FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Pesquisa e Extração de Informação de Grupos de Discussão na Web

Jorge Miguel Amado Moreira

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Maria Eduarda Silva Mendes Rodrigues (Professora Doutora)

Co-orientador: Luís Cláudio dos Santos Barradas (Engenheiro)

18 de Fevereiro de 2013

Pesquisa e Extração de Informação de Grupos de Discussão na Web

Jorge Miguel Amado Moreira

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Ricardo Santos Morla (Professor Doutor)

Arguente: Pedro Henriques Abreu (Professor Doutor)

Vogal: Maria Eduarda Silva Mendes Rodrigues (Professora Doutora)

---

18 de Fevereiro de 2013

Resumo

Nos dias de hoje tem-se verificado um aumento nos conteúdos oferecidos pela Web, disponibilizando grandes quantidades de informação, tornando-a uma área fértil para a investigação no que se refere à extração do conhecimento. Este facto leva ao aparecimento de várias técnicas eficazes de mineração de dados. Entretanto, as empresas dão cada vez mais importância à inovação, para poderem ser competitivas e sobreviverem no mercado em que se inserem. Por outro lado, a inovação tem uma forte relação com o conhecimento e a obtenção deste pode resultar de vários agentes económicos e sociais (utilizadores, clientes, colaboradores, parceiros de negócio, entre outros), que contribuem diariamente com informação muito relevante para as empresas através de blogues, fóruns, redes sociais, entre outros. Neste contexto, a extração de dados de comunidades online mostra-se um processo muito importante. Assim, põe-se o problema de encontrar uma solução que permita a pesquisa, a análise e a extração de informação relevante para suportar processos de inovação em empresas. Para dar resposta a este problema, foram investigadas abordagens de recolha de dados de grupos de discussão na web. Decidiu-se utilizar a metodologia CRISP-DM na execução do projeto e as tecnologias usadas foram: o *hadoop*, o *nutch* e o *solr*. Aplicaram-se técnicas que levaram ao desenvolvimento dum sistema capaz de extrair conteúdos existentes em fóruns de discussão, armazená-los numa plataforma de processamento de dados escalável e efetuar a exploração dos mesmos, apresentando os resultados da análise num formato normalizado que possa ser consumido por outras. Realizaram-se também testes à plataforma desenvolvida, tendo-se concluído que o sistema criado permite efetuar a extração dos dados de fóruns de discussão e a pesquisa dessa informação.

Abstract

Nowadays there has been an increase in the content offered by the Web, providing large amounts of information, making it a fertile area for the investigation with regard to the extraction of knowledge and leading to the appearance of several effective data mining techniques. However, companies give more importance to innovation, to become competitive and survive in the market in which they operate. On the other hand, innovation has a strong relationship with knowledge and this may proceed from economic and social agents (users, customers, employees, business partners, and so on), who daily contribute with very relevant information to companies through blogs, forums, social networks, among others. In this context, the data extraction from online communities becomes a very important process. Thus arises the problem of finding a solution that provides search, analysis and extraction of relevant information to support innovation processes in companies. To address this problem, approaches have been investigated for collecting data from online discussion foruns. It was decided to use the CRISP-DM methodology to implement the project and the technologies used were: hadoop, nutch and solr. Techniques were applied in order to extract the contents present in discussion forums. It was developed a system able to extract existing content in discussion foruns, store them in a scalable data processing platform and perform exploitation of them, presenting the analysis results in a standardized format that can be consumed by others. Tests have also been made to the platform developed, it was concluded that the system created allows to perform data extraction from discussion forums and search this information.

Agradecimentos

Aos meu orientadores, a professora Eduarda Mendes Rodrigues e o engenheiro Cláudio Barradas por me terem orientado ao longo de todo o percurso da dissertação e por estarem sempre disponíveis para discutir os problemas que fui tendo ao longo deste período.

Ao Arian Rodrigo Pasquali por me ter ajudado no projeto.

Jorge Miguel Amado Moreira

"Só fazemos melhor aquilo que repetidamente insistimos em melhorar. A busca da excelência não deve ser um objetivo e sim um hábito"

Aristóteles

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

API	Application Programming Interface
CRISP-DM	Cross-Industry Standard Process for Data Mining
CRUD	Create read update delete
CSV	Comma-separated values
FIFO	First-In-First-Out
HDFS	Hadoop Distributed File System
HTML	HyperText Markup Language
KL	Kullback-Libel
MAP	Mean Average Precision
MRR	Mean of the Reciprocal Ranks
NLP	Natural Language Processing
POS	Part-of-Speech
POSIX	Portable Operating System Interface
REST	Representational State Transfer
RPC	Remote Procedure Recall
RSS	Really Simple Syndication
SVM	Support Vector Machine
URL	Uniform Resource Locator
WWW	*World Wide Web*
5W1H	cinco palavras interrogativas começadas por w e uma por h

Capítulo 1

Introdução

A presente dissertação pretende descrever o trabalho desenvolvido na criação de um sistema de pesquisa, análise e extração de informação sobre tópicos relevantes, no âmbito de processos de inovação de empresas, tendo em conta as contribuições de certos utilizadores em fóruns de discussão.

Neste capítulo introdutório apresenta-se o contexto do projeto, a motivação e a descrição sumária da plataforma desenvolvida subjacentes ao trabalho realizado, bem como os objetivos do projeto. É também apresentada uma visão global sobre a estrutura da dissertação.

1.1 Contexto do Trabalho

No meio empresarial cada vez tem mais importância o processo de inovação pois que esta é considerada como a chave para a sobrevivência e sustentabilidade das empresas a longo prazo, sobretudo as que se inserem em mercados voláteis e exigentes. A fase inicial do processo de inovação (*Fuzzy Front-end*, Fase 0) [WS02] é iterativa e resulta da contribuição de muitos e diferentes agentes sociais como utilizadores, clientes, colaboradores, parceiros de negócio, entre outros, possuindo cada um destes agentes diferentes níveis de conhecimento. As empresas poderão não ser autónomas na geração de novo conhecimento, uma vez que as fontes de novo conhecimento podem ser externas. Com o enorme crescimento da web, verifica-se um aumento do volume de dados e da informação publicada na rede [Liu04]. A grande quantidade de informação disponível *online*, faz com que a web seja uma área propícia à extração de conhecimento [Kos00]. Muitos utilizadores dessa rede contribuem diariamente de forma direta ou indireta com informação valiosa para as empresas, através de plataformas online como blogues, fóruns, redes sociais, grupos de discussão, entre outras. Essas contribuições poderão traduzir-se na forma de opiniões, conhecimento, experiência própria, ideias, soluções para problemas, respostas técnicas e factuais. Fornecer soluções práticas, conseguir reconhecimento social, reputação, auto-expressão ou altruísmo, são exemplos

de razões pelas quais pessoas fora do ambiente empresarial (ou mesmo dentro) contribuem voluntariamente com o seu tempo, energia e conhecimento. Dentro deste conjunto de utilizadores destacam-se os chamados "*Lead Users*" [vH86], os quais desempenham um papel importante no processo de inovação, uma vez que adaptam, melhoram e transformam produtos já estabelecidos no mercado, podendo ainda encontrar novas aplicações para os mesmos. Estes utilizadores distinguem-se ainda dos demais pelo facto de estarem à frente das tendências de mercado e por se beneficiarem a si próprios com a inovação [vH86]. Conseguem sentir as necessidades que ainda são desconhecidas no mercado mas que este beneficiaria se existisse uma solução que as satisfizesse. Os *Lead Users* são caracterizados por Von Hippel [vH86] como sendo:

- utilizadores que estão na linha da frente de cada tendência identificada em termos de novos produtos relacionados e necessidades do processo;
- utilizadores que esperam obter um benefício relativamente alto de soluções para essas necessidades.

Neste contexto a identificação e a recolha das informações proveniente de fóruns de discussão será essencial para a inovação tão desejada nas empresas.

1.2 Motivação

A web tem-se tornado numa grande fonte de informação, bastante diversificada em termos de conteúdos, sendo um meio onde os diversos utilizadores podem disponibilizar os seus conhecimentos e opiniões sobre os mais diversos assuntos. Nesse sentido, grande parte dessa informação pode ser extraída para ser trabalhada pelas empresas e ser geradora de novos conceitos e ideias.

Os fóruns possuem uma grande quantidade de conteúdos gerados pelos utilizadores numa grande variedade de tópicos e é altamente desejável que os dados possam ser extraídos e utilizados [CWLS08].

Neste contexto, o desenvolvimento de uma plataforma escalável que permita a recolha automática destas contribuições dispersas pela rede será proveitoso para o meio empresarial. Assim, este trabalho pretende fornecer conteúdos enriquecedores a partir de fóruns de discussão em determinada área. Para além disso, existe uma motivação adicional pelo facto de contribuir com informação relevante para um *Mashup web 2.0*.

1.3 Plataforma de Extração de Dados de Suporte à Inovação

Esta dissertação pretende alcançar a disponibilização dum serviço Web para extração de informação de grupos de discussão relacionada em certas temáticas, utilizando técnicas de *Big Data*, *Web Crawling* e *Web Mining*, bem como a apresentação de resultados num formato normalizado. Assim, a informação resultante do processo fica disponível para ser utilizada em diversas aplicações ou até ser combinada com outras informações em *Mashups Web 2.0*.

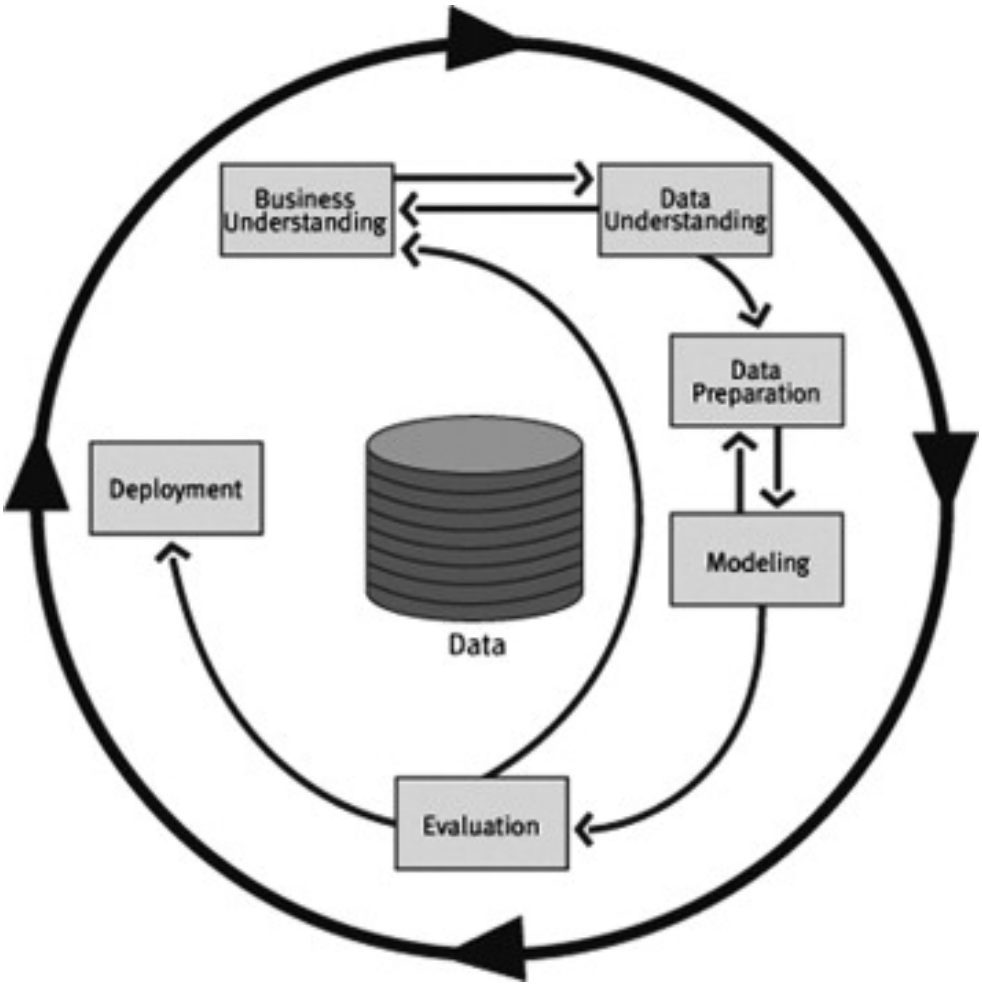


Figura 1.1: Metodologia CRISP-DM

Para a realização do projeto foi necessário definir uma metodologia adequada ao mesmo. Portanto, a que foi utilizada para este trabalho de extração de conhecimento, baseia-se em [WH00].

A metodologia descrita, denominada por CRISP-DM, é uma metodologia compreensiva de mineração de dados e também um modelo de processo que permite a peritos e a novatos nesta área uma maneira de conduzir um projeto desta natureza. O modelo de referência CRISP-DM fornece uma visão geral do ciclo de vida de um projeto de *data mining*.

Como se pode ver na figura 1.1 [1] este processo atravessa as seguintes fases:

1. **Entendimento do negócio**: pretende-se determinar os objetivos do negócio, avaliar a situação, determinar os objetivos ao nível da mineração de dados e produzir um plano do projeto; No início do projeto em questão, foi delineado melhor o problema a tratar bem como os objetivos propostos para o mesmo. Para além disso, foi definido um planeamento referente às diversas etapas que irão ser executadas ao longo do desenvolvimento.

[1] http://info-pro.tumblr.com/post/3566112554/geuder-crisp-dm-model-cross-industry-standard

2. **Entendimento dos dados**: nesta fase é crucial colecionar os dados iniciais, descrever e explorar os dados e verificar a qualidade dos mesmos. Assim, foi efetuado um estudo sobre os fóruns de discussão fornecidos pela empresa Ideia.m e outros fóruns, para se perceber quais os dados transversais a estas plataformas e como estavam organizados. Por outro lado, foi necessário também verificar a disponibilidade dos mesmos. A recolha inicial dos dados ocorreu com a realização do *crawl* em alguns destes fóruns sem ser feito qualquer tipo de tratamento.

3. **Preparação de Dados**: a este nível efetua-se a seleção dos dados, a limpeza, construção, integração e formatação dos mesmos. Neste sentido, foram desenvolvidas funcionalidades capazes de extrair os campos principais do fórum de discussão e foram efetuadas operações de limpeza, formatação e construção de dados.

4. **Modelação**: seleciona-se a técnica de modelação, gera-se o desenho de teste, constrói-se o modelo e avalia-se o mesmo. Com base nos dados extraídos foi desenhado um esquema de dados para armazenar os dados recolhidos. Foram feitos testes para avaliar a veracidade dos dados extraídos.

5. **Avaliação**: avaliam-se os resultados, revê-se o processo e determinam-se os próximos passos. Depois de se reverem os resultados obtidos com os diversos testes à plataforma, foi necessário refinar alguns pontos no pré-processamento dos dados.

6. **Distribuição**: efetua-se um plano de distribuição, de monitorização e manutenção, produz-se um relatório final e revê-se o projeto desenvolvido. Quando se atingir um nível bastante satisfatório relativamente ao tratamento dos dados extraídos será efetuado o *deploy* do sistema.

Na prática muitas das etapas podem ser executadas numa ordem diferente e será necessário por vezes voltar a tarefas anteriores e repetir certa ações. O modelo apresentado pretende tornar os projetos de *data mining*, com menos custos, mais confiáveis, mais flexíveis e rápidos.

1.4 Objetivos

O projeto a ser executado tem como objetivos principais:

- o desenvolvimento de uma plataforma de *crawl* escalável capaz de recolher dados de fóruns de discussão;

- a indexação dos resultados obtidos e pré-processados.

Desta forma, pretende-se que o utilizador deste serviço web seja capaz de obter informação relevante de fóruns de discussão, a partir de determinadas consultas que este efetue ao sistema.

1.5 Estrutura da Dissertação

A presente dissertação encontra-se estruturada em cinco capítulos. Para além deste, o segundo descreve uma breve apresentação de vários conceitos introdutórios, seguida duma descrição detalhada do estado de arte existente nas várias áreas abordadas no projeto: fóruns de discussão, *Web Crawling*, *Web Mining*, sumarização de texto, *Big Data*, computação distribuída e *MapReduce*. A finalizar, são exibidas as diversas tecnologias existentes no mercado, em cada uma das áreas do conhecimento acima referidas. No terceiro capítulo, explica-se a arquitetura da ferramenta desenvolvida, bem como todas as componentes que a integram. No que diz respeito ao quarto capítulo, descreve-se toda a avaliação efetuada da plataforma, até à data. Por último, relatam-se os pontos que serão desenvolvidos no trabalho futuro e as considerações finais.

1.6 Conclusão

Neste capítulo introdutório, descreveu-se o trabalho que foi efetuado, bem como o contexto em que se insere. Foram também delineados os objetivos do projeto e estudadas metodologias adequadas ao mesmo, sendo enunciada uma que se considere apropriada aos projetos desta natureza. Foi ainda definida uma estrutura para o presente documento, tendo por base o trabalho efetuado.

Capítulo 2

Revisão de Literatura

Após ter sido fornecida uma visão geral sobre o documento apresentado e quais os conteúdos a abordar no mesmo, referir-se-ão, nesta secção, os tópicos resultantes do levantamento do estado da arte inerentes ao domínio em estudo. Inicialmente será fornecido algum *background* relativo aos conceitos abordados na dissertação, destacando-se as seguintes temáticas:

- Fóruns de Discussão;
- *Web Crawling*;
- *Web Mining*;
- Sumarização do texto;
- *Big Data*;
- Computação distribuída;
- *MapReduce*;

De seguida serão apresentados trabalhos relacionados com o tema da tese, concluindo-se com uma breve análise comparativa dos mesmos em relação ao projeto desenvolvido. Ao terminar será feito um estudo das tecnologias que poderão ser utilizadas na realização do projeto.

2.1 Background

Nesta secção serão referidos conceitos importantes para a compreensão da dissertação.

2.1.1 Fóruns de discussão

O fórum de discussão é um espaço web dinâmico que possibilita a diversos utilizadores comunicarem. É composto geralmente por diferentes tópicos de discussão onde a primeira mensagem

de um *thread* define a discussão e as mensagens seguintes são, geralmente, contribuições sobre o assunto iniciado.

Quanto à estrutura e ao sistema de classificação das mensagens e dos utilizadores da comunidade, estes diferem muito de fórum para fórum. No entanto, a presença de moderadores é sempre aconselhável para que os conteúdos apresentados nas diversas páginas que compõe o fórum sejam confiáveis.

2.1.2 Web Crawling

A pesquisa na web é atualmente o meio mais adotado para a obtenção de conteúdos e de informação. Isto deve-se à crescente facilidade no acesso à internet e, principalmente, à existência de motores de busca bastante eficazes na pesquisa de páginas com conteúdos relevantes para o utilizador. As visitas às bibliotecas estão a perder cada vez mais o seu espaço e cada vez mais as pesquisas online têm tido um papel fundamental na busca de informação. Assim, com o aumento da utilização deste método de obtenção de informação e com o aumento de informação disponível nesse meio, é necessária a criação de motores de busca eficazes que retornem páginas com conteúdos relevantes. Estes motores integram *Web crawlers*, *spiders*, *robots*, [Liu09], programas que automaticamente descarregam páginas Web.

A informação existente na rede encontra-se disseminada através de milhões de páginas, alojadas em milhões de servidores. O utilizador poderá aceder à informação através de hiperligações existentes, permitindo a este mover-se entre as páginas existentes. Da mesma forma, um *crawler* pode visitar muitas páginas com o intuito de recolher informação que pode ser posteriormente analisada e explorada.

A natureza dinâmica da web implica que não seja suficiente obter a informação e guardá-la num repositório. É preciso que os *crawlers* se mantenham informados sobre as páginas e os links que sofreram alterações. Este tipo de programas tem uma vasto leque de aplicações práticas:

1. Inteligência de negócio: obter informações de concorrentes e de colaboradores;
2. Monitorização de páginas web interessantes: o utilizador é notificado quando existem novas informações.
3. Suporte a motores de busca: coleciona as informações que serão indexadas pelo motor.

Após esta visão geral sobre os *web crawlers* e as suas aplicações, pretende-se mostrar em seguida os passos básicos de execução de um.

1. A lista é inicializada com os *seed* URLs;
2. Em cada iteração, o *crawler* seleciona o próxima URL na fronteira (lista de URLs que ainda não foram visitados pelo *crawler*);
3. Busca a página corresponde do URL através do HTTP;
4. Analisa a página e extrai os seus URLs;

5. Adiciona os novos URLs à fronteira e guarda a página;

Tendo por base a análise efetuada por [Liu09], os *crawlers* podem ser definidos em três grandes tipos: *crawlers* universais, *crawlers* focados e *crawlers* de tópicos. Apresentam-se de seguida cada um destes tipos.

2.1.2.1 Crawler Universal

Para se compreender melhor o conceito de *crawler* universal, é necessário compreender o que é um *crawler* que utiliza uma estratégia primeiro em largura. Sendo assim, um *crawler breadth first search* é aquele que apresenta uma fronteira implementada segundo uma fila *First-In-First-Out* (FIFO), indicando que um novo URL é adicionado à cauda da fila e o próximo a ser rastreado é o que se encontra na cabeça da fila. Deste modo, um *crawler universal*, difere do *crawler* primeiro em largura, ao nível do desempenho, uma vez que necessita de buscar e analisar milhares de páginas por segundo. No que diz respeito à política usada pelos universais estes tentam englobar as páginas da web mais importantes, enquanto mantêm os seus índices o mais possível atualizados.

No trabalho desenvolvido em [HN99], é apresentado um *crawler* escalável e extensível escrito na linguagem java, denominado por *Mercator*. O objetivo deste projeto seria obter os milhões de documentos existentes em toda a web. Conclui-se que a construção de um *crawler* escalável não é uma tarefa trivial, uma vez que os dados manipulados pelo *crawler* são enormes para serem colocados em memória, o que leva à existência de problemas de performance relacionados com o balanceamento entre o uso do disco e da memória.

2.1.2.2 Crawler Focado

Os *crawlers* focados são utilizados quando se pretende pesquisar páginas de uma dada categoria, tentando enviesar a pesquisa de acordo com os interesses do utilizador, não sendo efetuada uma pesquisa por todas as páginas da web. Nesta área têm sido desenvolvidos diversos trabalhos pelo que será dada uma breve descrição de alguns deles. Em [CB99] é proposto um *crawler* deste tipo. Nesta solução, são utilizados dois elementos fundamentais para guiar o processo: um classificador, que avalia a relevância dum documento web relativamente ao tópico em foco, e um destilador, que tem como missão identificar páginas tidas como bons pontos de acesso para muitas outras relevantes, os designados *hubs*. De salientar que os tópicos a investigar são fornecidos ao sistema sob a forma de exemplos que serão apreendidos pelo classificador. Para efetuar a classificação é usada uma taxonomia base. Este modelo proposto apresenta-se capaz de recolher informações relevantes e identificar recursos populares, bem como regiões com uma relevância elevada. Mostrou-se robusto em diferentes situações iniciais e demonstrou ser capaz de encontrar bons dados longe do sítio de origem. Comparando este método com um *crawler* normal verifica-se que os normais tendem a perder-se com o ruído existente, mesmo começando nos mesmos pontos de partida.

Nos trabalhos efetuados em [LPD+09], pretende-se mostrar uma nova abordagem de *crawling* em que a prioridade dada às páginas web a serem exploradas tem em conta dois fatores:

- análise semântica da página web;
- análise dos *links* da página web;

Através da comparação do conteúdo da página com um assunto pretendido pelo utilizador, o *crawler* implementado pretende garantir que as páginas obtidas se enquadram no tópico pretendido. Para além disso, a análise feita ao nível dos *links* contidos na página, consegue reduzir o número de páginas que o *crawler* necessita de efetuar *download*.

Um outro trabalho interessante, [YH11] explicitou um novo algoritmo para melhorar a qualidade dos resultados obtidos por meio de um *crawler* focado. Foi proposto um algoritmo genético que pretende formar coleções de domínios específicos, a ser utilizado por um motor de busca. Verificou-se que a junção da análise dos conteúdos e dos links das páginas web, adicionado à facilidade desta técnica efetuar pesquisa global, ajuda na resolução de certos problemas encontrados pelos *crawlers* focados tradicionais. As experiências desenvolvidas, confirmaram também que com o "*crawler* genético"se consegue atravessar o espaço da web de forma mais abrangente que outros algoritmos de pesquisa nesse espaço.

Noutro trabalho [EGK01], pretendia-se especificar o interesse do utilizador com base numa fórmula lógica usando tópicos de uma taxonomia e com isso medir a relevância de uma página web. Para além disso, visava ainda propor um esquema geral para o processo de *crawling*.

2.1.2.3 Crawler de Tópico

Os *crawlers* de tópico não possuem classificadores de texto para os guiar. O tópico pode ser uma ou mais páginas de exemplos ou até mesmo uma pequena consulta. São bastante úteis quando os exemplos de páginas classificadas não se encontram disponíveis em número suficiente para treinar o *crawler* focado antes de começar o processo.

Neste âmbito foi desenvolvido o seguinte trabalho, [MPS04], que pretendia avaliar os diversos algoritmos de *topical web crawling*. Foram introduzidos dois novos algoritmos o BFS256 e o *InfoSpiders*. Os métodos adaptativos mostraram ter uma performance superior à dos algoritmos tradicionais.

2.1.2.4 Optimizações

Muitas das abordagens de *crawling* estudadas apresentam um objetivo em comum, minimizar o custo de processamento, sendo algumas abordagens mais favoráveis em determinados contextos de utilização. Yang et al. [YCW09] desenvolveram uma metodologia de *crawling* incremental tendo por base uma lista "inteligente" [YCWH09]. Ao contrário das abordagens tradicionais, que se focam essencialmente na calendarização da revisita de páginas a nível individual, este método pretende revisitá-las tendo em conta o conhecimento do mapa do sítio, através da sua reconstrução. A abordagem apresentada pretende juntar as mensagens da mesma *thread*, distribuídas em várias páginas de acordo com o tempo de colocação que apresentam. De seguida, para cada *thread* é aplicado um modelo de regressão com a finalidade de prever o tempo de chegada de uma nova

mensagem. Apresenta-se com um desempenho acima de 260% sobre alguns métodos existentes. Outros estudos como o apresentado em [CGM98] analisaram várias formas de ordenar os URLs existentes na fronteira, de forma a otimizar a obtenção das páginas mais importantes, em primeiro lugar. Foram aplicadas várias métricas e verificou-se que o *PageRank* é uma excelente métrica quando são usadas páginas com *backlink* ou *PageRank* elevado. *Backlinks* são links de entrada para uma página web. A contagem de *backlinks* é uma das métricas utilizadas para medir a importância de uma página. Segundo Cho e Garcia-Molina, [CGM98], esta contagem indica o número de links duma página que se encontram na web. De acordo com [Bri98] *PageRank* é um algoritmo que usa a estrutura de *links* da web para calcular o ranking de qualidade para cada página web. O *PageRank* de uma página é definido recursivamente através do número e da métrica *PageRank* das páginas que estão ligadas a ela. Segundo a definição em [Bri98] o *PangeRank* fornece uma aproximação da importância de uma página não apenas pela contagem das ligações (*backlinks*) mas também reconhece que a importância desses links não deve ser igual por todas as páginas. Esse valor deve ser normalizado tendo em conta o número de ligações que a página tem.

Em [CM04], definiu-se como objetivos, testar certas estratégias para ordenar páginas web durante o processo de *crawling*, num simulador de um *crawler* para poderem ser comparadas. A técnica com melhor desempenho será testada num *crawler* real e será avaliado o facto de as páginas mais importantes serem exploradas numa fase inicial do *crawl*. Com base nos resultados obtidos através do simulador definiu-se que o algoritmo que ordena as páginas com base na profundidade era o melhor, uma vez que apesar da sua simplicidade é bastante bom para cobrir de forma eficiente grande parte do domínio em teste. Para confirmar esta teoria, o algoritmo foi corrido num *crawler* real e verificaram-se os mesmos resultados.

Num outro trabalho [OP08] o processo de *recrawl* é otimizado tendo em conta o conceito da longevidade da informação. Neste caso pretende-se maximizar: $informaocorreta \times tempo$.

Menczer et al, em [MPR] efetuaram uma avaliação com base em três algoritmos de *crawling* (agentes adaptativos - *InfoSpiders*, ranking de similaridade- *Best First* e análise de links - *PageRank*) e com três métodos de teste (classificadores, sistema de recuperação - *SMART* e similaridade média do tópico). Conclui-se que os três testes efetuados revelaram resultados interessantes. No entanto, a avaliação por meio de classificadores podia ser melhorada devido ao facto de no conjunto de treino os exemplos negativos serem de páginas fora do tópico. Por outro lado, os testes com base no sistema de recuperação e na média da similaridade do tópico forneceram informações sobre a dinâmica do *crawler*. Apesar destes reparos acerca do modo como foi efetuada a avaliação, descobriu-se que o *Best First* foi o melhor algoritmo comparando com os outros dois. A explicação para este resultado deve-se ao facto deste algoritmo, que é conduzido pelo tópico, possuir apenas, como forma de ordenar os *links* existentes na fronteira, a similaridade com o mesmo.

O fraco resultado obtido pelo *PageRank* pode ser explicado pelo facto de que o algoritmo favorece páginas de autoridade num contexto geral, sem considerar o tópico. É uma métrica muito geral para uma tarefa que é guiada pelo tópico.

No que diz respeito ao *InfoSpiders*, verificou-se que apesar de os agentes de *crawler* usarem a similaridade para convergirem numa certa zona da web e de usarem redes neuronais para discri-

	Web Mining			
	Web Content Mining		Web Structure Mining	Web Usage Mining
	IR View	DB View		
View of data	- Unstructured - Semi sctrutured	- Semi structured - Web site as DB	- Links structure	- Interactivity
Main data	- Text documents - Hypertext documents	- Hypertext documents	- Links structure	- Server Logs - Browser Logs
Representation	- Bag of words, n-grams - Terms, phrases - Concepts or ontology - Relational	- Edge-labeled graph (OEM) - Relational	- Graph	- Relational Table - Graph
Method	- TFIDF and variants - Machine learning - Statistical (NLP)	- Propietary algorithms - ILP - (Modified) association rules	- Proprietary algorithms	- Machine Learning - Statistical - (Modified) association rules
Application Categories	- Categorization - Clustering - Finding extraction Rules - Finding patterns in text - User modeling	- Finding frequent sub-structures - Web site schema discovery	- Categorization - Clustering	- Site construction, adaptation, management - Marketing - User modeling

Figura 2.1: Categorias do web mining

minar os *links* duma página, os resultados apresentados não foram satisfatórios. No entanto, foi efetuado um teste adicional com o sistema de recuperação em que foram recolhidas as cinquenta melhores páginas retornadas pelo sistema e verificou-se que este algoritmo apresentava o maior número de páginas dentro deste conjunto. Assim, confirmou-se no que previamente se sabia que esta abordagem permite cobrir o maior número de páginas relevantes.

2.1.3 *Web Mining*

O principal objetivo do *web mining* é descobrir padrões em conteúdos web. De acordo com esta área do conhecimento pode ser dividido em três grandes sub-áreas de acordo com o seu objetivo de análise: [Kos00]

- *Web Content Mining*: extração e integração de dados úteis de conteúdos das páginas web;
- *Web Usage Mining*: É a análise dos registos de navegação dos utilizadores nos documentos da internet. A principal utilização é a descoberta de padrões de navegação que podem ajudar a melhorar a navegabilidade dos sites.
- *Web Structure Mining*: estuda o relacionamento entre as páginas web através dos seus *links*;

A figura 2.1 apresenta sumariamente os vários métodos e técnicas referentes as estas três categorias.

2.1.3.1 Extração de dados estruturados

A extração de dados estruturados também conhecida como *Wrapping*, pode ser feita de três maneiras:

- Abordagem Manual: pela observação da página web e o seu código fonte, o programador encontra alguns padrões e escreve um programa para extrair os dados alvo.
- *Wrapper induction*: abordagem de aprendizagem supervisionada e semi-automática. Consiste num conjunto de regras de extração aprendidas a partir de uma coleção de páginas classificadas manualmente ou de registos de dados. As regras são aplicadas para extrair itens de dados alvo a partir de outras com formato similar.
- Extração automática: abordagem não supervisionada. Dado uma ou várias páginas, automaticamente encontra padrões ou gramáticas para extração de dados.

Um exemplo prático deste tipo de extração é descrito em [LH05] na extração de comentários das páginas web. Neste caso foi utilizada uma extração automática fornecida pela ferramenta MDR-2, que possibilita a obtenção de campos de dados individuais nos registos de dados existentes [Zha05]. O MDR-2 pode-se definir em dois passos:

1. a identificação dos registos de dados individuais dentro duma página;
2. o alinhamento e extração dos itens a partir dos registos.

Na etapa inicial é usado um método baseado na informação visual e na etapa final uma nova abordagem de alinhamento parcial baseado na correspondência da árvore. Este alinhamento parcial significa apenas alinhar os campos de dados num par de registos de dados que podem ser correspondentes, quando existe certeza, e não estabelecer nenhum compromisso com o resto dos campos. Para se utilizar este tipo de abordagem, é necessário ter em atenção que os sítios onde se vai efetuar a extração devem ter *templates de layouts* fixos.

Numa outra abordagem [MMKR], desenvolveu-se o chamado *stalker*, que apenas tem de encontrar os pontos de referência que aparecem na maioria das vezes na proximidade imediata dos itens a serem extraídos.

2.1.3.2 Extração de dados não estruturados

Neste âmbito existem bastantes abordagens relativas à extração de dados. No trabalho desenvolvido em [HZ07], pretendia-se extrair pares de alta qualidade <Título, Resposta> como conhecimento de *chat* a partir de fóruns com uma *framework* em cascata. Muito sumariamente o processo decorrente baseia-se em duas fases principais. Numa primeira etapa, as respostas relevantes relativamente ao título da *thread* da mensagem raiz, são obtidas através de um classificador *Support Vector Machine* (SVM), baseando-se em correlações na estrutura e no conteúdo.

Nesta primeira parte do procedimento, foram utilizadas como caso de comparação todas as mensagens entre a raiz e as correspondentes respostas. Após um pré-processamento dos dados

iniciais, constatou-se que com as características estruturais o sistema apresenta valores de desempenho altos. No entanto, as características de conteúdo fizeram decrescer a performance quando aplicadas sozinhas. Por fim, ao se adicionar ambas as características verificou-se que a precisão obtida tinha aumentado, enquanto que o *recall* permaneceu com os mesmos valores, indiciando que as características de conteúdo ajudam a melhorar a precisão.

Cong et al. [CWLS08] propõem um modo alternativo de classificar as questões, uma vez que nem todas estas aparecem da mesma forma, em fóruns online, não sendo obrigatória a presença de pontos de interrogação nem de palavras interrogativas "5W1H". Nesta abordagem as questões são identificadas tendo por base uma extração automática de padrões sequenciais.

Ao nível das respostas, é proposta uma abordagem baseada em grafos não supervisionada para ordenar as respostas candidatas. Para esta ordenação foram utilizados diversos métodos que vão ser explicitados posteriormente nesta secção. Foram feitos algumas experiências visando a deteção de questões e respostas. Relativamente à deteção de questões utilizaram-se para a comparação métodos onde a identificação era feita através do 5WH1, por meio do ponto de interrogação e o método SM explorado em [SM04].

As métricas utilizadas para medir o desempenho das várias abordagens foram a precisão, o *recall* e o *F1-score*. Verificou-se que o método de identificação através do ponto de interrogação apresenta uma boa precisão mas um *recall* baixo. O método proposto supera todos os outros tendo mostrado melhoramentos significativos. A principal razão apontada para este sucesso é o facto de este método descobrir padrões sequenciais classificados, permitindo detetar de forma flexível as mais diversas questões. No que diz respeito à deteção das respostas utilizaram-se as métricas *Mean Average Precision*(MAP), *Mean of the Reciprocal Ranks* (MRR) e a precisão@1 [CWLS08]. O MAP pode ser definido como

> “ média da média das precisões computadas depois de truncar a lista, após cada uma das respostas corretas, para um conjunto de perguntas .” [CWLS08]

Enquanto que o MRR como

> “ média das posições recíprocas das primeiras respostas corretas sobre um conjunto de questões.” [CWLS08]

.

Para se ter uma ideia da aplicação desta métrica, ela indica o quanto se tem de olhar para baixo numa lista ordenada, a fim de encontrar uma resposta correta. A grande diferença entre as duas métricas é que o MAP considera todas as respostas corretas e o MRR só considera a primeira resposta correta. A precisão@1 define-se como

> “ a fração do top-1 das respostas candidatas obtidas que são corretas.” [CWLS08]

Foram utilizados vários métodos de comparação, destacando-se a métrica de divergência de *Kullback-Libel* (KL) e a sua combinação com o grafo do modelo de linguagem, que superaram a *baseline* oferecida pelo método "resposta mais próxima".

Os resultados experimentais em dados reais mostram que esta abordagem é eficaz.

Por exemplo, no caso de [Hu06], onde existiam dados semi-estruturados, para se extrair características dos produtos a partir dos vários formatos de opiniões correu-se um *NLProcessor*, aplicação que trabalha com texto em linguagem natural e apresenta como objetivo a geração de marcas *Part-of-Speech* (POS). Depois de serem catalogadas as opiniões com as *tags*, para as opiniões onde se encontravam dados não estruturados, efetuou-se a identificação em primeiro lugar, de substantivos ou frases nominais como características, usando regras de associação. As características frequentes são usadas com a finalidade de encontrar potenciais adjetivos de opinião, que são empregues para extrair características não tão frequentes associadas. Por outro lado, o método acima referido não trabalha bem com a identificação de vantagens e desvantagens dos produtos, quando estas não são dadas diretamente, uma vez que não encontra características implícitas (características que não são substantivos). Propõe-se então um método de mineração supervisionado para gerar padrões de linguagem a partir dos comentários de treino e usar esses padrões para extrair características do produto dos comentários de teste. Na mineração de regras de associação das classes podem gerar-se padrões. Estes padrões permitem extrair o substantivo como uma característica do produto se for seguido pelo verbo *is* e um adjetivo.

O uso de padrões de linguagens geradas para extrair características a partir de pequenas frases leva a resultados bastante precisos, contudo, não dão tão bom resultado em frases completas. A principal razão para tal acontecer é que o sistema gera muitos padrões, sendo difícil saber que padrões usar, dada uma frase. Ao nível de resultados obtidos, verificou-se que, considerando os verbos e adjetivos como possíveis características e tendo as orientações de opinião divididas à priori, obtiveram-se os melhores resultados na identificação nas expressões favoráveis aos produtos tanto em termos de reinvocação como precisão.

Num trabalho muito idêntico ao que foi descrito anteriormente, Liu e Hu [LH05] desenvolveram uma *framework* onde se propõem analisar e comparar as opiniões dos consumidores sobre produtos concorrentes e propõem uma novo método baseado na mineração dos padrões de linguagem para extrair características do produto a partir dos prós e contras num tipo particular de comentários. Os resultados deste projeto são apresentados com mais detalhe na secção 2.2.

2.1.4 Sumarização do Texto

No trabalho desenvolvido em [Hu06] o sumário do texto é feito com base em técnicas clássicas de sumarizar o texto.

> "O sumário das opiniões é feito de forma estruturada em vez parágrafos curtos ou com texto solto.." [Hu06]

Em [HL] é apresentada uma ferramenta para sumariar texto de uma automatizada e multilingual *SUMARIST*. O objetivo deste sistema é criar sumários de textos aleatórios e selecionar outras línguas. Este mecanismo utiliza técnicas de *parsing* específicas da linguagem e uma análise semântica, e combina processamentos de NLP com o léxico *SENSUS* derivado do *WordNet* e aumentado com dicionário e outros recursos.

De forma breve, o *SUMARIST* para extrair conteúdos, precisa primeiro de identificar as passagens mais importantes do texto introduzido. Neste fase são retidos os tópicos mais importantes e centrais. Numa segunda etapa elabora os sumários, executando um processo de interpretação. Neste processo, é realizada uma compactação através da re-interpretação e fusão dos tópicos previamente extraídos. Por fim, numa fase de geração, é produzido o resultado do sumário. No caso de extrações, este passo não tem significado. No caso de ser produzido um resumo, o gerador tem de reformular o texto obtido da fase anterior num novo texto coerente.

Relativamente aos testes efetuados, é de realçar que ao nível da extração obtida se conseguiu uma pontuação média trinta por cento acima dos sumários efetuados por humanos e a seleção aleatória de frases foi tão boa como os resumos obtidos.

Um outro trabalho nesta área [RM98], apresenta uma ferramenta designada por *SUMMONS* que sumaria as notícias sobre eventos correntes. Gera sumários de múltiplos documentos nos mesmos eventos ou semelhantes, apresentando similaridades e diferenças, contradições e generalizações entre as fontes de informação.

2.1.5 *Big Data*

Com base no trabalho produzido em [Pur], *Big Data* pode ser considerada uma coleção de conjuntos de dados enormes e complexos que se torna difícil armazenar em bases de dados convencionais. Pode incluir dados estruturados, semi-estruturados e não estruturados.

No mundo empresarial, este tipo de dados é importante na medida em que permite:

- criação de transparência;
- melhoramento do desempenho;
- segmentação da população;
- suporte à tomada de decisão;
- inovação nos modelos de negócio, produtos e serviços.

A análise de *Big Data* abre novas oportunidades e desafios para a área empresarial. Fontes de dados que anteriormente não eram processadas, tornaram-se capazes de serem guardadas e processadas. No que diz respeito a dados não estruturados podem agora ser guardados de uma forma mais conveniente e num formato mais adequado, podendo ser aplicadas técnicas de pesquisa de texto.

Numa reflexão efetuada em [BCL12] descreveram-se problemas arquiteturais, em componentes e camadas que tem sido desenvolvidas recentemente e como elas têm sido usadas para tratar os desafios anunciados do *Big Data*.

2.1.6 Computação Distribuída

Pode ser definida como computação paralela e descentralizada, realizada por dois ou mais computadores ligados através duma rede, tendo como objetivo a conclusão de uma tarefa comum.

A computação distribuída pretende adicionar poder computacional de diversos computadores interligados através de uma rede de computadores.

2.1.7 *MapReduce*

O paradigma *MapReduce* pode ser definido como um modelo de programação moldado para processar grandes quantidades de dados. Maioritariamente é utilizado para computação distribuída num *cluster* de computadores, onde são executadas paralelamente operações de *map* e *reduce*.

Segundo [RR07] este paradigma permite aos programadores desenvolver código que é automaticamente executado de forma paralela e temporizado num sistema distribuído.

A operação de *map* caracteriza-se pelos seguintes passos:

1. o nó master recebe o *input*;
2. divide-o em sub-problemas de pequena dimensão;
3. distribui-os pelos nós trabalhadores;

Por sua vez, a operação de *reduce* executa as etapas:

1. o nó master recolhe todas as respostas de todos os sub-problemas previamente definidos;
2. combina-os para produzir um resultado;

No estudo elaborado por [RR07] pretendeu-se avaliar o modelo *MapReduce* para sistemas multi-core e multi-processador. Foi dada a conhecer a implementação do *Phoenix* que utiliza o *MapReduce* para sistemas de memória partilhada, com o intuito de minimizar os *overheads* nas tarefas de geração e de comunicação de dados. Para aferir das capacidades deste sistema foram efetuados alguns testes, comparando esta plataforma com código escrito para ser executado de forma paralela em *pthreads*. Retirou-se como resultado desta comparação o facto de que apesar dos *overheads* do *runtime*, o *Phoenix* parece ter uma performance muito semelhante para as várias aplicações, fornecendo, no geral, uma abordagem útil de programação e de gestão de concorrência em sistemas de memória partilhada. No entanto, existem aplicações para o qual o modelo *MapReduce* não se enquadra tão bem e onde se verificou que as pthreads seriam a melhor opção.

Em [CCA$^+$] é apresentada uma implementação modificada da arquitetura *MapReduce* de forma a que os dados sejam fluídos através de uma *pipeline* entre os operadores. Como tal, pretende-se reduzir os tempos de conclusão dos *jobs* e otimizar a utilização dos mesmos por parte do sistema. As vantagens que este sistema pretende trazer são:

- os *reducers* começarem a processar dados o mais rápido possível, após a produção de resultados dos *mappers*;
- o fornecimento de *continuous queries* onde os *jobs* que correm de forma continua aceitam novos dados à medida que vão sendo obtidos e que podem ser imediatamente analisados;
- criar uma *pipeline* para o fornecimento dos dados aos operadores de forma mais rápida.

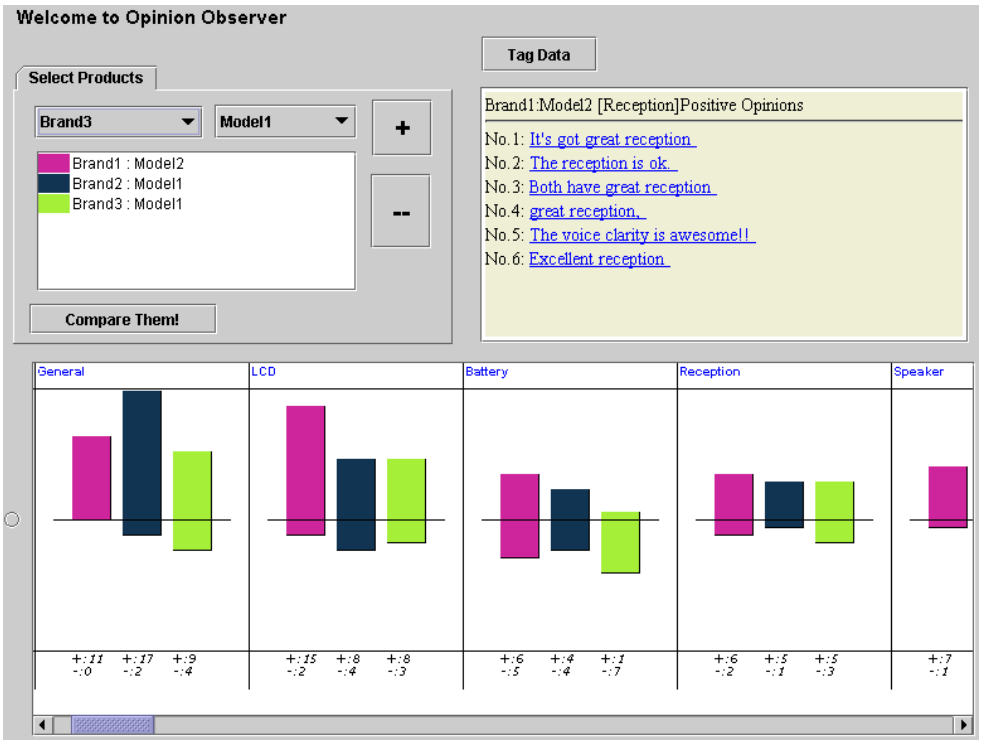


Figura 2.2: Interface do protótipo *Opinion Observer*

Em [EPF08] foi aplicado o paradigma *MapReduce* em dois analisadores de dados científicos. Para além disso, foi desenvolvida uma implementação deste tipo baseada em *streaming* e comparam-se os resultados com o *Hadoop* aplicando estes dois analisadores. Conclui-se que algumas funcionalidades como a necessidade de aceder a dados binários, o uso de diferentes linguagens de programação e o uso de algoritmos iterativos inseridos em aplicações científicas, pode limitar a aplicabilidade das implementações já existentes do *MapReduce* para tarefas de processamento destes dados.

2.2 Trabalhos Relacionados

Durante a pesquisa efetuada, verificou-se a existência de trabalhos relacionados com o tema desta dissertação ao que irá ser produzido, mas com algumas diferenças em vários níveis. Será apresentada uma descrição desses projetos, seguida de uma análise crítica face aos objetivos desta dissertação.

No trabalho explicitado em [DE05], pretende-se mostrar uma representação alternativa dos dados existentes em fóruns de discussão. O âmbito deste estudo enquadra-se num domínio académico de cursos *online* onde estudantes e instrutores interagem. Utilizaram-se técnicas de *text mining* e *data mining* para melhorar o entendimento que o instrutor tem da evolução da discussão que existe nas várias *threads*.

Os autores sustentam a ideia que diversos problemas, já relatados na literatura, contribuem para a dificuldade de avaliar a atividade existente num fórum e fornecer aos seus estudantes *feedback* relevante sobre os progressos obtidos.

O instrutor muitas vezes tem de criar mecanismos manuais para extrair os dados que pretende a partir do dados oferecidos pelo fórum, tornando este processo moroso e difícil.

Por exemplo, com este sistema o instrutor pode ter acesso a dados relacionados com o tempo, o ritmo e a sequência das trocas de contribuições que vão sendo feitas.

Esta extração dos dados, torna-se uma tarefa árdua devido à restrita organização das discussões nas *threads*. Para além disso, é muito comum verificar-se nas discussões existentes nos fóruns a fragmentação, a descontinuidade e até a perda de contexto da discussão iniciada.

Um outro projeto relacionado, foca-se nos comentários de clientes online sobre produtos. A interface deste projeto pode ser visualizada na figura 2.2 [LH05]. Fornece duas contribuições principais: propõe uma nova *framework* para analisar e comparar as opiniões dos consumidores sobre produtos competidores e propõe uma nova técnica baseada na extração de padrões de linguagem para obter características do produto a partir dos prós e contras num tipo particular de comentários. Para um potencial cliente, esta aplicação permite comparar produtos lado-a-lado e característica-a-característica com base nas opiniões dos consumidores destes produtos. Quanto ao fabricante do produto a comparação permite-lhe facilmente recolher inteligência de Marketing e informação sobre a avaliação comparativa do produto. Com uma olhadela na visualização do sistema o utilizador pode claramente ver as forças e as fraquezas de cada produto na perspetiva do consumidor. A comparação de opiniões é importante tanto para clientes como para fabricantes do produto. Foram analisadas opiniões de três tipos de formatos:

- Formato (1) – Prós e Contras: o comentador descreve os Prós e Contras separadamente;
- Formato (2) - Prós, Contras e comentário detalhado: o comentador descreve separadamente os Prós e Contras e também escreve;
- Formato (3) – Formato livre: o comentador pode escrever livremente. Exemplo: não há separação de Prós e Contras.

Neste trabalho [LH05], o método é baseado em processamento de linguagem natural e na descoberta de padrões supervisionada. É também demonstrado que as técnicas atuais não são adequadas para este formato uma vez que utiliza curtas frases ou frases incompletas, nos prós e contras. Não foram analisados comentários detalhados do formato dois por serem elaborações de Prós e Contras. Ao analisar-se segmentos curtos em Prós e Contras produzem-se resultados mais precisos. De notar que este sistema de visualização é aplicável aos três formatos. Dado um conjunto de produtos (que podem ser da mesma marca ou de diferentes) e um conjunto de URLs de *websites* que contêm comentários de clientes, o *Opinion Observer* trabalha em duas etapas:

1. Extrair e analisar os comentários dos clientes em dois passos:

(a) Este passo automaticamente conecta e faz o *download* de todos comentários de clientes das páginas dadas. Por consequência, o sistema monitoriza essas páginas para periodicamente efetuar *download* de novos comentários se existirem. Todas os comentários em bruto são guardados na base de dados.

(b) Todos os novos comentários (que ainda não foram analisados antes) de cada produto são analisados. Duas tarefas são realizadas, a identificação das características do produto e as orientações das opiniões a partir de cada comentário. Isto pode ser feito de forma automática ou semi-automática.

2. Baseado na análise dos resultados, diferentes utilizadores podem visualizar e comparar opiniões de diferentes produtos usando uma interface.

A etapa um é executada pelo sistema ou em conjunto com analistas humanos. A etapa dois é feita por qualquer um que tenha permissão para ver os resultados.

Foram realizados testes a comentários do formato(2). Foi determinado o tempo recuperado da marcação semi-automática relativamente à manual. As métricas de avaliação foram a reinvocação (r) e precisão (p). A partir dos resultados obtidos, pode-se verificar que os Prós possuem melhores resultados que os Contras. Conclui-se que as pessoas tendem a usar palavras semelhantes como "excelente" e "bom" nos Prós para várias características do produto. Ao contrário, para os Contras as palavras usadas diferem muito. Testes para marcação semi-automática: Marcação manual - o analista lê e extrai manualmente cada característica e decide a orientação da opinião; Marcação semi-automática - o analista apenas corrige os erros. Foram feitos testes com 2 marcadores humanos que mostraram que a quantidade de tempo salvo pelo segundo método é por volta de 45%. Sem a interface visual, o método manual consumia mais tempo. Quanto ao agrupamento de sinónimos, o método apresentado atinge 52% de recordação e 100% de precisão para estes dados. O problema é que não lida com sinónimos dependentes do contexto.

O trabalho proposto por Glance et al. [GHNS05], descreve um sistema que recolhe determinados tipos de conteúdos e mostra estatísticas baseadas em métodos, *NLP*, encontro de frases e outras técnicas de exploração de dados que possam ser utilizados no âmbito da inteligência de marketing. A análise efetuada pelo sistema permite ao utilizador caracterizar os dados que tem em mãos e descobrir alguns problemas. Este sistema fornece dados qualitativos ou quantitativos de características a partir de mensagens online.

2.2.1 Discussão dos Trabalhos Relacionados

O trabalho [LH05] mostra-se como uma ferramenta capaz de extrair informações das opiniões dos utilizadores, desviando-se um pouco da temática deste projeto na medida em que nele se pretende analisar discussões nos grupos apropriados. No entanto, apresenta-se como sendo uma ferramenta de extração, classificação e apresentação das opiniões existentes em vários sítios web.

O trabalho [GHNS05] é uma aplicação que pretende configurar a informação colecionada a partir da discussão na *net* segundo um domínio alvo e classificá-la através de certos parâmetros. Por fim efetua uma breve análise das combinações dos tópicos.

No trabalho apresentado em [DE05], pretendia-se atingir três objetivos:

- discutir os problemas gerais relacionados com o sistema que contribuem para a dificuldade de avaliar fóruns de discussão assíncrona;
- identificar indicadores de participação comuns que os instrutores desejem extrair do fórum como dados para avaliar o progresso do estudante e o desempenho nas discussões online;
- descrever os dados e *text mining* como uma estratégia para avaliar fóruns, particularmente ao fornecer vistas configuráveis dos dados.

Sendo assim, o grande objetivo deste trabalho era utilizar ferramentas de *text mining* e de visualização para facilitar a avaliação dos conteúdos do fórum por parte do instrutor. Deste modo, o projeto desenvolvido visa fornecer um *output* mais refinado do fórum para mostrar as tendências os padrões, sumários e estatísticas de modo a demonstrar a extensão da contribuição e progresso num fórum.

Comparativamente com a presente dissertação, esta não pretende utilizar técnicas de mineração tendo em vista a otimização da visualização dos conteúdos do fórum para facilitar a avaliação dos mesmos, mas sim fornecer os conteúdos existentes no fórum.

2.3 Tecnologias

Foi elaborado um estudo das tecnologias existentes no mercado para ser possível averiguar qual delas se adequaria melhor ao problema em questão. Como tal, foram analisadas ferramentas ao nível de duas áreas: *Web Crawling* e *Big Data*. O estudo pode ser visto no anexo A.

2.3.1 *Web Crawling*

Existe um vasto leque de tecnologias na área de *web crawling*. Muitas das ferramentas pesquisadas são *open source* e baseadas na tecnologia Java. Uma outra característica apreciável e que é comum a muitos dos *crawlers* comerciais é o facto de priorizarem os URLs que ainda não foram explorados, de forma a obter as páginas mais relevantes primeiro. Uma outra característica importante dos *crawlers* é o facto de permitirem o *multithread* das operações de forma a reduzir o esforço de *crawling*. Para facilitar a utilização dos sistemas, algumas das aplicações oferecem uma interface gráfica. Tendo em conta estas características, o *Nutch* [1]. parece ser a melhor solução, uma vez que se apresenta como sendo uma aplicação que pode ser integrada com o sistema *Hadoop* [2] e efetuar um *crawler* distribuído. Para além de ser escalável, possui um *parser* de HTML e de outros formatos sendo também bastante robusto.

[1] http://nutch.apache.org/about.html
[2] http://hadoop.apache.org/

2.3.2 *Big Data*

Nas duas áreas acima referidas, algumas das ferramentas enunciadas servem para o processamento de *big data*. De seguida, serão exploradas as características das plataformas para o tratamento deste tipo de dados usadas no projeto.

2.3.2.1 *Hadoop*

Uma das ferramentas bastante conhecidas para armazenamento de grandes conjuntos de dados de forma confiável é o *Hadoop* que se caracteriza-se por ser escalável e por ser aplicada em computação distribuída. Utiliza o paradigma *MapReduce* na sua estrutura, caracterizado por ser utilizado em plataformas como *yahoo* entre outras.

A partir da documentação fornecida pela comunidade *Hadoop*, [had13] foram retiradas as seguintes informações sobre a plataforma.

O *Hadoop Distributed File System* (HDFS) foi desenhado para correr num hardware genérico/comum. As principais diferenças para outros sistemas de ficheiros distribuídos são:

- altamente tolerante a falhas;
- desenhado para ser implementado em hardware de baixo custo;
- processamento de acesso aos dados da aplicação elevado;
- apropriado para aplicações com conjuntos de dados grandes;
- relaxa alguns requisitos POSIX (Portable Operating System Interface) para permitir o fluxo contínuo de acesso aos dados do sistema de ficheiros.

Ao nível das falhas de hardware pode-se dizer que, devido ao facto de existir um grande número de componentes no sistema e de cada um deles ter uma determinada probabilidade de falhar, então qualquer um deles no HFDS pode ser não-funcional e portanto, a deteção de falhas e a rápida e automática recuperação dos erros são os objetivos fundamentais do mesmo.

O HDFS está desenhado para processamento em *batch* em vez de interativo, usado pelos utilizadores. Dá ênfase ao alto fluxo de dados, em vez da baixa latência no acesso aos mesmos. O POSIX impõe vários requisitos árduos que não são necessários para aplicações dirigidas pelo HDFS. A semântica de POSIX nalgumas áreas chave foi remodelada para aumentar as taxas de transferência de dados.

As aplicações que correm o HDFS têm grandes conjuntos de dados. Um ficheiro típico no HDFS tem o tamanho entre *gigabytes* e *terabytes*. Deste modo, o HDFS é ajustado para suportar ficheiros grandes. Deve fornecer uma largura de banda de dados agregada e escalar centenas de nós num único *cluster*. Pode suportar dezenas de milhões de ficheiros numa única instância.

As aplicações HDFS necessitam de um modelo de acesso " ler uma vez escrever muitas " para os ficheiros. Esta infraestrutura pressupõe que um ficheiro uma vez criado, escrito e fechado não necessita de ser alterado (esta suposição simplifica os problemas de coerência de dados e permite

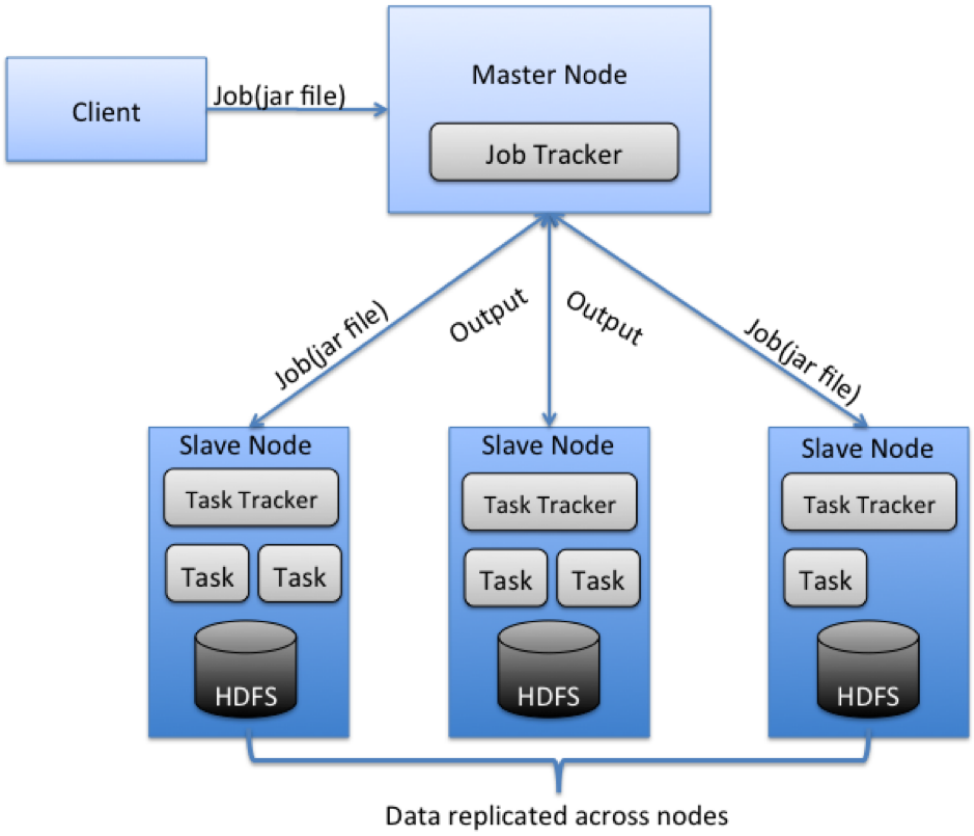


Figura 2.3: Arquitetura do HDFS

uma transferência alta de acesso aos conteúdos). A aplicação *MapReduce* ou um *web crawler* ajustam-se perfeitamente a este modelo. Existe um plano para suportar a adição de escrita nos ficheiros no futuro.

Uma computação requerida por uma aplicação é muito mais eficiente se for executada próxima dos dados em que esta opera. Isto torna-se realmente importante quando o tamanho dos conjuntos de dados é enorme, fazendo minimizar a congestão da rede e aumentar o rendimento global do sistema. A suposição é que frequentemente é melhor migrar a computação para perto dos dados do que mover os dados para onde se encontra a correr a aplicação. O HDFS fornece interfaces para as aplicações se moverem perto da localização dos dados.

O HDFS está desenhado para ser facilmente movido de uma plataforma para outra, facilitando a adoção muito difundida do HDFS como uma plataforma de referência para um grande conjunto de aplicações.

Ao nível da arquitetura, o HDFS apresenta-se como *master/slave*, como pode ser visto na figura 2.3.

Assim, pode-se definir um *cluster* neste sistema da seguinte forma: um único *NameNode*, um servidor *master* que gere o *namespace* do sistema de ficheiros e regula o acesso aos ficheiros por parte dos clientes. Por outro lado, existem *DataNodes*, normalmente um por nó num *cluster*, que gerem o armazenamento anexado aos nós onde correm. Por sua vez, um ficheiro armazenado no

HDFS é dividido num ou mais blocos que são guardados num conjunto de *DataNodes*.

Relativamente às operações, o *NameNode* apresenta como essenciais:

1. operações do *namespace* do sistema de ficheiros (ex: abrir, fechar e renomear ficheiros/diretorias);
2. determinar o mapeamento dos blocos para os *DataNodes*.

Os *DataNodes* apresentam como operações:

1. servir os pedidos de escrita e leitura dos sistemas de ficheiros dos clientes;
2. executar a criação, remoção e replicação segundo instruções do *NameNode*.

Estas máquinas normalmente correm no sistema operativo GNU_ LINUX. O *Hadoop* foi construído com a linguagem java. Qualquer máquina com java pode correr o *NameNode* ou um *DataNode*. A distribuição típica que é efetuada é ter uma máquina dedicada que só corre o *NameNode*. Cada uma das máquinas no *cluster* corre uma instância do software do *DataNode*.

A arquitetura não impede que múltiplos *DataNodes* corram na mesma máquina mas é uma situação rara.

Para acrescentar o *NameNode* pode ser considerado como o árbitro e o repositório para todos os metadados do HDFS. Deste modo os dados do utilizador nunca fluem para ele.

O *HDFS* suporta uma organização de ficheiros hierárquica tradicional. Um(a) utilizador/aplicação pode criar diretórios e guardar ficheiros dentro destes diretórios. Esta hierarquia do *namespace* do sistema de ficheiros é semelhante à maioria dos outros sistemas de ficheiros existentes. Pode-se:

- criar e remover ficheiros;
- Mover um ficheiro de um diretório para outro diretório;
- Renomear um ficheiro.

O HDFS não implementa cotas de utilizador e não suporta *hard* nem *soft links*. O *NameNode* mantém o *namespace* do sistema de ficheiros. Qualquer mudança no *namespace* ou nas suas propriedades é registada pelo *NameNode*. Uma aplicação pode especificar o número de réplicas de um ficheiro que deve ser mantido no HDFS. O número de cópias de um ficheiro é designado por fator de replicação desse ficheiro. Esta informação é guardada pelo *NameNode*.

O HDFS está desenhado para guardar de forma confiável ficheiros grandes através das máquinas num grande *cluster*. Ele guarda cada ficheiro como uma sequência de blocos. Todos os blocos de um ficheiro, exceto o último, são do mesmo tamanho. Os blocos são replicados para a tolerância a falhas. O tamanho dos blocos e o fator de replicação são configurados por ficheiro. O fator de replicação pode ser especificado aquando da criação do ficheiro e pode ser alterado posteriormente. Os ficheiros no HDFS são de escrita única e têm um único escritor a qualquer hora. O

NameNode toma todas as decisões em relação à replicação de blocos. Ele recebe periodicamente um *Heartbeat*, que implica que o *DataNode* está a funcionar corretamente, e um *Blockreport*, que contém uma lista de todos os blocos num *DataNode*, de cada *DataNode* existente no *cluster*.

No que diz respeito à colocação da réplica, esta tarefa requer afinação e experiência. O objetivo é melhorar a disponibilidade, a fiabilidade dos dados e a utilização da largura de banda da rede.

Grandes instâncias do HDFS correm num *cluster* de computadores que normalmente se espalham ao longo de vários *racks*. A comunicação entre dois nós em *racks* diferentes tem de ser feita através de comutadores. Geralmente, a largura de banda da rede entre máquinas situadas no mesmo *rack* é maior do que entre máquinas em diferentes *racks*. Por sua vez, o *NameNode* determina o identificador do *rack* a que cada *DataNode* pertence.

Política simples

Colocar as réplicas *racks* exclusivas. Deste modo, previne-se a perda de dados quando um *rack* inteiro falha e permite usar a largura de banda de múltiplos *racks* aquando da leitura dos dados. Esta política de forma uniforme distribui as replicas no *cluster* o que torna mais fácil balancear o carregamento na falha de um componente. No entanto, esta metodologia aumenta o custo de escrita porque a escrita necessita de transferir blocos de múltiplos *racks*.

Outra política

Fator de replicação: 3 (caso comum). Colocação: 1 réplica num nó no *rack* local, 1 réplica num nó num *rack* diferente (remoto) e 1 replica num nó diferente no mesmo *rack* remoto. Esta política corta o tráfico de escrita *inter-rack* que normalmente melhora o desempenho da escrita. A hipótese de falhas do *rack* é bem menor do que a de um nó falhar: não tem impacto na garantia de fiabilidade e disponibilidade dos dados. Contudo, reduz a largura de banda da rede agregada usada aquando da leitura de dados quando um bloco é colocado em apenas 2 únicos *racks* em vez de 3. Desta forma os blocos não são colocados de forma uniforme pelos *racks*.

Política implementada

De forma resumida podem-se definir as seguintes características:

- um terço das réplicas está num nó;
- dois terços das replicas está num *rack*, e um terço uniformemente distribuídas pelos *racks* restantes.

Esta política melhora a performance da escrita sem comprometer a fiabilidade e a leitura dos dados.

Seleção da réplica

Quanto à seleção da réplica os objetivos essenciais que pretende atingir são:

- minimizar o consumo global de largura de banda;
- minimizar a latência de leitura.

Se existe uma réplica no mesmo *rack* que o nó leitor, então essa réplica é a preferida para satisfazer o pedido de leitura (tenta-se satisfazer o pedido de leitura de uma réplica que se encontre mais próxima do leitor). Se um *cluster* se espalha por múltiplos centros de dados, então a replica que é residente no centro de dados local é a pretendida em vez de qualquer uma réplica remota.

Modo seguro

Na inicialização, o *NameNode* entra no estado *Safemode*. A replicação dos blocos não ocorre neste estado. O *NameNode* recebe mensagens *HeartBeat* e *Blockreport* dos *DataNodes*. Cada bloco tem um número mínimo especificado de réplicas. Por sua vez, um bloco é considerado replicado quando o número mínimo de réplicas dos dados do bloco é verificado dentro do *NameNode*. Depois de ser verificada uma percentagem considerável de blocos replicados no *NameNode*, este sai do *Safemode*. Então é determinada a lista de blocos de dados que continuam a ter um número inferior ao número de réplicas especificado. O *NameNode* replica esses blocos para os *DataNodes*.

Persistência dos metadados do sistema de ficheiros

O *NameNode* usa um registo de transações (*EditLog*) para registar cada mudança que ocorra nos metadados do sistema de ficheiros. Um dos exemplos para ilustrar esta situação pode ser o seguinte: ao criar um novo ficheiro no HDFS, o *NameNode* insere um registo no *EditLog* indicando esta operação.

O *NameNode* usa um ficheiro no *host* local (*OS file system*) para guardar o *EditLog*.

Todo o sistema *namespace*, incluindo o mapeamento dos blocos dos ficheiros e as propriedades do sistema, é guardado num ficheiro (*FsImage*). Este (*FsImage*) também é guardado no sistema de ficheiros local.

O *NameNode* guarda uma imagem de todo o *namespace* do sistema de ficheiros e o ficheiro *Blockmap* na memória. Este item chave de metadados está desenhado para ser compacto, de tal modo que o *NameNode* com 4 gigabytes é capaz de suportar um grande número de ficheiros e diretórios. Quando o *NameNode* inicia, lê o *FsImage* e o *EditLog* do disco, aplica todas as transações contidas no *EditLog* para uma representação *in-memory* do *FsImage*, e limpa esta nova versão numa nova *FsImage* no disco. Depois pode truncar o velho *EditLog* porque as suas transações foram aplicadas ao *FsImage* persistente. Este processo é designado por *checkpoint*, e só ocorre quando o *NameNode* inicia.

O *DataNode* guarda os dados do HDFS em ficheiros no seu sistema de ficheiros local. O *DataNode* não tem conhecimento sobre os ficheiros do HDFS, guardando cada bloco de dados do mesmo num ficheiro separado no seu sistema de ficheiros local. Os ficheiros não são todos criados no mesmo diretório. Em vez disso, ele usa uma heurística para determinar o número ótimo de ficheiros por diretório e cria sub-diretórios. Não é viável colocar todos os ficheiros no mesmo diretório uma vez que o sistema de ficheiros local pode não ser capaz de suportar eficazmente um grande número de ficheiros num único diretório. Quando o *DataNode* inicia, ele explora o seu sistema de ficheiros local, gerando uma lista de todos os blocos de dados HDFS que correspondem a cada um desses ficheiros locais e envia o relatório ao *NameNode* (*Blockreport*).

Protocolos de comunicação

Os protocolos de comunicação utilizados são: TCP/IP. De uma forma simplificada, pode-se definir o processo de comunicação efetuado da seguinte forma: um cliente estabelece uma conexão com uma porta *TCP* configurável na máquina *NameNode*. Ele fala do Protocolo cliente com o *NameNode*. O *DataNode* fala com o *NameNode* usando o protocolo *DataNode*. Uma abstração RPC envolve o Protocolo cliente e o Protocolo DataNode. O *NameNode* nunca inicia qualquer RPC. Em vez disso, apenas responde aos pedidos do RPC emitidos por *DataNodes* ou clientes.

Robustez

O grande objetivo é guardar dados de forma confiável mesmo na presença de falhas. As falhas mais comuns são:

1. falhas do *NameNode*;
2. falhas do *DataNode*;
3. partições da rede.

Falha do disco de dados, *Heartbeats* e re-replicação

Cada *DataNode* envia uma mensagem *Heartbeat* ao *NameNode* periodicamente. Uma partição da rede pode causar um subconjunto de *DataNodes* e perder a conectividade com o *NameNode*. O *NameNode* deteta esta condição pela ausência da mensagem *Heartbeat*, marca os *DataNodes* sem *Heartbeats* recentes como mortos e não encaminha nenhum IO novo. A morte de *DataNodes* pode causar a falha do fator de replicação por ficar abaixo do valor especificado. O *NameNode* constantemente sabe que blocos precisam de ser replicados e inicia a replicação quando for necessário. A necessidade de re-replicação pode surgir por causa de:

1. O *DataNode* tornar-se indisponível;
2. A replica ficar corrompida;

3. O disco rígido da *DataNode* falhar;

4. O fator de replicação de um ficheiro pode aumentar.

Rebalanceamento do cluster

A arquitetura HDFS é compatível com os esquemas de rebalanceamento dos dados. Um esquema pode automaticamente mover dados de um *DataNode* para outro se o espaço livre no *DataNode* cair abaixo de um determinado valor. Num evento de alta procura repentina para um ficheiro particular, o esquema pode dinamicamente criar réplicas adicionais e rebalancear outros dados no *cluster*. Estes tipos de esquemas não estão implementados.

Integridade dos dados

É possível que um bloco de dados retirado de um *DataNode* chegue corrompido. Esta corrupção pode ocorrer devido a falhas no dispositivo de armazenamento ou de software com *bugs*. Os clientes implementam a verificação do *checksum* sobre os conteúdos dos ficheiros HDFS. Quando um cliente obtém um ficheiro HDFS, ele computa uma *checksum* de cada bloco do ficheiro e guarda esses *checksums* num ficheiro separado oculto no mesmo *namespace* HDFS. Quando o cliente obtém os conteúdos do ficheiro verifica se os dados recebidos de cada *DataNode* correspondem à *checksum* guardada no ficheiro associado. Se não, então o cliente pode optar por obter o bloco através de outro *DataNode* que contenha uma réplica desse bloco.

Falha do disco de metadados

O *FsImage* e o *Editlog* são estruturas de dados centrais no HDFS. A corrupção destes ficheiros pode tornar uma instância do HDFS não funcional. Por este motivo, o *NameNode* pode ser configurado para suportar a manutenção de múltiplas cópias. Qualquer *update* no *FsImage* ou *EditLog* causa um *update* síncrono em todas as réplicas. Este *update* das cópias pode degradar a taxa de transações do *namespace* por segundo que um *NameNode* pode suportar. Contudo, esta degradação é aceitável porque apesar das aplicações do HDFS serem muito intensivas com os dados, não o são com os metadados. Quando um *NameNode* reinicia, seleciona o último *FsImage* e *EditLog* para usar. Se um *NameNode* falhar, então é preciso a intervenção manual. O reiniciar automático e a recuperação após falha do *NameNode* noutra máquina, não é suportado.

Snapshots

O uso dos *snapshots* consiste em fazer *rollback* de uma instância do HDFS corrompida para uma instância conhecida como bom ponto no tempo. O HDFS não suporta esta funcionalidade.

Organização dos dados

Blocos de dados

O tamanho típico de um bloco do HDFS é 64 *megabytes*. Então o ficheiro HDFS é cortado em chunks de 64 MB e se possível cada *chunk* irá residir em *DataNodes* diferentes.

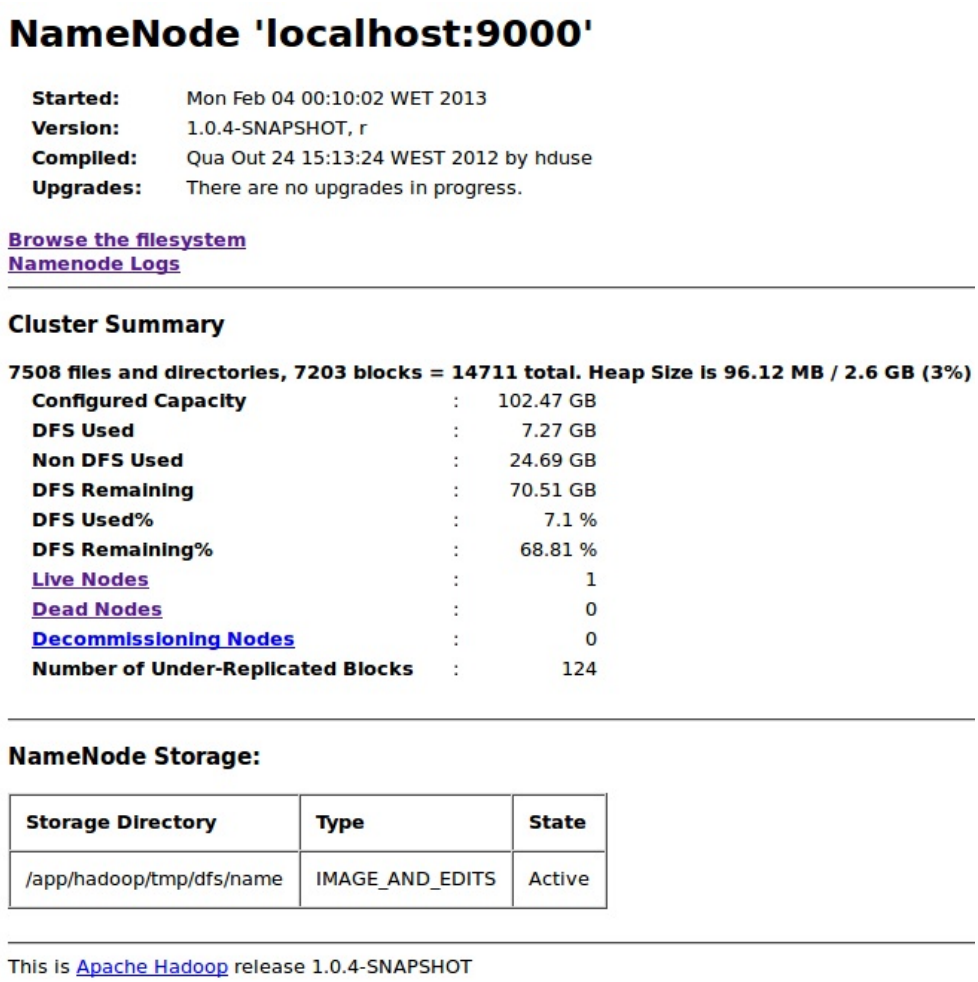
Encenação

Um cliente ao pedir a criação de um ficheiro não atinge o *NameNode* imediatamente. Inicialmente o cliente HDFS põe em *cache* os dados do ficheiro num ficheiro local temporário. A escrita da aplicação é redirecionada para este ficheiro local temporário e quando este acumula dados acima do tamanho de um bloco HDFS, o cliente contacta com o *NameNode*. O *NameNode* insere o nome do ficheiro na hierarquia do sistema de ficheiros e aloca um bloco de dados para o efeito. O *NameNode* responde ao pedido do cliente com a identidade do *DataNode* e o bloco de dados destino. Então o cliente limpa o bloco de dados do ficheiro local temporário para o *DataNode*. Quando um ficheiro é fechado, os dados não limpos restantes no ficheiro local temporário são transferidos para o *DataNode*. O cliente então diz ao *NameNode* que o ficheiro está fechado e nesse momento, o *NameNode* faz *commit* da operação de criação do ficheiro num armazenamento persistente. Se o *NameNode* falhar antes de o ficheiro ser fechado, o ficheiro é perdido.

Foi adotada esta política uma vez que se o cliente escrevesse diretamente para um ficheiro remoto sem nenhum *buffer* no lado do cliente, a velocidade da rede e o congestionamento na mesma afetaria a transferência consideravelmente.

Pipeling da replicação

O fator de replicação padrão é três. Quando o ficheiro local acumula um bloco cheio com dados do utilizador, o cliente obtém uma lista de *DataNodes* a partir do *NameNode*. Esta lista possui os *DataNodes* que conterão uma replica do bloco. Então o cliente limpa o bloco de dados para o primeiro *DataNode* e este começa a receber os dados em pequenas porções (4 KB), escreve cada porção para o seu repositório local e transfere essa porção para o segundo *DataNode* na lista. O segundo *DataNode* efetua as mesmas operações e passa a informação ao terceiro. Por fim, o terceiro escreve os dados no seu repositório. Então, um *DataNode* pode receber os dados de um anterior na *pipeline* e ao mesmo tempo enviar para o próximo.



NameNode 'localhost:9000'

Started: Mon Feb 04 00:10:02 WET 2013
Version: 1.0.4-SNAPSHOT, r
Compiled: Qua Out 24 15:13:24 WEST 2012 by hduse
Upgrades: There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

7508 files and directories, 7203 blocks = 14711 total. Heap Size is 96.12 MB / 2.6 GB (3%)

Configured Capacity	:	102.47 GB
DFS Used	:	7.27 GB
Non DFS Used	:	24.69 GB
DFS Remaining	:	70.51 GB
DFS Used%	:	7.1 %
DFS Remaining%	:	68.81 %
Live Nodes	:	1
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	124

NameNode Storage:

Storage Directory	Type	State
/app/hadoop/tmp/dfs/name	IMAGE_AND_EDITS	Active

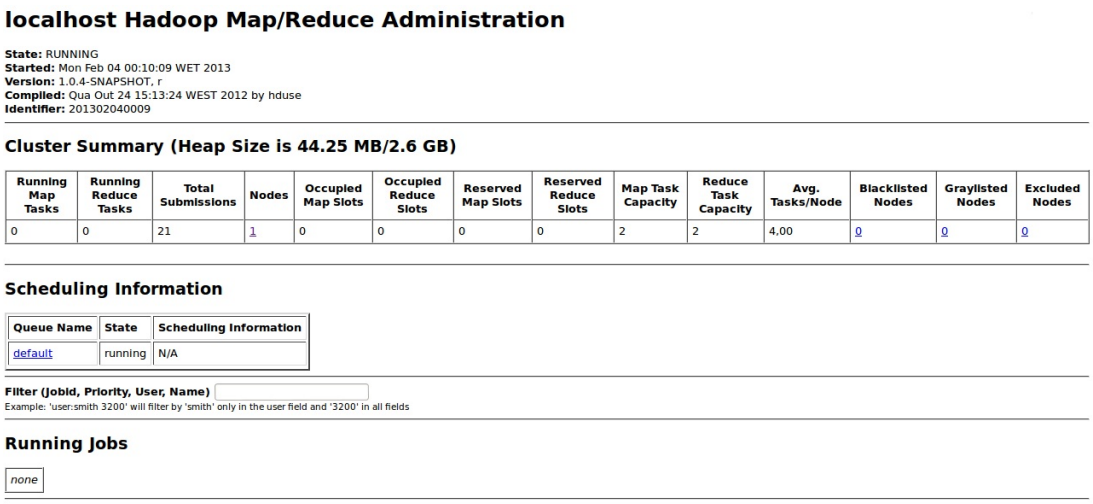
This is Apache Hadoop release 1.0.4-SNAPSHOT

Figura 2.4: Exemplo da interface web do NameNode

Para o utilizador poder coordenar todas as operações que estão a ocorrer no *Hadoop* são fornecidas algumas interfaces web relevantes:

- interface web *NameNode*(2.4): apresenta o sumário das informações do *cluster* como a informação sobre a capacidade total e restante, os nós no *cluster* ativos e inativos. Permite também navegar pelos *namespace* do HDFS e visualizar os conteúdos dos seus ficheiros a partir de um *web browser*. Adicionalmente permite também aceder aos ficheiros *log* existentes na máquina local, bastante úteis para tarefas de *debug*.

- interface web *JobTracker*(2.5): fornece informações sobre as estatísticas gerais dos *jobs* existentes no *cluster* do *Hadoop* (*jobs* completos, em execução ou falhados).Para além disso permite ver um ficheiro *log* histórico sobre o *job* submetido. Tal como no caso da interface do *Namenode*, também possibilita a consulta aos ficheiros de log do *Hadoop* na máquina onde esta interface estiver a correr.



localhost Hadoop Map/Reduce Administration

State: RUNNING
Started: Mon Feb 04 00:10:09 WET 2013
Version: 1.0.4-SNAPSHOT, r
Compiled: Qua Out 24 15:13:24 WEST 2012 by hduse
Identifier: 201302040009

Cluster Summary (Heap Size is 44.25 MB/2.6 GB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
0	0	21	1	0	0	0	0	2	2	4,00	0	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (jobId, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

Figura 2.5: Exemplo da interface web do JobTracker

- interface web *TaskTracker*(2.6): mostra as tarefas que estão a executar e as que não estão. Também permite o acesso ao ficheiros log da máquina local.

Através de Borkar et. al, em [BCL12], algumas vantagens conseguidas pelo *Hadoop*:

- *Open-source*;
- suporte para acesso a dados externos baseados em ficheiros;
- suporte automático e incremental de recuperação de *jobs* com *tasks* falhadas.
- facilidade em calendarizar *jobs* grandes em pequenos *chunks*;
- colocação automática de dados e rebalancemanto dos dados à medida que estes crescem e o número de máquinas aumenta ou diminui;
- suporte para a replicação e para a falha na máquina sem intervenção humana.

No entanto, no mesmo documento são referidas algumas contrapartidas que podem ser observadas na mesma plataforma e que deverão ser evitadas nas plataformas que lidam com este tipo de dados.

Algumas melhorias ao *Hadoop* foram já realizadas, como é caso apresentado em [NM10]. Neste trabalho implementou-se uma camada de armazenamento de dados optimizada ao nível da concorrência, baseada num serviço de gestão de dados *BlobSeer* que substitui a camada original existente no *HDFS*. Sendo assim, esta nova camada pretende possibilitar ao *Hadoop* a extensão das suas funcionalidades. Com os testes efetuados consegue-se observar que devido a esta camada BlobSeer o rendimento do *Hadoop* é melhorado em cenários que apresentam alta concorrência no acesso aos ficheiros partilhados.

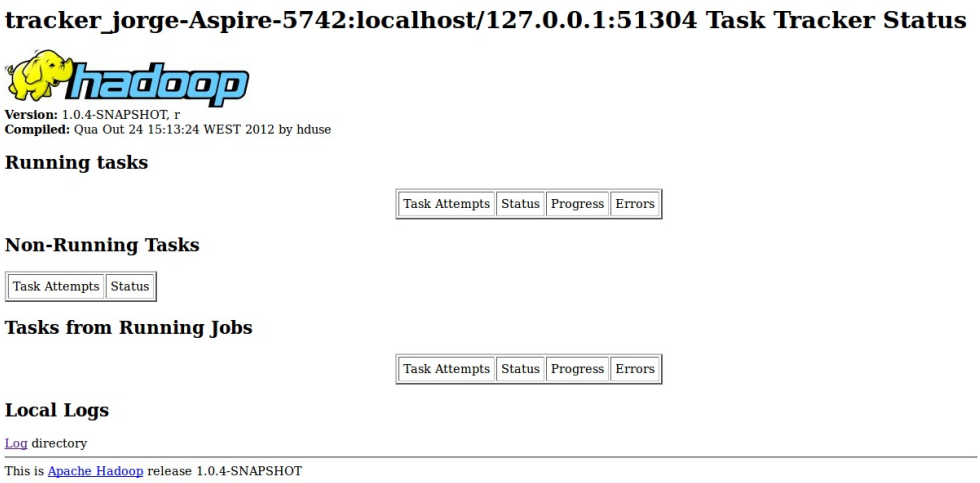


Figura 2.6: Exemplo da interface web do Tasktracker

2.3.2.2 *Nutch*

O *Nutch* é um projeto *open-source* da apache que permite pesquisar na web. É tida como uma ferramenta que implementa todas as funcionalidades principais de um motor de busca típico e permite uma fácil costumização devido à sua arquitetura modular. O sistema pode ser melhorado e estendido através de uma infraestrutura de *plugins* existente.

Os autores do estudo publicado em [MMD+07] que testaram esta ferramenta em termos de escalabilidade, concluíram que a mesma é altamente escalável, demonstrado através de várias configurações realizadas.

Num outro trabalho [KCSR04], os autores do mesmo pretenderam demonstrar como a arquitetura do *Nutch* pode ser flexível e escalável. Afirmaram que o *Nutch* é uma plataforma que pode ser usada na escala global, local e até pessoal e que o grande objetivo desta arquitetura é fornecer uma forma transparente para um motor de busca global.

2.3.2.3 *Solr*

O *Solr* é uma ferramenta de pesquisa construída a partir da plataforma apache *Lucene*. Encontra-se escrita em java e como principais funcionalidades caracteriza-se por fornecer uma API REST e permitir a pesquisa de texto livre. Para além disso, produz estatísticas sobre os dados indexados.

2.4 Conclusão

Após ter sido feito um estudo sobre a literatura existente nas áreas envolvidas no projeto, foram identificadas várias abordagens e métodos.

Ao nível do *web crawling*, pode-se verificar que existem diversos tipos de mecanismos que efetuam a busca e obtenção de páginas web. Dos tipos mencionados, o universal *crawler* devido à sua política, exige que sejam efetuados processos de *crawling* bastante eficientes e eficazes.

Constatando-se que é praticamente impossível obter todas as páginas da web, devido a imensidão de dados existentes, é necessário efetuar um bom *tradeoff* entre o armazenamento dos dados e a performance do *crawler*. É preciso otimizar os dados que são colocados em memória e os que são colocados em disco.

Quanto às abordagens ao *crawler* focado, todas elas pretendem vasculhar um determinado domínio de páginas e não toda a web. Foram detalhadas algumas abordagens que pretendem guiar o processo de *crawl* de uma ou de outra maneira. O modo como é guiado, varia de método para método, no entanto o grande objetivo destas técnicas é cobrir o maior número de páginas relevantes, não consumindo muitos recursos computacionais, uma vez que tem de prever as qualidades das páginas que pretendem explorar. Porém, os processos que utilizam taxonomias para a classificação das páginas, em certos domínios de atividade, podem não obter taxonomias com dados adequados e completos.

No que concerne à extração de dados, existem também numerosas técnicas, embora apresentem as suas especificidades.

No que se refere ao resumo de texto existem duas técnicas principais: a extração de passagens ou o texto integral.

Foram também fornecidas informações relativas ao contexto do *Big Data* e as vantagens que podem advir do tratamento de grandes quantidades de informação. Completa-se esta parte com a apresentação de algumas tecnologias existentes que já proporcionam o tratamento de *Big Data*. Descreveram-se algumas aplicações efetuadas com estas tecnologias.

Para finalizar deu-se a conhecer o paradigma *MapReduce*, com algumas referências à utilização do mesmo.

Capítulo 3

Plataforma de Extração de Dados

3.1 Introdução

Nesta secção pretende-se mostrar a estrutura da plataforma desenvolvida, bem como as explicações inerentes à descrição da mesma. Inicialmente será dada uma visão geral de todo o sistema e em seguida serão apresentadas com mais detalhe as operações efetuadas em cada parte específica da plataforma.

3.2 Arquitetura do Sistema

Como se pode verificar no sistema apresentado 3.1 foi instalada uma versão do *Nutch* (1.5.1) com o *Hadoop* (1.0.3). A indexação dos conteúdos pretendidos é feita através do versão 3.6.2 do *Solr*. As indicações sobre a instalação de todo o sistema podem ser encontradas no anexo C.

3.3 Fontes de informação

Como foi referido nos capítulos anteriores a fonte essencial de informação deste sistema são os fóruns de discussão. Assim, foi efetuado um estudo sobre a organização e as características de cada tipo de fórum fornecido pela empresa Ideia.m e sobre alguns existentes na web. Esse estudo pode ser lido na integra na secção B. Verificou-se que a organização dos mesmos pode variar muito nos diversos fóruns que foram analisados. Por exemplo, pode-se visualizar os seguintes códigos fonte de duas páginas com *posts* em dois fóruns diferentes. É possível ver nas imagens apresentadas 3.2 e 3.3, que a organização dos *posts* ao longo dos diversos domínios se releva bastante heterogénea.

No caso do *guitarnoise*, a informação necessária para ser extraída do *post* encontra-se dentro de uma divisão da página web onde o valor do atributo *class* é *postbody*. O nome do autor e o *timestamp* dos *post* podem ser obtidos a partir de um parágrafo, cujo atributo *class* se denomina *author* existente dentro desta secção. Por sua vez, o conteúdo do *post* pode ser obtido através

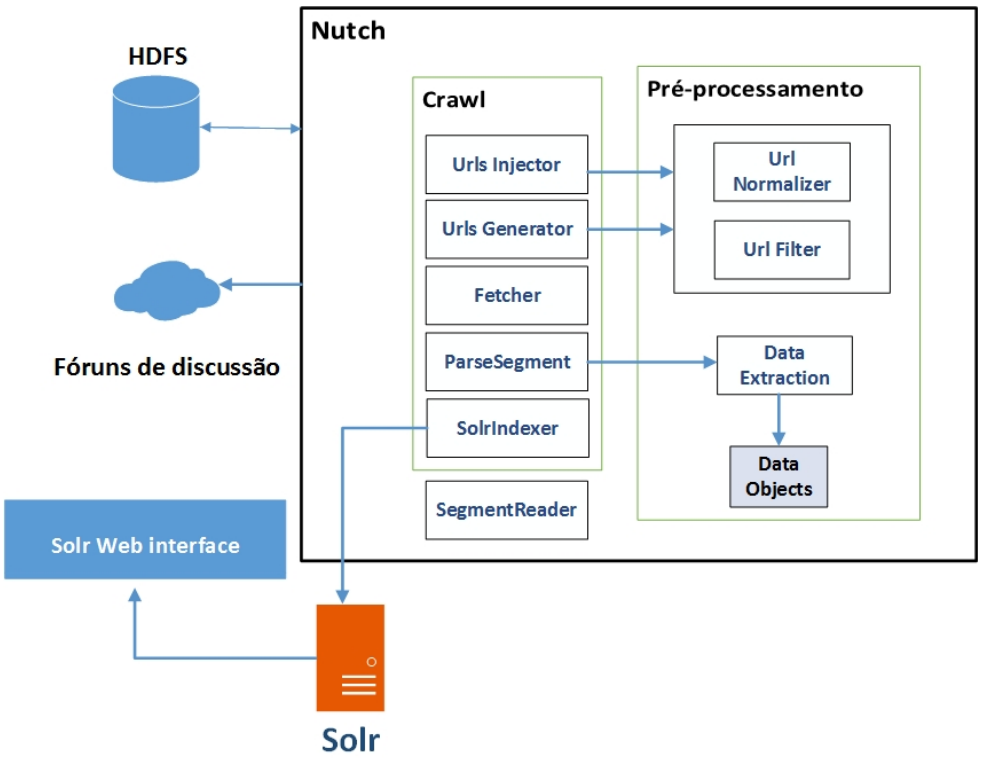


Figura 3.1: Arquitetura do Sistema

de uma outra divisão que se encontra inserida na divisão já mencionada e cujo atributo *class* se denomina *comment*.

No exemplo do *musicplayer* as informações são dispostas em tabelas. Pode-se verificar que a informação do autor é apresentada dentro de uma célula duma tabela HTML cujo atributo *class* é *author-content*. Mais especificamente no texto âncora dum link que permite visualizar as informações detalhadas do utilizador em questão. No que diz respeito ao conteúdo da mensagem, esta apresenta-se também ela dentro de uma célula da tabela mas com o atributo *class* designado por *post-content*. Dentro desta célula é ainda necessário percorrer duas divisões para se obter o conteúdo pretendido. Quanto ao *timestamp* do *post* verifica-se que apresenta um formato diferente do utilizado no *guitarnoise* e, embora a imagem não permita observar também se encontra inserido numa célula da tabela que constitui toda a informação do *post*.

Estes são dois exemplos que demonstram a grande heterogeneidade das páginas existentes nos diversos fóruns, complicando a extração dos seus dados por parte de um *crawler*.

3.4 *Crawler Nutch*

A pesquisa das páginas da web neste *crawler* é feita segundo o algoritmo primeiro em largura.

As estruturas de dados utilizadas pelo *Nutch*, que são escritas para o *Hadoop* e que são bastante úteis no processo de *crawling* são:

```
<div class="postbody">
    <h3 ><a href="#p469664">Re: SSG YR 11 WK 13 Empty Sky</a></h3>
    <p class="author">
        <a href="./viewtopic.php?p=469664#p469664">
            <img src="./styles/guitarnoise-2013-february/imageset/icon_topic_latest.png"
            width="11" height="9" alt="Post" title="Post" />
        </a>by
        <strong>
            <a href="./memberlist.php?mode=viewprofile&amp;u=26816">MrEWorm</a>
        </strong> &raquo; Thu Jan 31, 2013 6:36 am
    </p>
    <div class="content">Welcome to the forum.</div>
    <div id="sig469664" class="signature">Guitars, ukuleles and singing for
    fun<br />Ramps and StairLifts pay the bills.<br /><!-- w -->
    </div>
</div>
```

Figura 3.2: Código fonte de um post no fórum guitarnoise

- *Crawldb*: base de dados do *crawl*, que contém toda a informação sobre cada um dos URLs conhecidos pelo *Nutch*, incluindo se foi pesquisado e se foi, quando;
- *Linkdb*: base de dados dos *links*, contem a lista de links conhecidos de cada URL, incluindo o texto âncora e a fonte do URL;
- *Segments*: conjunto de segmentos que são conjunto de URLs pesquisados como uma unidade. Os segmentos são compostos por:
  - *crawl_generate*: nomeia um conjunto de URLs a serem pesquisados;
  - *crawl_fetch*: contém o estado de cada URL pesquisado;
  - *content*: contém o conteúdo em bruto retornado por cada URL;
  - *parse_text*: contém o texto de cada URL depois do parse;
  - *parse_data*: contém os *outlinks* e os metadados do parse efetuados a cada URL;
  - *crawl_parse*: contém os *outlink URLs*, usados para atualizar o *crawldb*.

A criação e a execução de algumas destas tarefas por parte do *Nutch* podem ser configuradas no *script* aquando da chamada do comando *crawl*.

Estes conteúdos resultaram dos diversos *jobs* efetuados e que foram submetidos ao *Hadoop*.

3.4.1 Processo de *Crawl*

O processo de *crawl* no *Nutch* pode ser efetuado de duas maneiras alternativas:

- através do comando *crawl*;
- através da execução de diversos comandos individuais, que compõe o comando *crawl*;

A abordagem utilizada neste projeto centrou-se na utilização do comando *crawl*. Este efetua várias operações: *injector*, *generator*, *fetcher* e *parsesegment*;

```
<span id="number2466550">#2466550</span> - <span class="date">01/29/13</span>
<span class="time">07:09 AM</span>
...
<td width="17%" valign="top" class="author-content">
<b><span id="menu_control_2466550"><a href="javascript:void(0);"
onclick="showHideMenu('menu_control_2466550','profile_popup_2466550');">Dpendery</a> </span></b>
<img src="/images/moods/default/offline.gif" alt="Offline" title="Offline" />
<span class="small">
Member
Registered:  07/06/11
Posts: 14
Loc:  Taipei, Taiwan
</span>
</td>
....
<td width="83%" class="post-content" valign="top">
<div class="post_inner">
<div id="body4">Above is all good advice.
I do know I need a better amp, yes.
That will come in time. Better delay is a very good idea, the delay on the Cube is basic.
I use a Telecaster by the way, if that helps.</div>
<div class="signature">
_____<br />
Long live rock.
</div>
</div>
</td>
```

Figura 3.3: Parte do código fonte de um post no fórum musicplayer

- *injector*: recebe um ficheiro com URLs e adiciona às páginas que necessitam de ser buscadas;
- *generator*: gera os *outlinks* para serem explorados.
- *fetcher*: robot do *Nutch*;
- *parsesegment*: efetua o parse dos conteúdos;

3.4.1.1 *Injector*

No início do processo de *crawl*, o *injector* é responsável por iniciar a *crawldb* com base nos *seed* URLs existentes num ficheiro de texto. O URL injetado corresponde à página inicial do fórum uma vez que, segundo [BPM], este URL permite aceder a páginas importantes, neste caso *threads* no fóruns de discussão.

A função *map* desenvolvida neste objeto pretende normalizar e filtrar os URLs injetados. São utilizados os filtros de normalização de URLs e de filtragem. A função *reduce* pretende recolher os resultados obtidos do *map* anterior para o URL.

3.4.1.2 *Generator*

Depois de terem sido colocados na *crawlDB* os *links* iniciais, o *generator* será responsável por expandir o processo de *crawl*. Numa primeira parte, executa uma operação de *map* e *reduce* em que verifica se os *outlinks* obtidos no *parse* anterior respeitam os filtros impostos pela expressão

regular definida para o *crawler*. Aqueles que respeitarem as condições impostas serão distribuídos por segmentos que serão lidos pelo *fetcher*. Numa segunda fase, é feita uma atualização na base de dados *crawlDB* para que uma nova execução do processo não gere os mesmos URLs.

No final deste processo será colocado no *Hadoop* a pasta *crawl_generate* com os resultados produzidos.

3.4.1.3 *Fetcher*

O *fecther* é responsável por obter as páginas web. Lê os URLs gerados na fase anterior e utiliza diversas *threads* sincronizadas para efetuar este processo. Os seus conteúdos são colocados numa pasta *crawl_fetch* e *content*.

3.4.1.4 *ParseSegment*

Nesta parte são efetuadas operações de *parse* dos dados. É utilizado o *plugin parse_HTML*, que sofreu algumas alterações, para recolher os dados relativos aos fóruns. Nesta fase as operações de *map* e *reduce* são responsáveis por ler o resultado obtido pelo *parser* e colecionar todas as estruturas obtidas. No final do processo a *crawldb* é atualizada com os novos URLs descobertos no *parse* das novas páginas obtidas pelo *fetcher*.

parse_text, *parse_data* e *crawl_parse*, são as pastas colocadas no *Hadoop* provenientes deste processo.

3.4.1.5 *SolrIndexer*

No final do processo de *crawl*, este objeto é responsável pela indexação e pela remoção de registos duplicados. Para que os conteúdos pretendidos fossem indexados foi necessário adicionar ao documento a ser enviado para o *Solr* os conteúdos obtidos na fase anterior. Internamente o *Nutch* cria um documento onde vão sendo adicionados os campos para serem lidos no esquema definido no *Solr*. Assim, foram acrescentados os campos referentes ao título da *thread*, e as informações dos *posts*, extraindo os conteúdos que provinham do *parse* dos segmentos.

3.4.2 *SegmentReader*

A leitura dos segmentos é feita pelo *SegmentReader* que efetua uma operação de *dump* para o *Hadoop* dos conteúdos obtidos. Esta operação lê todo o conteúdo binário existente nos diversos dados segmentados que foram criados nas etapas intermédias do *crawl*.

Este comando pode ser utilizado com as seguintes opções:

- *nocontent*: ignora o diretório do conteúdo;
- *nofetch*: ignora o diretório *crawl_fetch*;
- *nogenerate*: ignora o diretório *crawl_generate*;

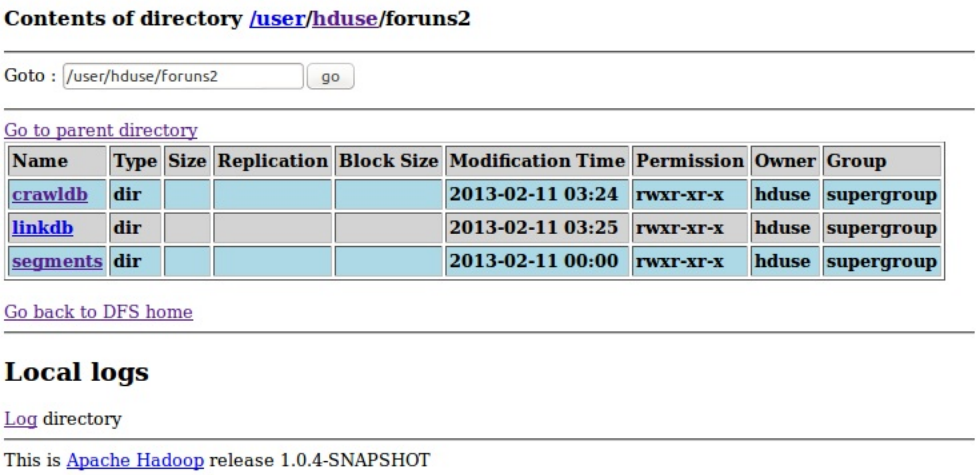


Figura 3.4: Pasta com os conteúdos do crawl

- *noparse*: ignora o diretório *crawl_parse*;
- *noparsedata*: ignora o diretório *parse_data*;
- *noparsetext*: ignora o diretório *parse_text*;

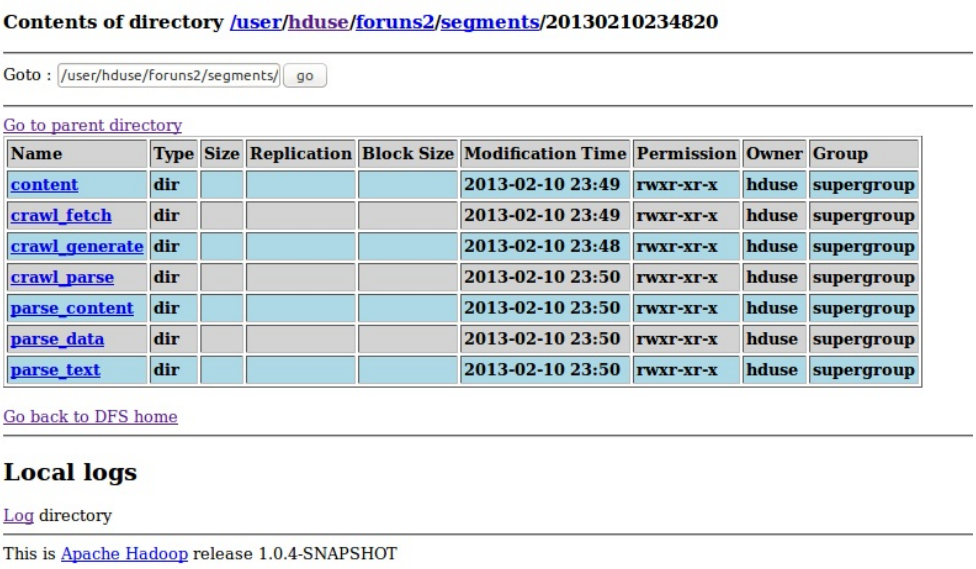
Neste caso optou-se por não ativar nenhuma das opções sendo visível todos os conteúdos intermédios.

Internamente esta funcionalidade cria uma *thread* para ler cada conjunto de segmentos referentes a cada secção. Para melhorar a eficiência e confiabilidade deste processo, os conteúdos dos fóruns foram colocados numa *thread* à parte para serem lidos. A principal utilidade desta operação foi essencialmente para *debug* do processo. Com esta operação é possível visualizar todos os URLs pesquisados, os *outlinks* obtidos a partir de cada página web, conteúdos da página entre outras.

3.5 HDFS

O HDFS tornou-se bastante importante neste projeto uma vez que permitiu armazenar as estruturas intermédias utilizadas pelo *crawler* bem como os resultados do processo de *crawl*. Como é concluído em [HN99] para se construir um *crawler* escalável é necessário saber onde colocar os dados produzidos do *crawl* uma vez que os dados manipulados por este são bastante grandes, sendo impraticável colocá-los em memória.

Como se pode verificar na figura 3.4 todos os dados resultantes do processo de crawl do *Nutch* são colocados em diversas secções. Por sua vez na secção dos segmentos, cada um deles possui os diversos segmentos que foram criados ao longo dos diversos *jobs*. No final efetua-se o *dump* destes segmentos sendo possível observar toda a informação contida neles e que refletem os dados extraídos das páginas web.



Contents of directory /user/hduse/foruns2/segments/20130210234820

Goto : /user/hduse/foruns2/segments/ go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
content	dir				2013-02-10 23:49	rwxr-xr-x	hduse	supergroup
crawl_fetch	dir				2013-02-10 23:49	rwxr-xr-x	hduse	supergroup
crawl_generate	dir				2013-02-10 23:48	rwxr-xr-x	hduse	supergroup
crawl_parse	dir				2013-02-10 23:50	rwxr-xr-x	hduse	supergroup
parse_content	dir				2013-02-10 23:50	rwxr-xr-x	hduse	supergroup
parse_data	dir				2013-02-10 23:50	rwxr-xr-x	hduse	supergroup
parse_text	dir				2013-02-10 23:50	rwxr-xr-x	hduse	supergroup

Go back to DFS home

Local logs

Log directory

This is Apache Hadoop release 1.0.4-SNAPSHOT

Figura 3.5: Conteúdos do segmento

3.6 Interligação entre *Hadoop e Nutch*

Para se efetuar a interligação entre estas duas ferramentas foi necessário replicar os ficheiros de configuração do *Hadoop* para a localização dos ficheiros de configuração do *Nutch*.

Foi utilizado um *bash script* que permitia conduzir todo o processo descrito. O código fonte referente a este *script* pode ser observado no anexo D. Este *script* inicialmente efetua algumas verificações obre a localização das aplicações e termina esta fase inicial com o processo de *crawl* executando o objeto com o mesmo nome, construído no *Nutch*, e que vai ser lido pela consola do *Hadoop*.

Objeto Crawl [1]

- -dir": indica o nome do diretório onde o *crawler* irá colocar os resultados do processo.
- -threads": número de *threads* que irão correr em paralelo para ir buscar a páginas web;
- -depth": indica até que profundidade o *crawler* deve ser feito. Por outras palavras, define quantas vezes o ciclo de operações (*generator*, *fetcher* e *parsesegment*) deve ser feito.
- -topN N": determina o número máximo (N) de *outlinks* que o *crawler* irá extrair de cada página explorada.

Se não for ativa qualquer uma das opções o *Nutch* irá colocar os valores padrão.

Os diversos testes que foram efetuados incidiram sobre a variação do parâmetro *depth*.

Um outro atributo utilizado aquando da indexação foi o *-solr* indicando a localização do servidor *solr*.

[1] http://wiki.apache.org/nutch/bin/nutch_crawl

No final do *script* é feita a leitura dos segmentos através do objeto *SegmentReader* também ele construído pelo *Nutch*.

3.7 Pré-Processamento

Ao longo do processo de *crawl* são executadas diversas tarefas de pré-processamento dos dados. Esta fase é importante para a limpeza dos dados e para a fonte de dados utilizada a sua importância é maior devido ao facto de ser uma fonte bastante ruidosa.

3.7.1 *URL Filter*

Para limitar as páginas que eram expandidas pelo *crawler* foram definidas expressões regulares para limitar o *crawler*, usando o *plugin regex-urlfilter*. O *Nutch* fornece ainda mais alguns filtros como o *prefix* e *suffix* para melhor definir quais os prefixos e sufixos que as páginas a percorrer devem ou não conter. Neste caso, só foi necessário definir o *plugin suffix* uma vez que os efeitos do *preffix* foram englobados no *regex-urlfilter*.

As expressões regulares definidas foram as seguintes:

- (.)+(feed|index|faq|search|ucp|memberlist|style|posting)(.)*, com esta expressão pretende-se que o *fetcher* não obtenha as páginas onde estas palavras se encontrem expressas no URL.
- $\hat{}$((http)s?)://([a-z0-9]+[]*)+/foruns/.*), indica que os URLs explorados se encontram no domínio do fórum.

Estas expressões foram submetidas no ficheiro *regex-urlfilter.txt* que irá ser utilizado pelo *plugin* com o mesmo nome e foi necessário efetuar algumas modificações ao ficheiro original uma vez que algumas expressões padrão já definidas impediam o funcionamento normal do *crawler* que se pretendia implementar.

3.7.2 *URL Normalizer*

Foi ativada a funcionalidade desenvolvida pela comunidade *Nutch*. Não foram efetuadas alterações mas foi usada. Com esta funcionalidade pretende-se que os URLs sejam normalizados, ou seja, que os URLs pesquisados sejam reduzidos à sua forma canonical. Como acontece no caso anterior, existe um ficheiro de texto para efetuar as operações pretendidas neste campo (denominado neste caso por *regex-normalize*).

3.7.3 Extração de Dados

Ao nível da extração foi efetuada uma primeira abordagem que não foi conseguida. Tentou-se filtrar o conteúdo do *parse* com base na aplicação de filtros nos conteúdos que eram obtidos do *crawl*.

O *Nutch* armazena todo o conteúdo resultante dos diversos *parsers* implementados por *plugins*, num objeto designado por *ParseResult*. Por sua vez, este objeto é submetido a uma série de filtros que o utilizador pode desenvolver, com a adição de *plugins*, para filtrar os conteúdos que pretende com base nesses resultados.

Deste modo, desenvolveu-se um novo *plugin* em que a sua única função seria a aplicação de uma expressão regular ao conteúdo da página obtida. Assim, toda a informação que respeitasse a expressão seria extraída. Ao se ativar este *plugin* na infraestrutura de *crawl* verificou-se que esta metodologia impedia o funcionamento normal do sistema, mais especificamente na geração de links a explorar optando-se por outra abordagem.

Por outro lado, como o *Nutch* fornece um *parse_HTML* necessário para efetuar a expansão do *crawler* foi estendida a API, desenvolvendo-se funcionalidades para extrair as informações dos *posts*. Para efetuar esta extração foi necessário visualizar o código fonte das páginas web e verificar em que *tags* estavam a ser mostrados os conteúdos pretendidos. Como tal, foi usada uma abordagem baseada na árvore das páginas web que eram vasculhadas pelo *Nutch*. Esta plataforma gera uma *DOM tree* para cada página web a partir da qual o sistema inicialmente retira os metadados e os *outlinks* da mesma para suportar o sistema de *crawl*. Deste modo, foram também desenvolvidas funções percorrendo a árvore gerada pelo *Nutch* para cada página e extraído o conteúdo pretendido para um modelo de dados inserido na plataforma.

Sobre o modelo de dados utilizado para armazenar a informação obtida serão dadas mais explicações na próxima sub-secção.

No caso de estudo utilizado, verificou-se que nas páginas onde se apresentavam os *posts* de uma *thread*, o nome da mesma encontrava-se no conteúdo da única *tag* h2 existente na página. Este elemento HTML tem sempre inserido um link cujo texto âncora fornece o título da *thread*.

Verificou-se que as informações de cada *post* se encontravam dentro de uma secção denominada por *postbody*. Toda a informação essencial do *post* se encontra encerrada nessa secção, como pode ser visto no exemplo fornecido 3.6.

Após a visualização do conteúdo englobado por esta divisão, observou-se que existiam dois blocos essenciais para retirar a restante informação, a secção referente ao autor do *post* e a do comentário produzido.

Procedeu-se então à exploração das sub-árvores correspondente a cada uma destas sub-secções. No que diz respeito à do autor, é possível obter o *timestamp* através do conteúdo da *class author* definida numa *tag* p.

Por outro lado, o nome do autor é determinado através da análise do texto âncora do *link* que efetua a ligação para o perfil do mesmo.

A realização de testes de validação permitiu descobrir que nem todos os autores eram encontrados com esta estratégia. Verificou-se ainda que, todos os *posts* em que este tipo de autores apareciam o *timestamp* da mensagem não era revelado. Conclui-se que existiam autores a colocar conteúdos no fórum sem se encontraram registados na comunidade, isto é, sem perfil definido. Nestes casos, o fórum só regista o nome do autor e *timestamp* do *post*. Assim, sempre que o nome

```
<div class="postbody">

        <h3 ><a href="#p462512">Re: How's my singing?</a></h3>
        <p class="author"><a href="./viewtopic.php?p=462512#p462512"><img src="./styles/guitarnoise-2012-
c/imageset/icon_post_target.gif" width="11" height="9" alt="Post" title="Post" /></a>by <strong><a href="./memberlist.php?
mode=viewprofile&amp;u=29632">EzraplaysEzra</a></strong> &raquo; Tue May 08, 2012 12:26 pm </p>

        <div class="content">I'm no expert, but I bet you could sing pretty well if you studied up and worked at
it.</div>

</div>

        <dl class="postprofile" id="profile462512">
        <dt>
                <a href="./memberlist.php?mode=viewprofile&amp;u=29632">EzraplaysEzra</a>
        </dt>

        <dd>Full Member</dd>

<dd> </dd>

<dd><strong>Posts:</strong> 392</dd><dd><strong>Joined:</strong> Fri Apr 13, 2012 6:07 pm</dd>

</dl>
```

Figura 3.6: Exemplo do código fonte de uma página do *guitarnoise*

do autor não for encontrado no texto âncora obtém-se através do conteúdo *tag strong* indiciando que o utilizador não possui um perfil definido.

É importante salientar que o nome do autor poderia ser obtido por meio de uma outra secção fora do bloco *postbody*, que contém as informações do perfil do autor que proferiu determinado *post*. Não se optou por explorar esta abordagem uma vez que nesta secção não é possível obter o *timestamp* do *post*. O *timestamp* apresentado diz respeito à data em que o utilizador entrou na comunidade, sendo sempre necessário percorrer a sub-árvore referente à secção autor para obter o *timestamp* pretendido.

Relativamente à secção do comentário facilmente se obtém o conteúdo através da *tag div* com atributo *class* designado por *content*. No entanto, a variedade de informação existente nesta divisão da página obriga a realização de certas tarefas de pré-processamento, nomeadamente:

- tratamento de links: anotação dos texto âncora de links existentes no conteúdo do *post*;
- tratamento de citações: anotação do conteúdo da citação através da colocação do caracter "no início e no fim da citação.
- tratamento das *tags* br: substituição desta *tag* pelo caracter newline;
- remoção de certos caracteres: \r, \n.

Figura 3.7: Esquema de Classes

- substituição de certas expressões: por um espaço em branco.

Ao nível do conteúdo do *post*, são escritos todos os elementos de texto exceto aqueles encerrados em links e citações. Nesse caso, sempre que existe uma *tag* a no comentário é extraído o *link* através do conteúdo do atributo href e o texto âncora através do nó de texto que é filho deste nó. Ao nível das citações, o conteúdo é extraído através da identificação da *tag blockquote* que possui como nó filho uma *tag* div. Dentro desta divisão, pode existir um nó cite onde se pode extrair quem proferiu a citação. Caso não exista, verifica-se que é uma citação cujo autor se desconhece e tal como na primeira situação, os outros nós filhos desta secção contém o conteúdo da citação. Este texto inserido na *quote* é submetido também às outras tarefas de pré-processamento existente no corpo principal da mensagem.

3.8 Modelo de Dados

A imagem 3.7 apresenta o modelo de dados que foi necessário criar para adicionar ao modelo já existente no *Nutch*. A inclusão deste modelo de dados foi efetuada no objeto, *ParseImpl* utilizado pelo *Nutch* para recolher os dados do *parse*. Este objeto implementa a interface *Parse*.

O *ParseImpl* recebe o objeto *ParseContent* pelo construtor alterado e que é aplicado na função map no *parsesegement*. Aí o construtor recebe um objeto *Parse* através das entradas *ParseResult* que são obtidas através do *ParseUtil* que por sua vez possui um *ArrayList* dos *parsers*. O *ParseResult* é construído no *plugin* HTML, onde são extraídos os dados do fórum, e encapsulado no *ParseImpl*.

Para tal, foi construído o objeto *Post* com a finalidade de albergar o conteúdo necessário como: autor, o tempo em que o *post* foi publicado e o conteúdo do mesmo. Também foi necessário anotar o nome das *thread* que contêm estes *posts*. Estes conteúdos armazenados, foram definidos sabendo que são transversais a todos os tipos de fóruns e que numa posterior adição de novos fóruns, o modelo de dados será compatível com eles.

3.9 Interligação *Nutch Solr*

De modo a ser possível pesquisar as informações recolhidas pelo *crawl*, foi efetuada esta ligação entre as duas plataformas. Colocou-se a correr o servidor *solr* com um esquema definido

```
<!-- core fields -->
<field name="segment" type="string" stored="true" indexed="false"/>
<field name="digest" type="string" stored="true" indexed="false"/>
<field name="boost" type="float" stored="true" indexed="false"/>
<!-- fields for index-basic plugin -->
<field name="host" type="string" stored="true" indexed="true"/>
<field name="url" type="url" stored="true" indexed="true"
    required="true"/>
<field name="content" type="text" stored="true" indexed="true"/>
<field name="title" type="text" stored="true" indexed="true"/>
<field name="thread_title" type="text" stored="true" indexed="true"/>
<field name="author" type="string" stored="true" indexed="true" multiValued="true"/>
<field name="comment" type="string" stored="true" indexed="true" multiValued="true"/>
<field name="time" type="string" stored="true" indexed="true" multiValued="true"/>
```

Figura 3.8: Alguns campos do esquema existente no *Solr*

de acordo com os campos que deveriam ser apresentados na interface web disponibilizada pelo mesmo. Esse mesmo foi replicado para os ficheiros de configuração do *Nutch*.

3.10 Servidor *Solr*

Para indexar os dados no *Solr* foi importante adicionar os campos necessários ao esquema do *Solr*. A imagem 3.8 mostra os campos que são relevantes para a análise dos conteúdos.

No final do processo, é possível ver os resultados obtidos com consultas efetuadas no *Solr*. Para exemplo foi efetuada uma consulta com os seguintes parâmetros:

- q=*author*
- fl = *author*, *time*, *thread_title*, *comment*
- valor= VicLewisVL
- *rows*= 100

Com esta *query*, pretendiam-se obter os documentos que contivessem como autor de um dos *posts* que compõe a *thread* representada no documento o *VicLewisVL*. Para além disso, só se pretendia que os campos *author*, *time*, *thread_title* e *comment* fossem retornados. Por fim, com a opção *rows* pretende-se visualizar os 100 primeiros documentos resultantes da pesquisa. Assim, um dos resultados dessa *query* pode ser vista na figura 3.9.

Na interface web disponibilizada pelo servidor é possível efetuar diversas *queries* aos dados recolhidos pelo sistema de *crawl*, segundo os campos definidos no *schema.xml* existente no *solr*. Neste mesmo documento foi definido como campo padrão para a pesquisa o *content* que possui o conteúdo de toda a página HTML sem as *tags* correspondentes.

```
-<doc>
 -<arr name="author">
    <str>MelodyMaker</str>
    <str>Vic Lewis VL</str>
  </arr>
 -<arr name="comment">
  -<str>
    Hey everyone! Wow, there are so many amazing songwriters in here! I'm pretty new to songwriting/lyric writing and was
    wondering if anyone out there would know of any great books that would help me progress in my songwriting career? I'm a
    huge fan of Taylor Swift, Miranda Lambert, and Carrie Underwood if that helps? Just need a book that would help me have a
    broader understanding of songwriting and creating hit songs. If you come across anything let me know, and I'll do the same!
    Thanks! -Melody
   </str>
  -<str>
    The Complete Idiot's guide to the art of songwriting, co-written by our very own David Hodge - yes, the same David Hodge
    who's currently setting the assignments/topics in the Sunday Songwriters Group forum. Drop in there, MelodyMaker... all new
    writers are welcome, and encouraged. Vic
   </str>
  </arr>
  <str name="thread_title">Songwriting Book?</str>
 -<arr name="time">
    <str>Thu Aug 09, 2012 8:36 am</str>
    <str>Thu Aug 09, 2012 4:42 pm</str>
  </arr>
</doc>
```


Figura 3.9: Exemplo de um dos documentos retornados segundo a *query* efetuada

3.11 Conclusão

Neste capítulo, foi dada uma visão de todo o sistema explicitando cada uma das componentes existentes no mesmo. Para a realização desta componente do trabalho, foi necessário estudar grande parte do código da ferramenta do *Nutch*. Registou-se alguma dificuldade na introdução dos conteúdos do *parse* no processo de *crawl* pois a primeira alternativa tornou-se inviável, uma vez que era bastante inflexível com a utilização de expressões regulares.

Capítulo 4

Avaliação

4.1 Introdução

Nesta secção pretende-se apresentar a avaliação efetuada ao sistema desenvolvido. Para tal, será dada a conhecer a metodologia utilizada nesta área e os testes realizados. Com vista a avaliar a ferramenta foi desenhado um conjunto de testes, a vários níveis, tendo por base a recolha de informação de um grupo de discussão designado por *guitarnoise*. [1]. As figuras 4.1 e 4.2 pretendem ilustrar o formato do fórum utilizado para a avaliação.

Todos os testes efetuados foram executados numa máquina com as seguintes características:

- Memória RAM: 4 GB
- Processador: Intel core i5-460M
- Versão Java: OpenJDK 1.6.0_24 64 bits

Nesta mesma máquina, efetuou-se uma pseudo-distribuição com um *datanode* e um *master-node*.

4.2 Teste de desempenho

4.2.1 Metodologia de Avaliação

Foram efetuados três testes um com profundidade quatro, outro com três e outro com profundidade dois ao nível do *crawl*. Para se efetuarem estes testes foi necessário definir uma memória do *heap* da consola do *Hadoop* de 2000MB. Para se dar início ao teste colocou-se a correr o *Hadoop* e o servidor *Solr*. De seguida definiu-se como *depth* no parâmetro de *crawl* a profundidade que se pretendia. Correu-se o *script continuouCrawl.sh*. O tempo foi cronometrado desde o início da execução do *script* até ao final da execução do mesmo e foi obtido através dos *prints* que são

[1] http://www.guitarnoise.com/forums/

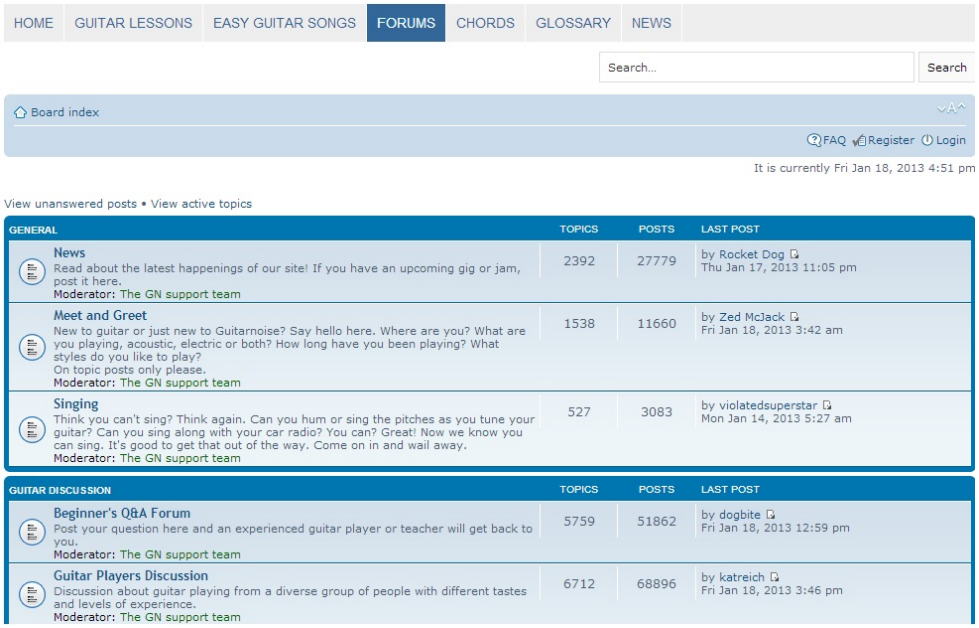


Figura 4.1: Página principal do Fórum *guitarnoise*

efetuados ao longo dos diversos *jobs* que mostram um rasto da execução. Efetuou-se o teste com um número máximo de *threads* para o *fetcher* igual a 20 para o processo de obtenção das páginas.

4.2.2 Resultados

Como pode ser observado na tabela 4.1 existe uma grande discrepância entre os tempos observados entre as profundidades. Esta situação deve-se ao facto de que as expressões regulares definidas para limitar o *crawler* incidem essencialmente para delimitar a árvore de pesquisa das páginas relativamente ao conteúdo que não é interessante. Desse modo, com profundidade 2 são selecionados as ligações que permitem o acesso aos conteúdos dos fóruns. A esta profundidade existem muitos links que não são relevantes e portanto são rejeitados. Ao serem efetuadas experiências com as profundidades três e quatro, verifica-se que o tempo aumenta substancialmente devido ao facto de que nestes casos as ligações relevantes são em maior número logo o número de URLs a explorar é maior.

Tabela 4.1: Resultados do teste de performance

Profundidade	Tempo
2	18 minutos
3	225 minutos
4	1740 minutos

Figura 4.2: Página com a exploração de uma *thread* do fórum *guitarnoise*

4.3 Teste de extração de Dados

4.3.1 Metodologia de Avaliação

Com vista à verificação dos dados extraídos, foi criado um *golden set* com as diversas páginas de tópicos existentes neste fórum, de forma aleatória e que não foram utilizadas para treinar o sistema.

Para se correr o teste colocaram-se como páginas *seed* os URLs das páginas que se queriam testar,figura (4.3), e definiu-se a profundidade do *crawl* a um pelo que, os *outlinks* destas páginas não serão explorados.

Os resultados obtidos através da indexação foram posteriormente extraídos para um ficheiro csv e colocados num folha de cálculo. Esta informação permitiu avaliar manualmente os campos referentes ao autor, tempo do *post* e título da *thread* com o intuito de verificar se os conteúdos foram corretamente extraídos.

Foram produzidas algumas estatísticas sobre estes dados. A saber:

- número total de *threads*: 19;
- número total de *posts*: 118;
- número médio de *posts* por *thread*: 6;

No caso do conteúdo do comentário, foi computada a distância *Levenshtein* entre duas *strings*. Para auxiliar a análise dos resultados foi criada uma folha de cálculo no *excel*, onde foi implementada uma macro em *Visual Basic*, com o algoritmo descrito acima, para a comparação entre duas

http://www.guitarnoise.com/forums/viewtopic.php?f=10&t=46072
http://www.guitarnoise.com/forums/viewtopic.php?f=10&t=4067
http://www.guitarnoise.com/forums/viewtopic.php?f=6&t=7020
http://www.guitarnoise.com/forums/viewtopic.php?f=6&t=52729
http://www.guitarnoise.com/forums/viewtopic.php?f=6&t=48102
http://www.guitarnoise.com/forums/viewtopic.php?f=6&t=52701
http://www.guitarnoise.com/forums/viewtopic.php?f=6&t=52701&start=15
http://www.guitarnoise.com/forums/viewtopic.php?f=6&t=52372
http://www.guitarnoise.com/forums/viewtopic.php?f=6&t=50701
http://www.guitarnoise.com/forums/viewtopic.php?f=6&t=36238
http://www.guitarnoise.com/forums/viewtopic.php?f=6&t=35122
http://www.guitarnoise.com/forums/viewtopic.php?f=6&t=36179
http://www.guitarnoise.com/forums/viewtopic.php?f=22&t=52753
http://www.guitarnoise.com/forums/viewtopic.php?f=33&t=53030
http://www.guitarnoise.com/forums/viewtopic.php?f=22&t=53479
http://www.guitarnoise.com/forums/viewtopic.php?f=22&t=52595
http://www.guitarnoise.com/forums/viewtopic.php?f=22&t=54015
http://www.guitarnoise.com/forums/viewtopic.php?f=22&t=52373
http://www.guitarnoise.com/forums/viewtopic.php?f=22&t=48701
http://www.guitarnoise.com/forums/viewtopic.php?f=3&t=54049

Figura 4.3: Lista de URLs utilizados para efetuar o teste de extração

strings. Retiraram-se os conteúdos dos *posts* a partir das páginas alocadas no fórum e efetuou-se o pré-processamento de acordo com as tarefas que foram realizadas no projeto. Deste modo, conseguiram-se recolher os dados que era previsto obter. De seguida, foram retirados dos resultados extraídos pelo sistema de recolha de dados, os comentários dos *posts* e feita a comparação com base na distância entre as mesmas.

4.3.2 Resultados

	Número de Autores	Número de *Timestamp*	Número de Títulos das *Threads*
Previsto	118	118	19
Obtido	118	118	19

Os resultados foram bastante satisfatórios. Todos os autores, *timestamp* de *posts* e títulos de *threads* foram obtidos de forma correta a partir da amostra retirada.

Quanto ao conteúdo dos *posts*, verificou-se que o conteúdo indexado era bastante semelhante ao que se esperava. Contudo, pretende-se efetuar mais testes e melhorar a anotação das citações. Observou-se também que poderia ser feita a anotação do texto inserido em imagens e efetuado um melhor pré-processamento dos conteúdos em tabelas.

4.4 Teste de cobertura

Ao longo da plataforma foram executados diversos testes de validação do software. Numa fase intermédia do desenvolvimento, estando concluída a parte referente ao processo de *crawl*, foi necessário analisar as páginas obtidas pelo *crawler*. Este tipo de teste pretende verificar qual o nível de cobertura que o *crawl* consegue obter relativamente ao acesso a páginas relevantes.

4.4.1 Metodologia de teste

Este teste foi desenhado de forma a ser efetuado um *crawl* a partir da *seed page* com profundidade 2. Para efetuar a análise dos resultados do teste, recorreu-se à funcionalidade do *Nutch*, de modo a ler os segmentos gerados pelo *crawl* e escritos no *Hadoop*. Estando escrito no *Hadoop*, todas as páginas vasculhadas e os links pesquisados, foi efetuado o *download* do ficheiro com essa informação para o sistema de ficheiros local. Obtida essa informação, analisou-se o ficheiro com base nos URLs que se esperavam obter e observou-se se os URLs pretendidos foram atingidos.

Também foi efetuado um teste com profundidade 3, mas devido ao elevado número de páginas a avaliar manualmente não foi possível tirar ilações conclusivas.

Numa fase mais avançada do projeto e depois de se ter feito a interligação com o servidor *Solr* voltou-se a efetuar o mesmo teste mas recorrendo às potencialidades do mesmo e acrescentando um grau de profundidade no teste.

4.4.2 Resultados

Com a realização do primeiro teste para a profundidade 2 constatou-se que todos os URLs das páginas pretendidas com essa profundidade foram exploradas. Os *links* obtidos diziam respeito a páginas que se encontravam no domínio do fórum. Para além disso, como o fórum se encontra embutido numa página web com outras funcionalidades observou-se que nenhum dos links explorados se encontravam nessas áreas do site principal. Nenhuma página referente à administração do fórum (como *login*, *faqs*, postagem de conteúdos, entre outras) foi explorada. No entanto, como a página principal do fórum efetua a listagem dos conteúdos através de secções temáticas, os URLs correspondentes também foram percorridos. Apesar deste pequena imperfeição do sistema, a performance do mesmo não fica comprometida uma vez que o número de URLs a explorar nestas circunstâncias é pequeno e os *outlinks* destas páginas são as próprias páginas de listagem contidas na secção.

Como se pode observar na figura 4.4, com profundidade quatro já se consegue obter um número considerável de *posts* e *threads* de todo o fórum.

Contudo, dos documentos indexados, alguns deles mostraram que algumas páginas acedem a conteúdos não pretendidos, como por exemplo a versão de impressão do *post*. Por outro lado, também foram indexadas páginas que não continham *posts* mas forneciam informação do fórum, como por exemplo páginas com listagem dos *posts* e a página principal.

Profundidade	Documentos Indexados	Nº Threads	Nº Posts
2	62	22	105
3	2181	866	6859
4	18076	4554	34700

Figura 4.4: Resultados obtidos para as diversas profundidades

4.5 Conclusão

Terminada a fase de testes, verificou-se que o sistema extrai e indexa informações do fórum de discussão, no entanto poder-se-iam efetuar mais testes para confirmar a validade do sistema.

Capítulo 5

Conclusão

5.1 Trabalho Futuro

Como trabalho futuro pretende-se efetuar mais alguns testes, bem como aperfeiçoar algumas etapas de todo o processo. Poderão também ser usados alguns casos de estudos já estudados para a aplicação desta plataforma e verificar a utilidade do mesmo para as empresas em questão.

Assim, algumas empresas existentes no mercado podem usufruir deste tipo de serviços para conseguirem obter informações relevantes. De seguida, apresentam-se dois casos de aplicação do sistema em empresas portuguesas.

No primeiro caso, trata-se de uma empresa alimentar portuguesa de grande dimensão que se encontra no setor agroalimentar e se apresenta como líder no mercado português. Ao nível das atividades caracterizam-se por efetuar a produção de alimentos, como pastas, pizzas, refeições pré-cozinhadas e biscoitos. Para além disso, efetuam moagem, comércio de cereais e derivados. As grandes vantagens que a aplicação de um sistema deste tipo traria a esta empresa seriam, a obtenção de conhecimento relativo ao modo como os consumidores usam os seus produtos e o conhecimento de quais as metodologias utilizadas para a preparação dos mesmos.

O segundo caso é a empresa denominada por Ideia.m. Neste caso, trata-se de uma empresa portuguesa de pequena dimensão inovadora na área do design, engenharia e materiais compositos.

O processo de trabalho desta empresa pode ser descrito com o fornecimento a pedido do desenho do produto e dos serviços de desenvolvimento.

As fontes de conhecimento para a inovação que utilizam são:

- Fornecedores (equipamento e matérias-primas);
- Observação direta da competição;
- Ideias geradas internamente;
- Clientes que têm uma relevância especial e que são bastante particulares (muitos deles são empresas ou empresários que têm uma ideia concreta sobre um dado produto mas que não

têm meios nem ferramentas para desenhar e desenvolvê-lo). Os clientes têm um papel fundamental numa fase preliminar do processo de inovação. No caso do design de guitarras com materiais compostos, a empresa usa uma mistura de design participativo e uma abordagem *lead user* para obter conhecimento externo. Convidam alguns músicos para avaliar os instrumentos e obter algum *feedback* sobre as características das guitarras. Esse *feedback* é útil para futuros melhoramentos.

O caso de teste utilizado neste projeto foi fornecido pela Ideia.m e recolhida informação sobre fóruns de discussão relacionados com as temáticas de guitarras e outros acessórios musicais eletrónicos.

Como se pode constatar, a presente plataforma desenvolvida pode ser bastante útil para fornecer conhecimento a variadíssimas empresas, levando-as a um processo de inovação.

De seguida serão aplicados algoritmos de classificação de autores para que o conteúdo disponibilizado, proveniente de fóruns, seja ordenado de acordo com o autor que proferiu esse comentário. Já foi elaborado um levantamento do estado de arte neste sentido, e futuramente pretende-se implementar estas novas funcionalidades no sistema.

5.2 Considerações Finais

A inovação é de enorme importância para as empresas e está estritamente relacionada com o conhecimento. Este pode adquirir-se a partir da grande quantidade de informação existente na web, nomeadamente em fóruns de discussão. Desta forma, tornou-se pertinente encontrar uma solução que permitisse pesquisar, analisar e retirar conteúdos existentes nestas plataformas.

Efetuou-se então um estudo sobre toda a temática relativa aos fóruns de discussão e a algumas técnicas para a extração do conhecimento. Procedeu-se em seguida a uma recolha de dados que possibilitasse idealizar uma ferramenta com vista à resolução deste problema.

Investigaram-se plataformas de tratamento de *Big Data* para permitir a escalabilidade do sistema, bem como a interligação entre as diversas ferramentas importantes para este processo e desenvolveu-se um sistema com essa finalidade.

Posteriormente foram pensados e realizados testes, tendo em vista a validação do sistema criado. Assim, de acordo com os resultados apurados constatou-se que os objetivos propostos foram alcançados, pois se conseguiu desenvolver uma plataforma de processamento de dados escalável e efetuar a indexação dos resultados obtidos.

A arquiteta apresentada é modular, sendo possível adicionar novos *plugins* na estrutura do *Nutch* para que possa ser feito o *parse* dos conteúdos de outros *templates* de fóruns de discussão existentes.

Um outro aspeto que pode ser realizado para melhorar a qualidade dos resultados será a introdução de um *cluster* de computadores interligados para efetuar as diversas operações descritas do processo. Neste caso, foi efetuada uma pseudo-distribuição com um único computador. Espera-se aperfeiçoar os resultados com base na inserção de novas máquinas na criação de um *cluster* e tirar partido da computação distribuída.

Torna-se no entanto evidente que o sistema criado poderia ser melhorado com a realização de mais testes,com maior profundidade e com a adição de novas funcionalidades.

Conclusão

Anexo A

Estudo sobre tecnologias

Tabela A.1: Ferramentas para Web Crawling - parte 1

Nome	**Características**
Weblech	- *MultiThread* - Desenvolvido em Java - *Open Source* - Autenticação HTTP básica - Ferramenta inteiramente caracterizada pelo obtenção de sítios *Web* - Simula o comportamento padrão de um navegador *Web* - Não tem GUI, a configuração é feita através do *spider* é através do uso de um simples ficheiro de configuração - Pesquisa em profundidade ou através do primeiro em largura de forma transversal no sítio web - Filtragem do URL candidato, permitindo pesquisar um servidor *Web*, um diretório ou toda a *web* - *Cache* configurável dos ficheiros obtidos, permitindo reiniciar sem existir a necessidade de descarregar tudo novamente - Priorização dos URLs, obtendo-se os ficheiros mais interessantes primeiro, e deixando os menos apetecidos para último ou até ignorá-los completamente - Ponto de verificação, pode ser feito um *snapshot* do estado do *crawler* no meio da sua execução e reiniciar sem perder o seu processamento - Não é adequado para obter URL únicos - Link: http://weblech.sourceforge.net/
Websphinx	- *crawler* para uma pequena parte da web(como um único sítio) automaticamente - *Open source* - Escito em java - *Crawler Workbench* - Biblioteca WebSPHINX - Link: http://www.cs.cmu.edu/ rcm/websphinx/
Jspider	- Escrito em java *Open Source* - Cria um Mapa do sítio - Motor web de *Spider* altamente customizável e configurável - *Download* completo dos sites - Verificação de link de externo ou interno - Link: http://j-spider.sourceforge.net/
Wget	- Escrito em C - *Open Source* - Usa HTTP, HTTPS e FTP - Ferramenta não interativa de linha de comandos - Link: - http://www.gnu.org/software/wget/

Tabela A.2: Ferramentas para Web Crawling - parte 2

Nome	**Características**
WebSuck	- Escrito em Java - Atravessa as páginas especificadas, verificando os links e os ficheiros de dados - É adequado para a obtenção de galeria de fotos - Possui interface gráfica - Link: http://www.ake.nu/software/websuck/
Crawler4j	- Escrito em Java - *Open source* - Fornece uma interface simples - Possui interface gráfica - Link: http://code.google.com/p/crawler4j/
Googlebot	- Pesquisa milhões de páginas - Usa um algoritmo que determina quais os sites pretendidos na busca, quantas vezes e quantas páginas devem ser buscadas em cada site - Começa com uma lista de URLs, gerados por processos anteriores de *crawl* e aumenta com dados dos *Sitemaps* fornecidos por webmaster - Link: http://support.google.com/webmasters/bin/answer.py?hl=en answer=182072;
WebWatcher	- Acompanha o utilizador página a página enquanto este navega na web, destacando os links que são de interesse - Tem uma estratégia para dar conselhos que é aprendida a partir do *feedback* de *tours* anteriores - Link: http://www.cs.cmu.edu/ webwatcher/
Teleport	- Java,C - Dez *threads* simultâneas - Usa FTP - *Project Scheduler* - Navegação *offline* completa - Filtros de carregamento de páginas - Suporte de *cookies* - Sistema de conexão automático - Suporte de *Firewall* e servidor *proxy* - Profundidade de exploração interna e externa configurável - Link: http://www.tenmax.com/teleport/pro/features.htm;
Heritrix	- Java - *Open source* - Extensível - Escalável - Arquivo de qualidade - Link: http://crawler.archive.org/
Nutch	- biblioteca do Hadoop - Escalável - Robusto - *Open source* - Link: http://nutch.apache.org/about.html
FAROO	- *Crawler* distribuído - O utilizador pode decidir quais os resultados que são mais importantes - Escala dinamicamente com o crescimento da web - Link: http://www.faroo.com/hp/p2p/p2p.html

Tabela A.3: Ferramentas para Pesquisa de Texto e Indexação

Nome	**Características**
Lucene	- Fornece uma tecnologia de pesquisa e indexação - Capacidades avançadas de análise - Realce de ocorrências - Baseado em java - Link: http://lucene.apache.org/core/
Solr	Fornece uma pesquisa distribuída e replicação dos índices - Pesquisa de Texto completo - Realce de ocorrências - Pesquisa facetada - Clustering dinâmico - Integração com Base de Dados - Tratamento de documentos como PDF e Word - Pesquisa geoespacial - Escalável - Pesquisa distribuída - Replicação de índice - Link: http://lucene.apache.org/solr/

Tabela A.4: Ferramentas para Armazenamento de Dados

Nome	**Características**
Neo4j	- *Open source* - Desenvolvido em java - Base de dados baseada em grafos - Escalável - Mecanismo de persistência totalmente transacional - Link:http://neo4j.org/learn/
MySql	*Open source* - Escalável e com alta performance - Portável e compatível - Replicação facilmente configurável - Interfaces gráficas de fácil utilização - Suporta *Triggers*, Cursores, controlo transacional - Link: http://www.mysql.com/products/
Hadoop	- Escalável - *Framework* com processamento distribuído de grandes conjuntos de dados, utilizando *clusters* de computadores - Oferece a cada computador uma computação local e armazenamento - *Open source* - Confiável - Computação distribuída - Link: http://hadoop.apache.org/

Anexo B

Estudo sobre fóruns de discussão

1. **Nome**: *SSGuitar.com*

   Url: http://www.ssguitar.com/

   Temática: *Amplifier Discussion* (discussão de amplificadores)

   Informações gerais:

Tipos de Utilizadores	Ascensão de utilizadores (nº posts)	Limite da caixa de entrada
New Member	0	5
Chipper	1	10
SSGuitar Apprentice	5	20
SSGuitar Regular	10	30
Master SSGuitarist	50	40
Elite SSGuitarist	100	50
Legendary	250	100

   Características específicas:

   - O *new member* só pode receber mensagens;
   - Para a classificação dos autores são utilizados os *chip points*;
   - Para votar é preciso ser no mínimo *SSGuitar Regular*;
   - A votação processa-se pela adição ou remoção de um ponto ao utilizador;

   Formato das mensagens:

   - Título;
   - Tipo de mensagem (*on*, *reply*) e data de envio do *post* (mês dia, ano, hora);
   - Mensagem;
   - Parte lateral:

  - Nome de utilizador;
  - Cargo;
  - *Chip Points*;
  - Número de *posts*;
  - *Avatar* (opcional).

Informações do perfil do utilizador:

- Número de *posts*;
- *Chip points*;
- Idade;
- Localização;
- Data do Registo;
- Tempo atual;
- Linguagem;
- Data da última participação;
- Ao lado:
  - Nome de utilizador;
  - Cargo;
  - *Avatar* (opcional);
  - Informação *offline*/*online*.
  - *Posts*: Lista de *posts* (filtrar por mensagens, tópicos, arquivos);
  - Estatísticas.

Template: SMF

2. **Nome**: *Composites Central*

   Url: http://www.compositescentral.com/

   Temática: Desenho de compósitos;

   Informações gerais:

   - Tem uma listagem dos integrantes do grupo de discussão.
   - Reputação: pode ser positiva, negativa (se o administrador permitir isso) ou neutral (se a pessoa que está a pontuar não percebe o método de avaliação fornecido pelo administrador).
   - Classificação de *threads*: pode ser feita através de uma escala que varia de 1 a 5 estrelas. É apresentada a média das pontuações dadas.

- Tratamento de *tags*: utilizadas para pesquisar *threads* de conteúdos semelhantes. Podem ser adicionadas *tags* à lista de *tags* para descrever com uma palavra ou frase o conteúdo da frase. Se já existir a *tag* correspondente só é preciso escolher a partir da lista fornecida. As *tags* são introduzidas na *thread* pelo utilizador que a inicializou, podendo outros utilizadores adicionar/removê-las.

Formato das Mensagens:

- Data do *post*(mês dia, ano, hora);
- Título (só na primeira mensagem da *thread*);
- Mensagem;
- Parte lateral:
  - Nome de utilizador;
  - Cargo;
  - Data de entrada no fórum do utilizador;
  - Localização;
  - Número de *posts*;
  - Número de agradecimentos;
  - Estado(*offline*/*online*).

Informações do utilizador:

(a) Mensagem para os visitantes;

(b) Sobre o utilizador:

i. Informações:
   - Biografia;
   - Localização;
   - Interesses;
   - Ocupação.

ii. Assinatura.

iii. Estatísticas do utilizador;

iv. Amigos (opcional).

Template: *vbulletin*

3. **Nome:** *The Acoustic Guitar*

   Url: http://www.acousticguitarforum.com/forums/index.php

   Temática: comunidade de guitarras acústicas.

   Informações gerais: muito semelhante ao segundo fórum apresentado.

   Formato das mensagens:

- Data da publicação;
- Nome de utilizador;
- Cargo;
- Mensagem.
- Parte Lateral : Informações do utilizador.

Informações do Utilizador:

(a) Sobre o utilizador:

i. Informações:

- Biografia;
- Localização;
- Interesses;
- Ocupação.

ii. Assinatura.

(b) Estatísticas do utilizador.

Template: *vbulletin*

4. **Nome:** *The Gear Page*

   Url: http://www.thegearpage.net/

   Temática: comunidade sobre música, focando-se primeiramente em guitarristas e baixistas.

   Informações gerais: Semelhante ao segundo fórum.

   Formato das mensagens: Semelhante ao segundo fórum.

   Informações do Utilizador: Só com registo.

   Template: *vbulletin*

5. **Nome:** *Ultimate Guitar OnLine*

   Url: http://ultimate-guitar-online.ultimate-online-services.com/forum/

   Temática: design e construção de guitarras.

   Informações gerais: semelhante ao primeiro caso apresentado;

   Formato das mensagens: semelhante ao primeiro caso;

   Informaçãoes do Utilizador: semelhante ao primeiro caso;

   Template: SMF.

Grupos de Discussão fornecidos pela empresa Ideia.m

1. **Nome:** *Guitar Player Magazine Forums* **Url:** http://forums.musicplayer.com/ubbthreads.php/category/10/G
   Temática: Guitarras e acessórios

   Formato das mensagens:

   - Título da mensagem, número da mensagem e data (dia/mês/ano hh:mm);
   - Corpo da mensagem;
   - Parte lateral:
     - Nome de utilizador;
     - Estado *offline*/*online*;
     - Cargo;
     - *Avatar*;
     - Data de Registo do utilizador;
     - Número de *Posts* efetuados;
     - Localização.

   Informações do Utilizador: é preciso estar registado

   Template: *UBB.Threads*

2. **Nome:** *Guitar World*

   Url: http://www.guitarworld.com/

   Temática: Guitarras e mundo da música.

   Informações: só tem um *blog*. As notícias são fornecidas por *NewBay Media*.

   Formato das mensagens: -

   Informações do Utilizador:-

   Template: -

3. **Nome:** *Premier Guitar*

   Url: http://www.premierguitar.com/

   Temática: Guitarras, acessórios de música e mundo da música.

   Informações: Apenas tem *reviews* de notícias e artigos. Para além disso, tem comentários a essas *reviews*.

   Formato das mensagens: -

   Informações do Utilizador:-

   Template: -

4. **Nome:** *Guitar Noize*

   Url: http://guitarnoize.com/

Temática: Notícias sobre guitarras

Informações: Formato de *blog* onde existe uma comunidade que coloca reações e comentários sobre uma notícia específica.

Formato das mensagens (neste caso, dos comentários):

- Nome de Utilizador;
- Tempo decorrido desde a escrita da mensagem;

Informações do Utilizador:

- Nome de utilizador;
- Localização;
- *Site*;
- Papéis que desempenha.

Template: -

5. **Nome:** *Guitar Noise*

   Url: http://www.guitarnoise.com/forums/

   Temática: Guitarras e acessórios. Música

   Formato das mensagens:

   - Nome do Post;
   - Autor;
   - *Timestamp* do *post*;
   - Comentário;
   - Assinatura;
   - Parte Lateral:
     - Nome de utilizador;
     - Cargo;
     - Número de *Posts*;
     - Data de entrada na comunidade;
     - Localização;

   Informações do Utilizador: é necessária autenticação para aceder ao perfil do utilizador.

   Template: *phpBB*

6. **Nome:** *Guitar lifestyle*

   Url: http://www.guitarlifestyle.com/

   Temática: Guitarristas, história das guitarras e guitarristas.

   Informações: Formato de *blog* onde são colocadas notícias e onde são feitos comentários.

   Formato das mensagens (neste caso, dos comentários):

   - Nome de utilizador (por vezes tem links para outras páginas web);
   - Corpo da mensagem;
   - Data(mês de dia th, ano at hh:mm)

   Informações do Utilizador:-

   Template: -

Anexo C

Manual de Utilização

Para correr a plataforma desenvolvida é necessário ter um ambiente *linux* (nativo ou máquina virtual). No início do projeto foi experimentado instalar o *Hadoop* em *windows* com recurso ao *Cygwin*. Este projeto pretende simular um ambiente *linux* em *windows*, fornecendo as ferramentas principais para esse efeito. Apesar destas facilidades concedidas pelo projeto acima descrito achou-se por bem desenvolver o projeto de dissertação num ambiente *linux* nativo uma vez que para um projeto desta envergadura questões de performance e fiabilidade devem ser tidos em conta.

Componentes:

- Versão *Hadoop*;
- Pasta extracaoWeb:
  - *NutchCrawl*: *script continuousCrawl.sh* e *Nutch*;
  - Servidor *Solr* 3.6.2;

Em primeiro lugar é necessário ter o *Hadoop* instalado na máquina.

C.1 Instalação do Hadoop

Para a execução do *Hadoop* é recomendado possuir na máquina uma versão do java superior à 1.6.

1. efetuar o download de uma versão binária do *Hadoop*(para as experiências efetuadas foi utilizada versão estável 1.0.3);
2. instalar na localização /usr/local/;

C.2 Configuração do *Nutch*

Para a realização das experiências descritas neste projeto, foi preciso obter uma versão estável do *Nutch*. As modificações efetuadas nesta ferramenta foram efetuadas no IDE eclipse com a introdução de alguns *plugins* do mesmo. De referir que segundo a documentação fornecida pelos programadores deste projeto a localização dos *plugins* tem de ser colocada com a localização existente no sistema de ficheiros local.

C.3 Configuração do Servidor *Solr*

Para se verificar uma maior compatibilidade entre as plataformas *Nutch* e *Solr* é conveniente instalar a versão recomendada do *Solr* uma vez que foi testada uma versão mais avançada do mesmo sistema, mais concretamente a versão 4.0, e verificou-se que embora fosse possível efetuar a ligação era necessária a transferência de diversos ficheiros jar da plataforma *Solr* para o *Nutch*.

De referir que o servidor deve ser previamente iniciado antes de ser corrido o *script continuousCrawl.sh*.

C.4 Funcionamento do Programa

- Colocar na pasta pessoal a pasta extracaoWeb.
- De seguida é recomendado que se definam algumas variáveis que vão ser utilizadas pelo *script*. Devem ser definidas as variáveis *NUTCH_HOME* e *HADOOP_HOME* com o caminho relativo à localização do *Nutch* e do *Hadoop* respetivamente. Se não se efetuar este passo o próprio *script* verificará que não estão definidas e atribuirá valores padrão. Os valores padrão definidos estão relacionados com a localização do *Nutch* e do *Hadoop* na presente máquina.
- Iniciar o *Hadoop*;
- Iniciar o servidor *Solr*;
- Correr o *script continuousCrawl.sh*;

O programa deverá iniciar de forma análoga à imagem C.1

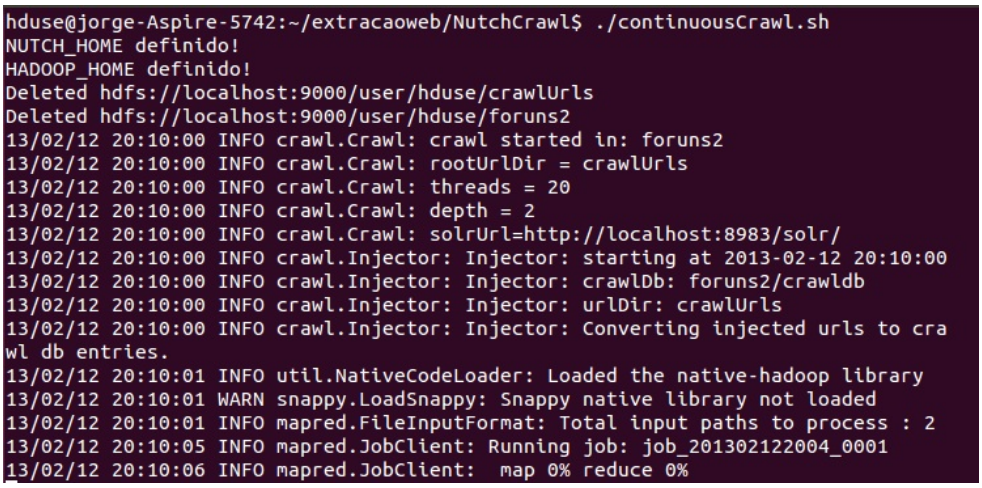


Figura C.1: Exemplo de execução da consola

Anexo D

Código do Script

```
#!/bin/bash

export HADOOP_SHELL=$HADOOP_HOME/bin/hadoop

function echoThenRun () { # echo and then run the command
  echo $1
  $1
  echo
}

#verifica se a ávarivel nutch_home áest definida
if [ "$NUTCH_HOME" == "$HOME/extracaoweb/NutchCrawl" ]; then
  echo "NUTCH_HOME definido!"
else
  echo "NUTCH_HOME ãno definido!"
  export NUTCH_HOME=$HOME/extracaoweb/NutchCrawl
fi

#verifica se a ávarivel hadoop_home áest definida
if [ "$HADOOP_HOME" == "/usr/local/hadoop" ]; then
  echo "HADOOP_HOME definido!"
else
  echo "HADOOP_HOME ãno definido!"
  export HADOOP_HOME="/usr/local/hadoop"
fi

#verifica se áj tem uma ãverso do projeto Nutch íconstruda
if [[ ! -d "build" ]]
then
       echoThenRun "ant"
fi

#verifica se a pasta dos crawlUrls existe
```

```
34 $HADOOP_SHELL fs -test -d crawlUrls
35 if [ $? -eq 0 ]; then
36   $HADOOP_SHELL fs -rmr crawlUrls
37   $HADOOP_SHELL fs -put $NUTCH_HOME/urls crawlUrls
38 else
39   $HADOOP_SHELL fs -put $NUTCH_HOME/urls crawlUrls
40 fi
41
42 #verifica se a pasta intermedia para o processo de crawl existe
43 $HADOOP_SHELL fs -test -d foruns2
44 if [ $? -eq 0 ]; then
45   $HADOOP_SHELL fs -rmr foruns2
46   $HADOOP_SHELL fs -mkdir foruns2
47 else
48   $HADOOP_SHELL fs -mkdir foruns2
49 fi
50
51 #corre o processo de crawl
52 $HADOOP_SHELL jar $NUTCH_HOME/build/nutch-1.5.1.job org.apache.nutch.crawl.Crawl
       crawlUrls -solr http://localhost:8983/solr/ -dir foruns2 -depth 3 -threads 20
53
54 #efetua o dump dos segmentos gerados pelo processo de crawl
55 $HADOOP_SHELL fs -test -d crawlData2
56 if [ $? -eq 0 ]; then
57   $HADOOP_SHELL fs -rmr crawlData2
58   $HADOOP_SHELL jar $NUTCH_HOME/build/nutch-1.5.1.job org.apache.nutch.segment.
         SegmentReader -dump foruns2/segments/* crawlData2/ -nogenerate #-noparse -
         noparsedata -noparsetext
59 else
60   $HADOOP_SHELL jar $NUTCH_HOME/build/nutch-1.5.1.job org.apache.nutch.segment.
         SegmentReader -dump foruns2/segments/* crawlData2/ -nogenerate #-noparse -
         noparsedata -noparsetext
61 fi
```

Referências

[BCL12] Vinayak Borkar, MJ Carey e Chen Li. Inside Big Data management: ogres, onions, or parfaits? *Joint Conference Berlin, Germany*, pages 3–14, 2012. Citado nas páginas 16 e 31.

[BPM] Sotiris Batsakis, Euripides G M Petrakis e Evangelos Milios. Improving the Performance of Focused Web Crawlers. pages 1–29. Citado na página 38.

[Bri98] Sergey Brin. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, April 1998. doi:10.1016/S0169-7552(98)00110-X. Citado na página 11.

[CB99] Soumen Chakrabarti e Martin Van Den Berg. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks*, 31:1623–1640, 1999. Citado na página 9.

[CCA+] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M Hellerstein, Khaled Elmeleegy e Russell Sears. MapReduce Online. Citado na página 17.

[CGM98] Junghoo Cho e Hector Garcia-Molina. Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1-7):161–172, April 1998. doi:10.1016/S0169-7552(98)00108-1. Citado na página 11.

[CM04] Carlos Castillo e Mauricio Marin. Scheduling algorithms for Web crawling. *... and LA-Web, 2004. ...*, pages 0–7, 2004. Citado na página 11.

[CWLS08] Gao Cong, Long Wang, Chin-yew Lin e Young-in Song. Finding Question-Answer Pairs from Online Forums. pages 467–474, 2008. Citado nas páginas 2 e 14.

[DE05] Laurie P. Dringus e Timothy Ellis. Using data mining as a strategy for assessing asynchronous discussion forums. *Computers & Education*, 45(1):141–160, August 2005. doi:10.1016/j.compedu.2004.05.003. Citado nas páginas 18 e 21.

[EGK01] Martin Ester, Matthias Große HP Kriegel. Focused Web crawling: A generic framework for specifying the user interest and for adaptive crawling strategies. *... on Very Large ...*, 2001. Citado na página 10.

[EPF08] Jaliya Ekanayake, Shrideep Pallickara e Geoffrey Fox. MapReduce for Data Intensive Scientific Analyses. *2008 IEEE Fourth International Conference on eScience*, pages 277–284, December 2008. doi:10.1109/eScience.2008.59. Citado na página 18.

[GHNS05] Natalie Glance, Matthew Hurst, Kamal Nigam e Matthew Siegler. Deriving marketing intelligence from online discussion. *Proceedings of the*, page 419, 2005. doi:10.1145/1081870.1081919. Citado na página 20.

REFERÊNCIAS

[had13] Welcome to Apache Hadoop! Disponível em `http://hadoop.apache.org/`, Outubro 2013. Citado na página 22.

[HL] Eduard Hovy e Chin-yew Lin. Automated Text Summarization and the Summarist System. pages 197–214. Citado na página 15.

[HN99] Allan Heydon e Marc Najork. Mercator : A scalable , extensible Web crawler. 2:219–229, 1999. Citado nas páginas 9 e 40.

[Hu06] Minqing Hu. Opinion extraction and summarization on the web. *Proceedings Of The National Conference On Artificial*, pages 1621–1624, 2006. Citado na página 15.

[HZ07] Jizhou Huang e Ming Zhou. Extracting chatbot knowledge from online discussion forums. *Proceedings of IJCAI*, pages 423–428, 2007. Citado na página 13.

[KCSR04] Rohit Khare, Doug Cutting, K Sitaker e Adam Rifkin. Nutch: A flexible and scalable open-source web search engine. *Oregon State University*, 2004. Citado na página 32.

[Kos00] Raymond Kosala. Web mining research: A survey. *ACM Sigkdd Explorations Newsletter*, 2(1), 2000. Citado nas páginas 1 e 12.

[LH05] Bing Liu e M Hu. Opinion observer: analyzing and comparing opinions on the Web. *international conference on World Wide Web*, pages 342–351, 2005. Citado nas páginas 13, 15, 19 e 20.

[Liu04] Bing Liu. Editorial: special issue on web content mining. *Acm Sigkdd explorations newsletter*, 6(2):2–5, 2004. Citado na página 1.

[Liu09] Bing Liu. *Web Data Mining - Exploring Hyperlinks, Contents and Usage Data*. 2 edition, 2009. Citado nas páginas 8 e 9.

[LPD+09] Hui Li, Qiangqiang Peng, Yajun Du, Ying Zhao, Shaoming Chen e Zhaoqiong Gao. Focused Web Crawling Strategy Based on Web Semantic Analysis and Web Link Analysis. 6:1793–1800, 2009. Citado na página 9.

[MMD+07] José E. Moreira, Maged M. Michael, Dilma Da Silva, Doron Shiloach, Parijat Dube e Li Zhang. Scalability of the Nutch search engine. *Proceedings of the 21st annual international conference on Supercomputing - ICS '07*, page 3, 2007. doi:10.1145/1274971.1274975. Citado na página 32.

[MMKR] Ion Muslea, Steve Minton, Craig Knoblock e Marina Rey. A Hierarchical Approach to Wrapper Induction. pages 190–197. Citado na página 13.

[MPR] Filippo Menczer, Gautam Pant e Miguel E Ruiz. Evaluating Topic-Driven Web Crawlers Page importance. pages 241–249. Citado na página 11.

[MPS04] Filippo Menczer, Gautam Pant e Padmini Srinivasan. Topical web crawlers: Evaluating adaptive algorithms. *ACM Transactions on Internet Technology*, V(February):1–38, 2004. Citado na página 10.

[NM10] Bogdan Nicolae e Diana Moise. BlobSeer: Bringing high throughput under heavy concurrency to Hadoop Map-Reduce applications. *Parallel & Distributed . . .*, 2010. Citado na página 31.

REFERÊNCIAS

[OP08] Christopher Olston e Sandeep Pandey. Recrawl scheduling based on information longevity. *Proceeding of the 17th international conference on World Wide Web - WWW '08*, page 437, 2008. doi:10.1145/1367497.1367557. Citado na página 11.

[Pur] Bernice Purcell. The he emergence of " big data " technology and analytics. pages 1–7. Citado na página 16.

[RM98] Dragomir R Radev e Kathleen R Mckeown. Generating Natural Language Summaries from Multiple On-Line Sources. 1998. Citado na página 16.

[RR07] Colby Ranger e Ramanan Raghuraman. Evaluating mapreduce for multi-core and multiprocessor systems. *..., 2007. HPCA 2007. ...*, pages 13–24, 2007. doi:10.1109/HPCA.2007.346181. Citado na página 17.

[SM04] Lokesh Shrestha e Kathleen McKeown. Detection of question-answer pairs in email conversations. *Proceedings of the 20th international conference on Computational Linguistics - COLING '04*, pages 889–es, 2004. doi:10.3115/1220355.1220483. Citado na página 14.

[vH86] E. von Hippel. Lead Users: A Source of Novel Product Concepts. *Management Science*, 32(7):791–805, July 1986. doi:10.1287/mnsc.32.7.791. Citado na página 2.

[WH00] Rüdiger Wirth e J Hipp. CRISP-DM: Towards a standard process model for data mining. *...Applications of Knowledge Discovery and Data Mining*, (24959), 2000. Citado na página 3.

[WS02] J. Wiley e Sons. The PDMA Toolbook for New Product Development: effective methods, tools, and techniques. 2002. Citado na página 1.

[YCWH09] JM Yang, Rui Cai, Chunsong Wang e Hua Huang. Incorporating site-level knowledge for incremental crawling of web forums: A list-wise strategy. *on Knowledge*, pages 1375–1383, 2009. Citado na página 10.

[YH11] BW Yohanes e HKW Handoko. Focused Crawler Optimization Using Genetic Algorithm. *TELKOMNIKA*, 9(3), 2011. Citado na página 10.

[Zha05] Yanhong Zhai. Web data extraction based on partial tree alignment. *the 14th international conference on World Wide Web*, pages 76–85, 2005. Citado na página 13.