity of contents preservation increase with the variety of media types archived. Hence, web archivists focus their efforts on the preservation of contents with a selected set of media types. Table 6.6 presents the coverage of selection strategies according to the selected media types. 83 different media types have been identified. However, a web archive populated only with HTML pages, JPEG and GIF images would cover 95.2% of a national web.

Ignore robots exclusion mechanisms Web archives and search engines use crawlers to gather contents, but publishers may forbid their harvesting through the Robots Exclusion Protocol and ROBOTS meta-tag (Koster, 1994). Search engines present direct links to the pages containing relevant information to answer a given query. Some publishers only allow the crawl

of the site's home page to force readers to navigate through several pages containing advertisements until they find the desired page, instead of finding it directly from search engine results. One may argue that the public interest of preserving history should overcome private interests and the archive crawlers should ignore these exclusion mechanisms to achieve the maximum coverage of the web. However, the exclusion mechanisms are also used to prevent hazardous situations as discussed in Chapter 5. So, ignoring the exclusion mechanisms may degrade the performance of an archive crawler.

The obtained results show that 19.8% of the Portuguese sites contained the Robots Exclusion Protocol file (robots.txt), but they forbade the crawl of just 0.3% of the URLs. 10.5% of the pages contained the ROBOTS meta-tag but only 4.3% of them forbade the indexing of the page and 5% disallowed the following of links. The obtained results suggest that ignoring exclusion mechanisms does not significantly increase the coverage of a national web crawl. However, this behavior may degrade the crawler's performance, because exclusion mechanisms are also used to prevent crawlers against hazardous situations.

6.3.2.2 Architecture

Figure 6.12 describes the architecture of the Tomba web archive. The components in grey are part of the Webhouse software package.

Gatherer. Collects contents and integrating them in the archive. The *Gatherer*, composed by the *Loader* and the *VN crawler*, integrates web data in the Repository. The *Loader* was designed to support the delivery of contents by publishers and receive previously compiled collections of contents. Ideally, VN would crawl a page and the referenced contents sequentially to avoid that some of them become unavailable meanwhile. However, sequentially crawling all the contents referenced by a page degrades the crawler's performance, as discussed in Chapter 5. Crawling the contents of one site at a time in a breadth-first mode and postponing the crawl of external contents until the corresponding sites are visited, is a compromise solution that ensures that the majority (71%) of the embedded contents internal to each

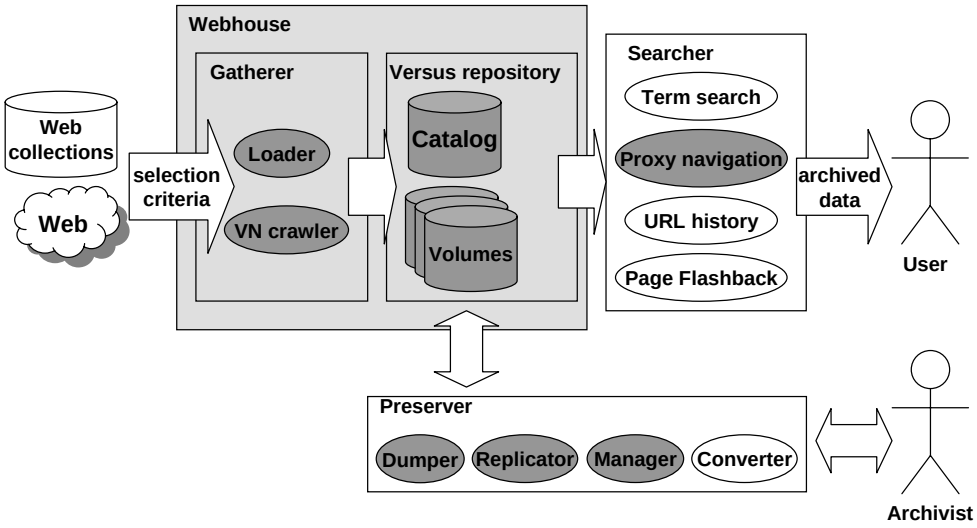


Figure 6.12: Architecture of the Tomba web archive.

site are crawled in a short notice, without requiring additional bandwidth usage (Marshak & Levy, 2003);

Versus repository. Stores the contents and their correspondent meta-data. The general data model of the Catalog (see Chapter 5) was mapped to the context of web archiving. The Source corresponds to an URL and a Version corresponds to a crawl of the referenced content. Each Layer corresponds to a crawl of the web. The Property lists keep meta-data about each Version, such as the date of crawl or its media type. The Content references the contents in their original format and the Facets are used to keep the contents in obsolete formats in alternative up-to-date formats. There are contents that can be used in the preservation of others. For instance, a page containing the specification of the HTML format could be used in the future to interpret contents written in this format. The Reference class enables the storage of associations of these contents. Each new snapshot of the web is kept in the Group Workspace during the crawl and then it is checked-in into the Archive Workspace, where it is appended to the previous ones;

Preserver. Provides tools to manage and preserve the archived data. Repli-

cation is crucial to prevent data loss and ensure the preservation of the archived contents. The replication of data across mirrored storage nodes must consider the available resources, such as disk throughput and network bandwidth. A new content loaded into the archive can be immediately stored across several mirrors, but this is less efficient than replicating contents in bulk. Considering that an archive is populated with contents crawled from the web within a limited time interval, the overhead of replicating each content individually could be prohibitive. The *Replicator* copies the information kept in a Volume to a mirror in batch after each crawl is finished. The *Dumper* exports the archived data to a file using three alternative formats:

1. The WARC format, proposed by the Internet Archive to facilitate the exportation of data to other web archives (Kunze *et al.*, 2006);
2. An XML based format to enable flexible automatic processing;
3. A textual format with minimum formatting created to minimize the space used by the dump file.

The dissemination of the archived contents as public collections is an indirect way to replicate them outside the archive, increasing their chance of persisting into the future. The main obstacles to the distribution of web collections are their large size, the lack of standards to format them in order to be easily integrated in external systems, and copyright legislation that requires authorization from the authors of the contents to distribute them. Obtaining these authorizations is problematic for web collections having millions of contents written by different authors. The archived contents in obsolete formats must be converted to up-to-date formats to maintain their contents accessible. The *Converter* iterates through the contents kept in the Repository and generates Facets containing alternative representations in different formats. The *Manager* allows a human user to access and alter the archived information;

Archivist. Human intervention must be minimized in web archiving. However, it cannot be completely excluded. The Archivist is a human expert that

Figure 6.13: Tomba web interface.

manages preservation tasks and defines selection criteria to automatically populate the archive;

Searcher. Enables human users to easily access the archived data. It provides four alternative access methods: *Term Search*, *Proxy Navigation*, *URL History* or *Page Flashback*. The Term Search method finds contents containing a given term. The contents are previously indexed to speed up the searches. The Proxy Navigation method enables browsing the archive using a web proxy. The URL History method finds the versions of a content referenced by an URL. The Page Flashback mechanism enables direct access to the archived versions of a content from the web being displayed on the browser. The user just needs to click on a toolbar icon and the versions of the page archived in Tomba will be immediately presented.

6.3.2.3 Web interface

Designing a web interface to access archived data is not straightforward. Figure 6.13 presents the public web interface of Tomba, implemented to support the URL history access method (available at `tomba.tumba.pt`). This section

discusses the requirements of the methods used to access the archived data and presents the solutions adopted on the Tomba web interface.

Search for URL aliases. Navigation within the archive begins with the submission of an URL in the input form of the Tomba home page. In general, multiple different URLs reference the same resource on the web and it may seem indifferent to users to submit any of them. If only exact matches on the submitted URL were accepted, some contents might not be found in the archive. Hence, Tomba expands each submitted URL to a set of URLs that are likely to reference the same resource, and then searches for them. For instance, if a user inputs the URL www.tumba.pt, Tomba will look for contents harvested from the URLs: www.dlig.org/, dlib.org, www.dlib.org/index.html, www.dlib.org/index.htm, www.dlib.org/index.php, www.dlib.org/index.asp;

Time awareness. An archived content may contain code that when interpreted by a browser redirects the user to the current web. For instance, the HTML meta-tag "refresh" embedded in a page redirects the browser to a different URL after the page was loaded. It is not possible to anticipate all the situations that may cause redirections to outside the archive. So, users must be aware of the time space of the contents they are browsing. The Tomba visualization interface displays on the left frame the archive dates of the available versions of a content. The most recent version of the content is initially presented on the right frame and users can switch to other versions by clicking on the associated dates, enabling a quick tracking of the evolution of a content. The presentation of the archived contents within a frame allows to visually identify a redirection to the current web because these contents are presented outside the frame. This approach differs from the one adopted by the Internet Archive that presents all the contents in a new window and a user must detect the redirections to the current web by examining the host name of the URL presented on the address bar of the browser;

Reproduce original layout and link navigation. The navigation within the archived data should reproduce the behavior of the web. However, the conditions in which the contents were originally published on the web may not be reproducible by the archive. For instance, the archive cannot reproduce the results of queries to databases. Nonetheless, web archives should try to mimic the original layout of the contents and enable users to follow links to other contents within the archive, when activating a link on a displayed page. For this purpose, Tomba archives the contents in their original formats, but modifies them before presentation. The contents are parsed and the URLs to embedded images and links to other contents are replaced to reference the archived contents. When a user clicks on a link, Tomba picks the version of the URL in the same Layer of the referrer content and displays it on the right frame along with the references to other versions on the left frame. A user may retrieve an archived content without modifications by checking the box *original content* below the submission form (Figure 6.13). This is an interesting feature for authors who want to recoverer old versions of a content;

Correct erroneous meta-data. There are web servers that provide erroneous meta-data on the HTTP headers. Some of these meta-data can prevent the content from being correctly interpreted and presented to the user. For example, consider a page written using the ISO-8859-1 character set encoding. If the web server indicates that the page was written using the UTF-16 character set encoding, a browser would be unable to correctly present the page. A web archive tries to reproduce an archived content by providing the meta-data received from the web servers. However, if this information was incorrect and the archived content is not correctly presented to the user, it would seem that there is a mal-function within the archive. Hence, a web archive should try to correct erroneous meta-data before presenting contents to its users. Tomba tries to correct erroneous character set identifications before presenting contents through the analysis of the meta-tags embedded in the pages;

Block access to contents. A web archive loaded with large amounts of data gathered from the web may contain offensive contents that should not be reproduced by the archive. For instance, some contents should not be reproduced due to the country's copyright legislation. Hence, the user interface must be able to quickly block access to contents. Tomba provides an exclusion mechanism that allows to block immediately the public access to a list of contents.

The URL History access method has three main limitations. First, users may not know which URL they should input to find the desired information. Second, the short life of URLs limits their history to a small number of versions. The Tomba prototype was loaded with 10 incremental crawls of the Portuguese Web but on average each URL referenced just 1.7 versions of a content. Third, the replacement of URLs may not be possible in pages containing format errors or complex scripts to generate links. If these URLs reference contents that are still online, the archived information may be presented along with current contents. The Term Search and Navigation complement the URL History, but they have other limitations. The Term Search finds contents independently from URLs but some contents may not be found because the correspondent text could not be correctly extracted and indexed (Drugeon, 2005). The Navigation method enables browsing the archive without requiring the replacement of URLs because all the HTTP requests issued by the user's browser must pass through the proxy that returns contents only for archived contents. However, it might be hard to find the desired information by following links among millions of contents.

6.4 Conclusions

This chapter presented the experimental results obtained to validate Webhouse. It described experiments ran to evaluate the performance and scalability of the VN crawler and the Versus Content Manager, and the application of Webhouse in several real usage scenarios. The obtained results showed that the performance of Webhouse components enable the use of this WWh on data mining applications in a broad scope of usage contexts.

VN has been used in the past 4 years, having crawled over 54 million contents (1.5 TB). Its crawling activities originated just two complaints during this period of time, which shows that the adopted politeness measures were successful. The obtained results showed that VN is robust and scalable. The upcoming of large capacity disks at low prices helped crawlers extending its storage capacity. However, the storage throughput did not follow the pace of the disk's capacity and latency is a potential bottleneck that must be carefully addressed in the design of crawlers.

The experiments ran on the Versus Content Manager showed that it outperformed significantly NFS in read, write and delete operations. The Content Manager is platform-independent and runs at the application level. Its source code and the clients used to gather the presented results are available for download at http://webstore.sourceforge.net/. The obtained results showed that the algorithm for eliminating duplicates saves space and increases storage throughput.

Webhouse was successfully applied to support a search engine for the Portuguese Web, publicly available at www.tumba.pt since 2002, which provides a broad scope of search features. Webhouse was used as experimental platform to develop and evaluate data mining applications. Plus, the data gathered during the development of Webhouse enabled the creation of test collections used in research activities.

Webhouse was also used to support a national web archive prototype named Tomba (available at tomba.tumba.pt). This chapter discussed the design and selection strategies to populate a national web archive. No criteria alone provides the solution for selecting the contents to archive. Thus, combinations must be used. The costs and complexity of the preservation of contents increase with the variety of media types archived, but archiving contents of just three media types (HTML, GIF and JPEG) reduced the coverage of a national web by only 5%. This is an interesting selection criterion to simplify web archival, in exchange for a small reduction on the coverage of the web. Designing a web user interface for a web archive is also a challenging task. There are alternative access methods but none of them is complete by itself, so they must be used in conjunction to provide access to the archived data.

Chapter 7

Conclusions

The web is a powerful source of information, but additional tools to help users in taking advantage from its potential are required. One of the problems that these tools must address is how to cope with a data source which was not designed to be automatically interpreted by software applications. Web warehousing is an approach to tackle this problem. It consists on extracting data from the web, storing it locally and then, providing uniform access methods that facilitate its automatic processing and reuse by different applications. This approach is conceptually similar to Data Warehousing approaches, used to integrate information from relational databases. However, the peculiar characteristics of the web, such as its dynamics and heterogeneity, raise new problems that must addressed to design an efficient Web Warehouse (WWh).

In general, the characteristics of the data sources have a major influence on the design of the information systems that process the data. A major challenge in the design of Web Warehouses is that web data models are scarce and become quickly stale. Previous web characterization studies showed that the web is composed of distinct portions with peculiar characteristics. It is important to accurately define the boundaries of these portions and model them, so that the design of a WWh can reflect the characteristics of the data it will store. The methodology used to sample the web influences the derived characterizations. Hence, the samples used to model a portion of the web must be gathered using a methodology that emulates the extraction stage of the web data integration process.

Previous works focused on architectural aspects of crawlers and warehouses. There are high-performance crawlers that enable the quick refreshment of the data kept in a WWh. Web Warehousing projects produced complete systems able to harvest, store and provide access to large amounts of web data. However, most research in these domains assumed that the web is uniform. This assumption is not supported by the results obtained in web characterization studies.

This thesis addressed the problem of web modelling and studied the influence of web characteristics in Web Warehouses design. The research was conducted following mainly an experimental methodology. A model for the Portuguese Web was derived and considering it, a WWh prototype named Webhouse was developed. This thesis contributed to answer the following research questions:

Which features should be considered in a model of the web? The characterization of the sites, contents and link structure of a web portion is crucial to design an efficient Web Warehouse to store it. Some features derived from the analysis of the global web may not be representative of more restricted domains, such as national webs. However, these portions can be of interest to large communities and characterizing a small portion of the web is quite accessible and can be done with great accuracy. Some web characteristics, such as web data persistence, require periodical samples of the web to be modelled. These metrics should be included in web models because they enable the identification of evolution tendencies that are determinant to the design of efficient Web Warehouses, which keep incrementally built data collections.

How can the boundaries of a portion of the web be defined? A set of selection criteria delimits the boundaries of a web portion and it is defined according to the requirements of the application that will process the harvested data. The selection criteria should be easily implemented as an automatic harvesting policy. Selection criteria based on content classification and domain restrictions revealed to be suitable options.

What can bias a web model? The methodology used to gather web samples influences the obtained characterizations. In the context of Web Warehousing, crawling is an adequate sampling method because it is the most

commonly used in web data extraction. However, the configuration and technology used in the crawler, and the existence of hazardous situations on the web influences the derived models. Thus, the interpretation of statistics gathered from a web portion, such as a national web, is beyond a mathematical analysis. It requires knowledge about web technology and social reality of a national community.

How persistent is information on the web? Web data persistence cannot be modelled through the analysis of a single snapshot of the web. Hence, models for the persistence of URLs and contents of the Portuguese Web were derived from several snapshots gathered for three years. The lifetime of URLs and contents follows a exponential distribution. Most URLs have short lives and the death rate is higher in the first months, but there is a minority that persists for long periods of time.

The obtained half-life of URLs was two months and the main causes of death were the replacement of URLs and the deactivation of sites. Persistent URLs are mostly static, short and tend to be linked from other sites. The lifetime of sites (half-life of 556 days) is significantly larger than the lifetime of URLs. The obtained half-life for contents was just two days. The comparison of the obtained results with previous works suggests that the lifetime of contents is decreasing. Persistent contents were not related to depth and were not particularly distributed among sites. About half of the persistent URLs referenced the same content during their lifetime.

How do web characteristics affect Web Warehouses design? Web data models help on important design decisions in the initial phases of Web Warehousing projects. Duplication of contents is prevalent on the web and it is difficult to avoid the download of duplicates during a crawl because the duplicates are commonly referenced by distinct and apparently unrelated URLs. A collection of contents built incrementally presents an additional number of duplicates because many contents remain unchanged over time and are repeatedly stored. Hence, eliminating duplicates at storage level in a WWh is an appealing feature. However, the mechanisms adopted for the

elimination of duplicates must address URL transience that may prevent the implementation of algorithms based on historical analysis.

Web data persistence influences the definition of data structures and algorithms to manage them in a WWh. Models that help predicting web data persistence enable measuring the freshness of the information kept and scheduling refreshment operations. A WWh that keeps information for long periods of time must also address preservation issues to maintain the stored information accessible after it is no longer available on-line.

The extraction of information from the web is a very sensitive task, because the software component responsible for it must address unpredicted situations. Web models enable the identification of hazardous situations on the web and contribute to the design of robust crawlers. A WWh must present a distributed architecture to face the large size of the web. Web characteristics influence the partitioning strategies adopted to perform load balancing among the processes that compose a WWh.

This thesis mainly addressed the extraction and loading of web data into a WWh. Several versions of Webhouse were iteratively developed until its design could not be significantly improved (Engineering validation approach). Webhouse proved to present a flexible and efficient architecture that enabled its usage in broader scopes than Web Warehousing. The problem of designing efficient Web Warehouses is complex and required combining knowledge from different fields. This research provided contributions in Web Characterization, Crawling, Web Warehousing and it was also an useful tool that contributed to support other research studies.

7.1 Limitations

This thesis presented a thorough study of the characteristics of the Portuguese Web. However, it was not clear if the same results would be achieved in other webs. Perhaps the characteristics of the sites and their content depends more on the methodology used to sample the web than on the community they belong to. Although the characterization of the Portuguese Web may not be representative

of other web portions, it was representative of the data that populated Webhouse. The web model could be extended with metrics, such as, the importance of sites and content based on popularity algorithms (Page *et al.*, 1999), the characteristics of contents belonging to specific categories (newspapers, web portals, digital portals, digital libraries) or the presence of partial duplicates. The experimental data sets used in the experiments were mainly composed by textual contents. It would be interesting to study data sets containing a wider range of media types.

The amount of information available on the web is extremely large. The Webhouse prototype proved to be efficient when processing relatively small amounts of data. However, designing a WWh able to harness petabytes of data would raise new problems that could not be studied with the resources available for this research. The Versus repository was designed to be used in a broader scope than just Web Warehousing. Although it presented satisfactory results in the performed experiments while processing web data, I believe that a large-scale web repository would require an architecture assuming that it would store web contents exclusively. This assumption would enable deeper system optimizations to support large-scale Web Warehouses.

This thesis assumed a pull-model to perform web data extraction, in which a crawler actively harvests information and loads it into a repository. However, push-models, such as RSS feeds (RDF Site Summaries) (W3C, 2003), are gaining popularity and they should be addressed in the architecture of a WWh. The storage throughput of the harvested contents and the meta-data extraction from contents in proprietary formats are the main problems of the current version of Webhouse. These aspects could be improved by studying alternative storage techniques and conversion tools.

Currently, systems like Hadoop (The Apache Software Foundation, 2006) provide useful experimental platforms to validate research in Web Warehousing. However, when this research began, the offer of Web Warehousing software was very limited and I had to develop Webhouse to support it. Webhouse was implemented using off-the-shelf database management systems to save on development time. It was shown that these systems can be used to support Web Warehouses. However, they must be adequately configured and tuned, requiring deep knowledge about the functioning of the DBMS. In the end, a question remains: is it

easier to develop specific solutions from scratch to implement a Web Warehouse or to configure and optimize a DBMS to support the required functionalities. I believe that it depends on the background knowledge of the developers.

7.2 Future Work

The amount and diversity of data available on the web tends to increase. Thus, research related to the creation of tools able to harness this information should remain very active in the next years.

This thesis addressed the identification and modelling of a national community web. As the web continues to grow, I believe that research on community webs will gain more importance, because most users access a limited set of information sources related to the communities they belong. More accurate national community webs could be defined by combining crawling policies with geographical tools. It would also be interesting to crawl samples of community webs and the general web simultaneously using the same crawler configuration to determine the differences among them.

In future work, it would be interesting to study the influence of popularity in web data persistence and combine multiple features that influence persistence in a weighted model, sensitive to the peculiar characteristics of web collections. I suspect that images have higher persistency than textual contents. So, another important direction would be to study the persistence of a broader scope of media types.

Hazardous situations to the automatic processing of web data were documented in this thesis. However, some of them could not be completely solved and new ones keep appearing. The identification of new hazardous situations on the web and the mitigation of their effects in web data processing systems is a required research activity.

A main problem in the design of Web Warehouses is that most web data is not generated having automatic interpretation in mind. This research tried to solve this problem by mitigating the perverse effects that low-quality data has on Web Warehouses. On the other hand, one could try to investigate how to increase the quality of the contents published on the web to facilitate its automatic processing.

The main objective of the Semantic Web is to create this machine-interpretable web (W3C, 2004), but its implementation has been too slow in practice. The success of the web was due to its simplicity. This enabled the widespread of web publications. I believe that the proposed models to support the Semantic Web are too complex to be a replacement of the existent web. One alternative way to increase the contents quality could be the creation of tools that help publishers in the identification and solution of problems with their contents.

Search engines and web archives are deeply related to web warehousing. This research raised my interest on the ranking mechanisms used by search engines. The PageRank algorithm used by Google is commonly identified as the key to the success of this search engine (Page *et al.*, 1999). However, it was presented in 1998 and I believe that the current mechanisms used by Google do not match this initial proposal mainly because search engines must prevent against web spammers that try to manipulate these well-known ranking algorithms. The study and application of web models is essential to derive new ranking algorithms.

The information kept in the repositories of search engines is periodically refreshed to update the indexes and the old one is discarded. The storage requirements of these repositories are relatively small and there is no need for interaction among them, because search engines are maintained by independent and rival enterprises. The main objective of search engines is to provide a small set of relevant results to a user search, independently from the number of pages existent on the web that may contain that information. Thus, search engines do not need to perform exhaustive harvests from the web.

On their turn, web archives have more demanding requirements for Web Warehouses than search engines. Web archives need to perform exhaustive crawls of the web and have high storage requirements, because they need to preserve large web collections built incrementally. Web archiving is a daunting task and I believe that the cooperation among international institutions could make it easier. My vision of the future of web archiving is having several institutions responsible for the preservation of the data published in independent web portions. Merging all the portions would correspond to the global web space. This way, the coverage of web archival would increase and the effort required to archive the web

would be divided across national or international organizations avoiding duplications. One way to assign web archiving responsibilities could be to delegate them to the registrars that administrate the top-level domains. Each registrar would become responsible for archiving the contents published in the sites hosted under its top-level domain. However, this approach would probably cause the increase of the price of domain registrations and domain owners would have to be charged according to the amount of contents they publish. Nonetheless, the web is global and users could be interested in finding historical contents world-wide. Although the gathering, storage and preservation could be spread across several web archives, users should be able to easily access information in all of them. One possible approach to achieve this could be having a standard and uniform access method supported by all web archives, independently from the underlying technology. Users could employ a tool that communicates with the web archive access mechanisms and merges the obtained results. This approach is similar to *meta-search*, which merges results gathered from several search engines (Chignell *et al.*, 1999). However, meta-search engines have not been very successful, mainly because the results returned by different search engines are quite similar and the relevance of the merged results is not significantly better than the one obtained through query on a single search engine. On its turn, web archives would return very different results because each one of the archives holds a different portion of the web. The searches could be refined by users to restrict results. For instance, by selecting the source web archives or content languages. The widespread usage of Web Warehouses to support global web archiving raises new and stimulating problems in several research areas:

- Research in P2P systems enabled content sharing over the Internet (Dabek *et al.*, 2001). Web archives should be collaborative to avoid redundant efforts. Thus, they would need to communicate with each others, most likely through the Internet. The communication in Wide-Area Networks is a complex task and research in protocols that enable efficient comunication among Web Warehouses would be interesting;

- Grid computing aims to create world-wide distributed and collaborative clusters of computers that enable mass processing mainly to support re-

search. For instance, to process data generated by satellites (Hoschek *et al.*, 2000). Web warehousing also requires the processing of large amounts of data and the application of Grid computing research in distributed Web Warehouses could be a possible direction for future work;

- Web search has been evolving in the past years. However, searching in historical web collections raises new problems that are far from being solved. After a few years of archiving, a web archive would easily hold many petabytes of data. Engines able to search among these data are required. Moreover, retrieving relevant information in historical collections is not trivial. The ranking algorithms used to find relevant documents within a single snapshot of the web cannot cope with web collections incrementally built (Berberich *et al.*, 2006). For instance, in general, the PageRank algorithm assumes that each content is identified by one URL and the contents that receive more links are the most important. In a web collection incrementally built, the same URL may be referencing many contents and determining the relative importance of contents gathered in different periods of time is not straightforward;

- Finding ways to preserve web data is not trivial. For instance, it is unclear which meta-data or tools must be kept to ensure the access to the archived contents in the future. The definition of adequate preservation strategies and standards that enable the replication of contents among several Web Warehouses is an open issue.

I think that the success of web archiving initiatives depends on the advances provided by Web Warehousing research. Web modelling will have a key role in this research field because the characteristics of web data influence the design of Web Warehouses, as it was shown in this thesis.

References

Abiteboul, S., Buneman, P. & Suciu, D. (2000). *Data on the web: from relations to semistructured data and XML*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Abiteboul, S., Cobena, G., Masanes, J. & Sedrati, G. (2002). A first experience in archiving the French web. In *Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, 1–15, Springer-Verlag, London, UK.

Albertsen, K. (2003). The paradigma web harvesting environment. In *Proceedings of 3rd ECDL Workshop on Web Archives*, Trondheim, Norway.

Almeida, R.B. & Almeida, V.A.F. (2004). A community-aware search engine. In *Proceedings of the 13th International Conference on World Wide Web*, 413–421, ACM Press.

Arlitt, M.F. & Williamson, C.L. (1997). Internet web servers: workload characterization and performance implications. *IEEE/ACM Transactions Networking*, **5**, 631–645.

Baeza-Yates, R. & Castillo, C. (2000). Caracterizando la Web chilena. In *Encuentro chileno de ciencias de la computación*, Sociedad Chilena de Ciencias de la Computación, Punta Arenas, Chile.

Baeza-Yates, R. & Castillo, C. (2004). Crawling the infinite web: five levels are enough. In *Proceedings of the 3rd Workshop on Web Graphs (WAW)*, Springer LNCS, Rome, Italy.

Baeza-Yates, R. & Castillo, C. (2005). Caracteristicas de la web chilena 2004. Technical report, Center for Web Research, University of Chile.

Baeza-Yates, R. & Poblete, B. (2006). Dynamics of the chilean web structure. *Comput. Networks*, **50**, 1464–1473.

Baeza-Yates, R., Poblete, J. & Saint-Jean, F. (2003). Evolución de la web chilena 2001-2002. http://www.todocl.cl/stats.phtml.

Baeza-Yates, R., Castillo, C. & López, V. (2005). Characteristics of the web of Spain. *Cybermetrics - International Journal of Scientometrics, Informetrics and Bibliometrics*, **9**.

Baeza-Yates, R., Castillo, C. & Efthimiadis, E. (2007a). Characterization of national web domains. *ACM Transactions on Internet Technology*, **7**.

Baeza-Yates, R., Castillo, C. & Graells, E. (2007b). Características de la web chilena 2006. Technical report, Center for Web Research, University of Chile.

Baldi, P., Frasconi, P. & Smyth, P. (2003). *Modeling the Internet and the web: probabilistic methods and algorithms*. Wiley.

Barr, D. (1996). *Common DNS Operational and Configuration Errors*.

Barroso, L.A., Dean, J. & Hölzle, U. (2003). Web search for a planet: The Google cluster architecture. *IEEE Micro*, **23**, 22–28.

Beitzel, S.M., Jensen, E.C., Chowdhury, A., Grossman, D. & Frieder, O. (2004). Hourly analysis of a very large topically categorized web query log. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 321–328, ACM Press, New York, NY, USA.

Bent, L., Rabinovich, M., Voelker, G.M. & Xiao, Z. (2004). Characterization of a large web site population with implications for content delivery. In *Proceedings of the 13th International Conference on World Wide Web*, 522–533, ACM Press.

Berberich, K., Bedathur, S. & Weikum, G. (2006). Rank synopses for efficient time travel on the web graph. In *Proceedings of the 15th Conference on Information and Knowledge Management*, Arlington, USA.

Berk, E. & Ananian, C.S. (2005). JLex: a lexical analyzer generator for Java(TM). http://www.cs.princeton.edu/~appel/modern/java/JLex/.

Berliner, B. (1990). CVS II: Parallelizing software development. In *Proceedings of the USENIX Winter 1990 Technical Conference*, 341–352, USENIX Association, Berkeley, CA.

Berners-Lee, T., Fielding, R. & Masinter, L. (2005). *Uniform Resource Identifier (URI): Generic Syntax*.

Bharat, K. & Broder, A. (1999). Mirror, mirror on the web: a study of host pairs with replicated content. In *Proceedings of the Eighth International Conference on World Wide Web*, 1579–1590, Elsevier North-Holland, Inc.

Bharat, K., Broder, A.Z., Dean, J. & Henzinger, M.R. (2000). A comparison of techniques to find mirrored hosts on the WWW. *Journal of the American Society of Information Science*, **51**, 1114–1122.

Bharat, K., Chang, B.W., Henzinger, M.R. & Ruhl, M. (2001). Who links to whom: Mining linkage between web sites. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, 51–58, IEEE Computer Society.

Bhowmick, S., Madria, S. & Ng, W.K. (2003). *Web Data Management*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Boldi, P. & Vigna, S. (2004). The webgraph framework I: compression techniques. In *Proceedings of the 13th International Conference on World Wide Web*, 595–602, ACM Press, New York, NY, USA.

Boldi, P., Codenotti, B., Santini, M. & Vigna, S. (2002a). Structural properties of the African web. In *Proceedings of the 11th International World Wide Web Conference*, Honolulu, Hawaii.

Boldi, P., Codenotti, B., Santini, M. & Vigna, S. (2002b). Ubicrawler: A scalable fully distributed web crawler. In *Proceedings of the 8th Australian World Wide Web Conference*.

Borbinha, J., Freire, N., Silva, M.J. & Martins, B. (2003). Internet search engines and opacs: Getting the best of two worlds. In *ElPub 2003 - ICCC/IFIP 7th International Conference on Electronic Publishing*, Guimarães, Portugal.

Brandman, O., Cho, J., Garcia-Molina, H. & Shivakumar, N. (2000). Crawler-friendly web servers. In *Proceedings of the Workshop on Performance and Architecture of Web Servers (PAWS)*, ACM Press, Santa Clara, California.

Brewington, B.E. & Cybenko, G. (2000). How dynamic is the web? *Computer Networks*, **33**, 257–276.

Brin, S. & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, **30**, 107–117.

Brin, S., Davis, J. & Garcia-Molina, H. (1995). Copy detection mechanisms for digital documents. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, 398–409, ACM Press, New York, NY, USA.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A. & Wiener, J. (2000). Graph structure in the web. In *Proceedings of the 9th International World Wide Web Conference on Computer networks*, 309–320, North-Holland Publishing Co.

Broder, A.Z., Glassman, S.C., Manasse, M.S. & Zweig, G. (1997). Syntactic clustering of the web. In *Proceedings of the Sixth International conference on World Wide Web*, 1157–1166, Elsevier Science Publishers Ltd.

Broder, A.Z., Najork, M. & Wiener, J.L. (2003). Efficient URL caching for World Wide Web crawling. In *Proceedings of the 12th International Conference on World Wide Web*, 679–689, ACM Press.

Burkard, T. (2002). *Herodotus: A Peer-to-Peer Web Archival System*. Master thesis, Massachusetts Institute of Technology.

Caceres, R., Douglis, F., Feldmann, A., Glass, G. & Rabinovich, M. (1998). Web proxy caching: the devil is in the details. *SIGMETRICS Performance Evaluation Review*, **26**, 11–15.

Cafarella, M. & Cutting, D. (2004). Building nutch: Open source search. *Queue*, **2**, 54–61.

Callaghan, B., Pawlowski, B. & Staubach, P. (1995). *RFC 1813: NFS Version 3 Protocol Specification*. Sun Microsystems, Inc.

Campos, J. (2003). *Versus: a Web Repository*. Master thesis, Faculdade de ciências, Universidade de Lisboa.

Cardoso, N., Silva, M.J. & Costa, M. (2004). The XLDB Group at CLEF'2004. In *Cross Language Evaluation Forum - Working Notes for the CLEF 2004 Workshop*, Bath, UK.

Cardoso, N., Andrade, L., Simões, A. & Silva, M.J. (2005a). The XLDB group participation at the CLEF'2005 ad hoc task. In *Cross Language Evaluation Forum: Working Notes for the CLEF 2005 Workshop*, Wien, Austria.

Cardoso, N., Martins, B., Chaves, M., Andrade, L. & Silva, M.J. (2005b). The XLDB Group at GeoCLEF 2005. In *Working Notes for the CLEF 2005 Workshop*, Wien, Austria.

Cardoso, N., Martins, B., Gomes, D. & Silva, M.J. (2005c). *WPT 03: Recolha da Web Portuguesa*. Diana Santos.

Cardoso, N., Silva, M.J. & Martins, B. (2006). The University of Lisbon at 2006 Ad-Hoc Task. In *Cross Language Evaluation Forum: Working Notes for the CLEF 2006 Workshop*, Alicante, Spain.

Castillo, C. (2004). *Effective Web Crawling*. Ph.D. thesis, University of Chile.

Chakrabarti, S., van den Berg, M. & Dom, B. (1999). Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, **31**, 1623–1640.

Chakrabarti, S., Joshi, M.M., Punera, K. & Pennock, D.M. (2002). The structure of broad topics on the web. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, 251–262, ACM Press, New York, NY, USA.

Chaves, M.S., Silva, M.J. & Martins, B. (2005). A Geographic Knowledge Base for Semantic Web Applications. In *Proc. of the 20th Brazilian Symposium on Databases, Uberlândia, Minas Gerais, Brazil*, 40–54.

Chignell, M.H., Gwizdka, J. & Bodner, R.C. (1999). Discriminating meta-search: A framework for evaluation. *Information Processing and Management*, **35**, 337–362.

Cho, J. (2001). *Crawling the Web: Discovery and maintenance of large-scale web data*. Ph.D. thesis, Stanford University.

Cho, J. & Garcia-Molina, H. (2000a). The evolution of the web and implications for an incremental crawler. In *Proceedings of 26th International Conference on Very Large Data Bases*, 200–209.

Cho, J. & Garcia-Molina, H. (2000b). Synchronizing a database to improve freshness. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 117–128, ACM Press.

Cho, J. & Garcia-Molina, H. (2002). Parallel crawlers. In *Proceedings of the 11th International Conference on World Wide Web*, 124–135, ACM Press.

Cho, J. & Garcia-Molina, H. (2003). Estimating frequency of change. *ACM Transactions Internet Technology*, **3**, 256–290.

Cho, J. & Roy, S. (2004). Impact of search engines on page popularity. In *Proceedings of the 13th international conference on World Wide Web*, 20–29, ACM Press.

Cho, J., Garcia-Molina, H. & Page, L. (1998). Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, **30**, 161–172.

Cho, J., Garcia-Molina, H., Haveliwala, T., Lam, W., Paepcke, A., Raghavan, S. & Wesley, G. (2004). Stanford WebBase components and applications. Technical report, Stanford Database Group.

Christensen, N. (2005). Preserving the bits of the danish internet. In *5th International Web Archiving Workshop (IWAW05)*, Viena, Austria.

Cockburn, A. & McKenzie, B. (2001). What do web users do? An empirical analysis of web use. *International Journal Human-Computuer Studies*, **54**, 903–922.

Connolly, D. & Masinter, L. (2000). *The 'text/html' Media Type*.

Consortium, U.W.A. (2006). UK web archiving consortium: Project overview. http://info.webarchive.org.uk/.

Cooper, B.F., Crespo, A. & Garcia-Molina, H. (2002). The Stanford archival repository project: preserving our digital past. Technical report 2002-47, Department of Computer Science, Stanford University.

Costa, L. & Sarmento, L. (2006). Component evaluation in a question answering system. In N. Calzolari, K. Choukri, A. Gangemi, B. Maegaard, J. Mariani, J. Odjik & D. Tapias, eds., *Proceedings of the 5th International Conference on Language Resources and Evaluation*, 1520–1523.

Costa, M. (2004). *SIDRA: a Flexible Web Search System*. Master's thesis, Department of Informatics, University of Lisbon, DI/FCUL TR-04-17.

Cothey, V. (2004). Web-crawling reliability. *J. Am. Soc. Inf. Sci. Technol.*, **55**, 1228–1238.

Crespo, A. & Garcia-Molina, H. (1998). Archival storage for digital libraries. In *Proceedings of the Third ACM International Conference on Digital Libraries*.

Cunha, C., Bestavros, A. & Crovella, M. (1995). Characteristics of WWW client-based traces. Tech. rep., Boston, MA, USA.

da Costa Carvalho, A.L., de Souza Bezerra, A.J., de Moura, E.S., da Silva, A.S. & Peres, P.S. (2005). Detecção de réplicas utilizando conteúdo e estrutura. In *Simpósio Brasileiro de Banco de Dados*, 25–39, ACM Press, Uberlândia, Brasil.

Dabek, F., Brunskill, E., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I. & Balakrishnan, H. (2001). Building peer-to-peer systems with Chord, a distributed lookup service. 81–86.

Daigle, L., van Gulik, D., Iannella, R. & Faltstrom, P. (2002). *Uniform Resource Names (URN) Namespace Definition Mechanisms*.

Davis, C., Vixie, P., Goodwin, T. & Dickinson, I. (1996). *A Means for Expressing Location Information in the Domain Name System*.

Davison, B.D. (1999). Web traffic logs: An imperfect resource for evaluation. In *Proceedings of th INET'99 Conference*.

Day, M. (2003). Collecting and preserving the World Wide Web. http://www.jisc.ac.uk/uploaded_documents/archiving_feasibility.pdf.

Dean, J. & Ghemawat, S. (2004). In *MapReduce: simplified data processing on large clusters*, San Francisco, California.

DELOS (2006). Welcome to cross language evaluation forum. http://www.clef-campaign.org/.

Denehy, T. & Hsu, W. (2003). Duplicate management for reference data. Technical report RJ 10305, IBM Research.

Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., McCurley, K.S., Rajagopalan, S., Tomkins, A., Tomlin, J.A. & Zien, J.Y. (2004). A case for automated large scale semantic annotation. *Journal of Web Semantics*, **1**, 1–17.

Douglis, F., Feldmann, A., Krishnamurthy, B. & Mogul, J.C. (1997). Rate of change and other metrics: a live study of the World Wide Web. In *USENIX Symposium on Internet Technologies and Systems*.

Drugeon, T. (2005). A technical approach for the French web legal deposit. In *5th International Web Archiving Workshop (IWAW05)*, Viena, Austria.

Edwards, J., McCurley, K. & Tomlin, J. (2001). An adaptive model for optimizing performance of an incremental web crawler. In *Proceedings of the 10th international conference on World Wide Web*, 106–113, ACM Press, New York, NY, USA.

Eiron, N., McCurley, K.S. & Tomlin, J.A. (2004). Ranking the web frontier. In *Proceedings of the 13th international conference on World Wide Web*, 309–318, ACM Press.

Entlich, R. (2004). Blog today, gone tomorrow? Preservation of Weblogs. *RLG Diginews*, **8**.

Exposto, J., Macedo, J., Pina, A., Alves, A. & Rufino, J. (2005). Geographical partition for distributed web crawling. In *Proceedings of the 2005 Workshop on Geographic Information Retrieval*, 55–60, ACM Press, New York, NY, USA.

Fetterly, D., Manasse, M., Najork, M. & Wiener, J. (2003). A large-scale study of the evolution of web pages. In *Proceedings of the 12th International Conference on World Wide Web*, 669–678, ACM Press, New York, NY, USA.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. & Berners-Lee, T. (1999). *Hypertext Transfer Protocol – HTTP/1.1*.

Finkel, R.A., Zaslavsky, A., Monostori, K. & Schmidt, H. (2002). Signature extraction for overlap detection in documents. In *Twenty-Fifth Australasian Computer Science Conference (ACSC2002)*, ACS, Melbourne, Australia.

Flake, G., Lawrence, S. & Giles, C.L. (2000). Efficient identification of web communities. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 150–160, Boston, MA.

Freed, N. & Borenstein, N. (1996a). *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*.

Freed, N. & Borenstein, N. (1996b). *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*.

Freitas, S., Afonso, A.P. & Silva, M.J. (2006). Mobile Geotumba: Geographic information retrieval system for mobile devices. In *Proceedings of the 4th MiNEMA Workshop*, 83–87.

Funredes (2001). The place of Latin languages on the Internet. http://www.funredes.org/LC/english/L5/L5index_english.html.

Ghemawat, S., Gobioff, H. & Leung, S.T. (2003). The Google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, ACM, Bolton Landing, NY USA.

Gibson, D., Kleinberg, J.M. & Raghavan, P. (1998). Inferring web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, 225–234, Pittsburgh, Pennsylvania.

Gomes, D. & Silva, M.J. (2005). Characterizing a national community web. *ACM Transactions on Internet Technology*, **5**, 508–531.

Gomes, D. & Silva, M.J. (2006a). Modelling information persistence on the web. In *ICWE '06: Proceedings of the 6th international conference on Web engineering*, 193–200, ACM Press, New York, NY, USA.

Gomes, D. & Silva, M.J. (2006b). The Viuva Negra web crawler. DI/FCUL TR 06–21, Department of Informatics, University of Lisbon.

Gomes, D., Freitas, S. & Silva, M.J. (2006a). Design and selection criteria for a national web archive. In J. Gonzalo, C. Thanos, M.F. Verdejo & R.C.

Carrasco, eds., *Proc. 10th European Conference on Research and Advanced Technology for Digital Libraries, ECDL*, vol. 4172, Springer-Verlag.

Gomes, D., Santos, A.L. & Silva, M.J. (2006b). Managing duplicates in a web archive. In L.M. Liebrock, ed., *Proceedings of the 21th Annual ACM Symposium on Applied Computing (ACM-SAC-06)*, Dijon, France.

Google (2003). Google web search features. www.google.com/help/features.html#link.

Grefenstette, G. & Nioche, J. (2000). Estimation of English and non-English language use on the WWW. In *Proceedings of RIAO'2000, Content-Based Multimedia Information Access*, 237–246, Paris.

Gribble, S.D. & Brewer, E.A. (1997). System design issues for Internet middleware services: Deductions from a large client trace. In *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97)*, Monterey, CA.

Gruhl, D., Chavet, L., Gibson, D., Meyer, J., Pattanayak, P., Tomkins, A. & Zien, J. (2004). How to build a Webfountain: An architecture for very large-scale text analytics. *IBM Systems Journal*, **43**, 64–77.

Habib, M.A. & Abrams, M. (2000). Analysis of sources of latency in downloading web pages. In *WebNet*, 227–232, San Antonio, Texas, USA.

Hakala, J. (2001). Collecting and preserving the web: Developing and testing the NEDLIB harvester. *RLG Diginews*, **5**.

Hallam-Baker, P.M. & Connolly, D. (2005). Session identification URI. http://www.w3.org/TR/WD-session-id.html.

Hallgrímsson, T. & Bang, S. (2003). Nordic web archive. In *Proceedings of 3rd ECDL Workshop on Web Archives*, Trondheim, Norway.

Handschuh, S., Staab, S. & Volz, R. (2003). On deep annotation. In *Proceedings of the 12th International Conference on World Wide Web*, 431–438, ACM Press, New York, NY, USA.

Harrenstien, K., Stahl, M.K. & Feinler, E.J. (1985). *NICNAME/WHOIS*.

Henzinger, M. (2003). Algorithmic challenges in web search engines. *Journal of Internet Mathematics*, **1**, 115–126.

Henzinger, M.R., Heydon, A., Mitzenmacher, M. & Najork, M. (2000). On near-uniform url sampling. In *Proceedings of the 9th International World Wide Web Conference on Computer networks: the international journal of computer and telecommunications networking*, 295–308, North-Holland Publishing Co.

Henzinger, M.R., Motwani, R. & Silverstein, C. (2002). Challenges in web search engines. *SIGIR Forum*, **36**, 11–22.

Hexa Software Development Center (2003). Geo targeting IP address to country city region ISP latitude longitude database for Internet developers - ip2location. http://www.ip2location.com/.

Heydon, A. & Najork, M. (1999). Mercator: A scalable, extensible web crawler. *World Wide Web*, **2**, 219–229.

Hirai, J., Raghavan, S., Garcia-Molina, H. & Paepcke, A. (1999). WebBase: A repository of web pages. In *Proceedings of the 9th World-Wide Web Conference*.

Hoschek, W., Janez, F.J., Samar, A., Stockinger, H. & Stockinger, K. (2000). Data management in an international data grid project. In *GRID*, 77–90.

HyperSonicSQL (2001). HypersonicSQL. http://sourceforge.net/projects/hsqldb/.

ICANN (2004). ICANN | Verisign's wildcard service deployment. http://www.icann.org/topics/wildcard-history.html.

Iyengar, A.K., Squillante, M.S. & Zhang, L. (1999). Analysis and characterization of large-scale web server access patterns and performance. *World Wide Web*, **2**, 85–100.

Jahn (2004). Spider traps - an upcoming arms race. http://www.jahns-home.de/rentmei/html/sptraps.html.

Jaimes, A., del Solar, J.R., Verschae, R., Yaksic, D., Baeza-Yates, R., Davis, E. & Castillo, C. (2003). On the image content of the Chilean web. In *LA-WEB '03: Proceedings of the First Conference on Latin American Web Congress*, 72, IEEE Computer Society.

Jul, J. (2005). Calimaco, um repositório de documentos biológicos. Tecnhical report, Department of Informatics, University of Lisbon, Lisbon, Portugal, in Portuguese.

Kahle, B. (2002). The Internet Archive. *RLG Diginews*, **6**.

Katz, R.H. (1990). Toward a unified framework for version modeling in engineering databases. *ACM Computing Surveys*, **22**, 375–408.

Kelly, T. & Mogul, J. (2002). Aliasing on the World Wide Web: Prevalence and performance implications. In *Proceedings of the 11th International World Wide Web Conference*, Honolulu, Hawaii.

Kleinberg, J.M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, **46**, 604–632.

Koehler, W. (2002). Web page change and persistence–a four-year longitudinal study. *Journal of the American Society for Information Science and Technology*, **53**, 162–171.

Koht-Arsa, K. (2003). *High Performance Cluster-based Web Spiders*. Master's thesis, Graduate School, Kasetsart University.

Koster, M. (1994). A standard for robot exclusion. http://www.robotstxt.org/wc/norobots.html.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A. & Upfal, E. (2000). The Web as a graph. In *Proc. 19th ACM SIGACT-SIGMOD-AIGART Symp. Principles of Database Systems, PODS*, 1–10, ACM Press.

Kumar, R., Novak, J., Raghavan, P. & Tomkins, A. (2005). On the bursty evolution of blogspace. *World Wide Web*, **8**, 159–178.

Kunze, J., Arvidson, A., Mohr, G. & Stack, M. (2006). *The WARC File Format (Version 0.8 rev B)*. Internet Draft.

Lavoie, B., O'Neill, E.T. & McClain, P. (1997). OCLC office of research examines web-accessible information to find order in chaos.

Lawrence, S. & Giles, C.L. (1999). Accessibility of information on the web. *Nature*, **400**, 107–109.

Lawrence, S., Coetzee, F., Glover, E., Flake, G., Pennock, D., Krovetz, B., Nielsen, F., Kruger, A. & Giles, L. (2000). Persistence of information on the web: analyzing citations contained in research articles. In *Proceedings of the 9th International Conference on Information and Knowledge Management*, 235–242, ACM Press, New York, NY, USA.

Leung, S.T.A., Perl, S.E., Stata, R. & Wiener, J.L. (2001). Towards web-scale web archeology. Research Report 174, Compaq Research Center, Paolo Alto CA.

Lifantsev, M. (2000). Voting model for ranking web pages. In P. Graham & M. Maheswaran, eds., *International Conference on Internet Computing*, 143–148, Las Vegas, Nevada.

Lifantsev, M. & Chiueh, T.C. (2003). Implementation of a modern web search engine cluster. In *FREENIX Track: 2003 USENIX Annual Technical Conference*, San Antonio, Texas, USA.

Liu, B. (1998). *Characterizing Web Response Time*. Master's thesis, Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

MacDonald, J. (1999). Versioned file archiving, compression, and distribution. http://www.cs.berkeley.edu/~jmacd/.

Marktest (2003). Netpanel. http://netpanel.marktest.pt/.

Markwell, J. & Brooks, D.W. (2003). 'Link rot' limits the usefulness of web-based educational materials in biochemistry and molecular biology. *Biochemistry and Molecular Biology Education*, **31**, 69–72.

Marshak, M. & Levy, H. (2003). Evaluating web user perceived latency using server side measurements. *Computer Communications*, **26**, 872–887.

Martins, B. (2004). *Inter-Document Similarity in Web Searches*. Master's thesis, Department of Informatics, University of Lisbon.

Martins, B. & Silva, M.J. (2003). Web information retrieval with result set clustering. In *Progress in Artificial Intelligence*, 450–454, Springer Berlin/Heidelberg, Beja, Portugal.

Martins, B. & Silva, M.J. (2004a). Spelling correction for search engine queries. In *Proceedings of EsTAL - España for Natural Language Processing*, 372–383.

Martins, B. & Silva, M.J. (2004b). A statistical study of the WPT 03 corpus. In *Proceedings of EsTAL - España for Natural Language Processing*, Alicante, Spain.

Martins, B. & Silva, M.J. (2005a). Language identification in web pages. In *Proceedings of the 20th Annual ACM Symposium on Applied Computing (ACM-SAC-05)*, ACM Press, Santa Fe, New Mexico.

Martins, B. & Silva, M.J. (2005b). The WebCAT framework : Automatic generation of meta-data for web resources. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*.

Martins, B., Cardoso, N., Chaves, M., Andrade, L. & Silva, M.J. (2006). The University of Lisbon at GeoCLEF 2006. In *Cross Language Evaluation Forum: Working Notes for the CLEF 2006 Workshop*, Alicante, Espanha.

MAXMIND LLC (2003). Maxmind: How to locate your internet visitors geotargeting IP address to country state city ISP organization latitude longitude. http://www.maxmind.com/.

MCCURLEY, K.S. & TOMKINS, A. (2004). Mining and knowledge discovery from the web. In *2004 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04)*, 4, IEEE, Inc, Hong Kong, SAR, China.

MOCKAPETRIS, P.V. (1987). *Domain names - concepts and facilities*. United States.

MOGUL, J. (1999a). A trace-based analysis of duplicate suppression in HTTP. Technical Report 99/2, Compaq Computer Corporation Western Research Laboratory.

MOGUL, J. (1999b). Errors in timestamp-based HTTP header values. Research Report 99/3, Compaq Computer Corporation Western Research Laboratory.

MOHR, G., KIMPTON, M., STACK, M. & RANITOVIC, I. (2004). Introduction to heritrix, an archival quality web crawler. In *4th International Web Archiving Workshop (IWAW04)*, Bath, UK.

NAJORK, M. & HEYDON, A. (2001). On high-performance web crawling. SRC research report, Compaq Systems Research Center.

NAJORK, M. & WIENER, J.L. (2001). Breadth-first crawling yields high-quality pages. In *Proceedings of the 10th International World Wide Web Conference*, 114–118, Elsevier Science, Hong Kong.

NANAVATI, A., CHAKRABORTY, A., DEANGELIS, D., GODIL, H. & D'SILVA, T. (2004). An investigation of documents on the World Wide Web. Tech. rep.

NATIONAL LIBRARY OF AUSTRALIA (2006). Padi - web archiving. http://www.nla.gov.au/padi/topics/92.html, 18.

NETCRAFT LTD. (2004). Netcraft: April 2003 archives. http://news.netcraft.com/archives/2003/04/index.html.

Noronha, N., Campos, J.P., Gomes, D., Silva, M.J. & Borbinha, J. (2001). A deposit for digital collections. In P. Constantopoulos & I.T. Sølvberg, eds., *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries, ECDL*, vol. 2163 of *LNCS*, 200–212, Springer.

Ntoulas, A., Cho, J. & Olston, C. (2004). 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*, 1–12, ACM Press.

Ntoulas, A., Zerfos, P. & Cho, J. (2005). Downloading textual hidden web content through keyword queries. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital libraries*, 100–109, ACM Press, New York, NY, USA.

OCLC (2001). Impact of virtual hosting on the analysis. http://www.oclc.org/research/projects/archive/wcp/pubs/rn4-virtualhosting.htm.

Olsen, S. (2002). Does search engine's power threaten web's independence? *CNET News.com*.

O'Neill, E.T. (1999). Web sites: Concepts, issues, and definitions. http://wcp.oclc.org/pubs/rn1-websites.html.

O'Neill, E.T., Lavoie, B.F. & Bennett, R. (2003). How "world wide" is the web?: Trends in the evolution of the public web. *D-Lib Magazine*, **9**.

Oracle Corporation (2004). Oracle9i. http://www.oracle.com/ip/index.html?content.html.

Overture Services Inc. (2003). Alltheweb.com: Frequently asked questions - URL investigator. www.alltheweb.com/help/faqs/url_investigator.

Page, L., Brin, S., Motwani, R. & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Database Group.

Patterson, A. (2004). Why writing your own search engine is hard. *Queue*, **2**, 48–53.

Paulson, L.D. (2005). Building rich web applications with AJAX. *Computer*, **38**, 14–17.

Perseus Development Corp. (2004). Perseus blog survey.

Phillips, M. (2003). PANDORA, Australia's Web Archive, and the Digital Archiving System that Supports it. *DigiCULT.info*, 24.

Pike, R., Dorward, S., Griesemer, R. & Quinlan, S. (2005). Interpreting the data: Parallel analysis with sawzall. *Scientific Programming Journal - Special Issue on Grids and Worldwide Computing Programming Models and Infrastructure*, **13**, 227–298.

Pinkerton, B. (1994). Finding what people want: experiences with the webcrawler. In *2nd World Wide Web Conference*, Geneva, Switzerland.

Punpiti, S.S. (2000). Measuring and analysis of the Thai world wide web. In *Proceedings of the Asia Pacific Advance Network*, 225–230.

Qin, J., Zhou, Y. & Chau, M. (2004). Building domain-specific web collections for scientific digital libraries: a meta-search enhanced focused crawling method. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital libraries*, 135–141, ACM Press.

Rabin, M.O. (1979). Probabilistic algorithm in finite fields. Tech. Rep. MIT/LCS/TR-213.

Raghavan, S. & Garcia-Molina, H. (2001). Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Data Bases*, 129–138, Morgan Kaufmann Publishers Inc.

RARE (1992). RIPE ncc - network management database. http://www.ripn.net/nic/ripe-docs/ripe-078.ps.

Rauber, A., Aschenbrenner, A. & Witvoet, O. (2002). Austrian on-line archive processing: Analyzing archives of the World Wide Web.

Rhea, S., Eaton, P., Geels, D., Weatherspoon, H., Zhao, B. & Kubiatowicz, J. (2003). Pond: the Oceanstore Prototype. In *Proceedings of the 2nd Usenix Conference on File and Storage Technologies (FAST'03)*.

Richardson, M. & Domingos, P. (2002). The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. In *Advances in Neural Information Processing Systems 14*, MIT Press.

Rivest, R. (1992). *RFC 1321 - The MD5 Message-Digest Algorithm*.

Rodeh, O. & Teperman, A. (2003). zfs: A scalable distributed file system using object disks. In *Proceedings of the 20 th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies (MSS'03)*, 207, IEEE Computer Society.

Rosenstein, M. (2000). What is actually taking place on web sites: e-commerce lessons from web server logs. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, 38–43, ACM Press, New York, NY, USA.

Sanguanpong, S. & Koht-Arsa, K. (2003). Data partition and job scheduling for balancing load on cluster of web spiders. In *National Computer Science and Engineering Conference (NCSEC-2003)*, Chol Buri.

Santos, D. & Cardoso, N. (2005). Portuguese at CLEF 2005: Reflections and Challenges. In *Cross Language Evaluation Forum: Working Notes for the CLEF 2005 Workshop*, Wien, Austria.

Santos, D. & Chaves, M.S. (2006). The place of place in geographical IR. In *3rd Workshop on Geographic Information Retrieval, SIGIR'06*, 5–8, Seattle.

Sarmento, L. (2006). Baco - a large database of text and co-occurrences. In N. Calzolari, K. Choukri, A. Gangemi, B. Maegaard, J. Mariani, J. Odjik & D. Tapias, eds., *Proceedings of the 5th International Conference on Language Resources and Evaluation*, 1787–1790.

Sherfesee, D. & O'Driscoll, N. (2005). A web mining research platform. In *Proceedings of the 28th Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, 682–682, ACM Press, New York, NY, USA.

Shivakumar, N. & Garcia-Molina, H. (1995). SCAM: A copy detection mechanism for digital documents. In *Proceedings of the 2nd Annual Conference on the Theory and Practice of Digital Libraries*.

Shivakumar, N. & Garcia-Molina, H. (1999). Finding near-replicas of documents and servers on the web. In *Selected papers from the International Workshop on The World Wide Web and Databases*, 204–212, Springer-Verlag.

Shkapenyuk, V. & Suel, T. (2002). Design and implementation of a high-performance distributed web crawler. In *Proceedings of the 18th International Conference on Data Engineering*, 357, IEEE Computer Society.

Silva, A.S., Veloso, E.A., Golghe, P.B., Ribeiro-Neto, B., Laender, A.H.F. & Ziviani, N. (1999). Cobweb a crawler for the Brazilian web. In *Proceedings of the String Processing and Information Retrieval Symposium & International Workshop on Groupware*, 184, IEEE Computer Society.

Silva, L.O., Macedo, J., Costa, A., Belo, O. & Santos, A. (2002). Obtenção de estatísticas do www em Portugal. Tech. rep., OCT and DI, Universidade do Minho.

Silva, M.J. (2003). The case for a portuguese web search engine. In P. Isaias, ed., *Proceedings of IADIS International Conference WWW/Internet 2003*, Algarve, Portugal.

Silva, M.J., Martins, B., Chaves, M.S., Cardoso, N. & Afonso, A.P. (2006). Adding Geographic Scopes to Web Resources. *CEUS - Computers, Environment and Urban Systems, Elsevier Science*, **30**, 378–399.

Silverstein, C., Marais, H., Henzinger, M. & Moricz, M. (1999). Analysis of a very large web search engine query log. *SIGIR Forum*, **33**, 6–12.

Smith, Z. (1997). The truth about the web. *Web Techniques Magazine*, **2**.

Spinellis, D. (2003). The decay and failures of web references. *Communications of the ACM*, **46**, 71–77.

Sydow, M. (2004). Random surfer with back step. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers and Posters*, 352–353, ACM Press, New York, NY, USA.

The Apache Software Foundation (2004). *Apache HTTP Server Version 1.3: Module mod_include*.

The Apache Software Foundation (2006). About Hadoop. http://lucene.apache.org/hadoop/about.html.

The Library of Congress (2006). MINERVA home page (mapping the internet electronic resources virtual archive, library of congress web archiving). http://lcweb2.loc.gov/cocoon/minerva/html/minerva-home.html.

The Web Robots Pages (2005). HTML author's guide to the ROBOTS meta-tag. http://www.robotstxt.org/wc/meta-user.html.

Thelwall, M. (2002). A free database of university web links: data collection issues. *International Journal of Scientometrics, Informetrics and Bibliometrics*, **6/7**.

Thurow, S. (2002). *Search Engine Visibility*. New Riders Press, Indianapolis, USA.

W3C (1999). HTML 4.01 specification. http://www.w3.org/TR/html401/.

W3C (1999a). Web characterization activity statement. http://www.w3.org/WCA/Activity.html.

W3C (1999b). Web characterization terminology and definitions sheet. http://www.w3.org/1999/05/WCA-terms/.

W3C (2003). W3C RSS 1.0 news feed creation how-to. http://www.w3.org/2001/10/glance/doc/howto.

W3C (2004). W3C Semantic Web. http://www.w3.org/2001/sw/.

Webb, C. (2000). Towards a preserved national collection of selected Australian digital publications. In *Proceedings of Preservation 2000 Conference*, York, UK.

Wills, C.E. & Mikhailov, M. (1999). Examining the cacheability of user-requested web resources. In *Proceedings of the 4th International Web Caching Workshop*, San Diego, California.

Woodruff, A., Aoki, P.M., Brewer, E. & Gauthier, P. (1996). An investigation of documents from the World Wide Web. In *Proceedings of the 5th International World Wide Web Conference on Computer Networks and ISDN Systems*, 963–980, Elsevier Science Publishers B. V., Amsterdam, The Netherlands.

Wu, J. & Aberer, K. (2004). Using siterank for decentralized computation of web document ranking. In *Adaptive Hypermedia 2004*, Eindhoven University of Technology, The Netherlands.

Yan, H., Wang, J., Li, X. & Guo, L. (2002). Architectural design and evaluation of an efficient web-crawling system. *The Journal of Systems and Software*, **60**, 185–193.

You, L.L. & Karamanolis, C. (2004). Evaluation of efficient archival storage techniques. In *21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies*, College Park, MD.

Zelkowitz, M.V. & Wallace, D.R. (1998). Experimental models for validating technology. *Computer*, **31**, 23–31.

Zook, M. (2000). Internet metrics: using host and domain counts to map the Internet.

Index