DETECÇÃO DE RÉPLICAS DE SÍTIOS WEB EM MÁQUINAS DE BUSCA USANDO APRENDIZADO DE MÁQUINA

RICKSON GUIDOLINI

DETECÇÃO DE RÉPLICAS DE SÍTIOS WEB EM MÁQUINAS DE BUSCA USANDO APRENDIZADO DE MÁQUINA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: NIVIO ZIVIANI

Belo Horizonte

Março de 2011

Guidolini, Rickson

G948d Detecção de Réplicas de Sítios Web em Máquinas de Busca Usando Aprendizado de Máquina / Rickson Guidolini. — Belo Horizonte, 2011.
xvi, 58 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

Orientador: Nivio Ziviani.

1. Computação - Teses. 2. Recuperação de Informação - Teses. I. Orientador. II. Título.

CDU 519.6*73 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Detecção de réplicas de sítios Web em máquinas de busca usando aprendizado de máquina

RICKSON GUIDOLINI

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. NIVIO ZIVIANI - Orientador
Departamento de Ciência da Computação - UFMG

PROF. ADRIANO ALONSO VELOSO - Co-orientador
Departamento de Ciência da Computação - UFMG

PROF. EDLENO SILVA DE MOURA
Departamento de Ciência da Computação - UFAM

PROF. MARCO ANTÔNIO PINHEIRO DE CRISTO
Departamento de Ciência da Computação - UFAM

Belo Horizonte, 11 de março de 2011.

À memória de meu mais que primo, meu grande amigo Richardi Fassina.

Agradecimentos

Em primeiro lugar, agradeço a meus pais, Fernandes e Domingas, pelo amor, apoio e pelos sacrifícios realizados para que eu pudesse chegar até aqui. Agradeço também a meus irmãos, Rodrigo e Rânik, pela amizade, apoio e confiança.

Aos amigos da república LosComputeros, Anisio Lacerda, Tupy Carneiro, Wallace Favoreto e Alan Castro, pela amizade que sempre guardarei comigo, pelo companheirismo e ajuda.

Aos companheiros de LATIN, Fabiano Botelho, Guilherme Menezes, Thales Filizola, Vinícius Tinti e Wladmir Brandão, pela amizade e ajuda. Em especial ao Cristiano Carvalho, que batizou o algoritmo DREAM. Ao Thiago Salles, pela amizade e companhia nas idas ao café.

Aos Professores que me guiaram nesse projeto, Prof. Nivio Ziviani, Prof. Adriano Veloso e Prof. Edleno Moura. Em especial ao meu orientador Prof. Nivio, pela preocupação, disponibilidade e conhecimento transmitido. Também agradeço ao Prof. Altigran Soares, Prof. Marcos Gonçalves e Prof. Marco Cristo, pelas contribuições neste trabalho.

À minha namorada, Merielen, pela compreensão e cooperação para superar esses dois anos a mais de 500 km de distância.

Ao pessoal da secretaria do departamento, pela simpatia e disposição em ajudar. Em especial à Tulia, pelo sorriso e pelo "bom dia" que sempre animava o meu dia.

Acima de tudo, agradeço a Deus por ter me concedido sabedoria, força e proteção pra chegar até aqui.

Resumo

Estima-se que pelo menos 30% de todo o conteúdo disponibilizado na Web seja conteúdo replicado impondo sérios desafios às máquinas de busca, tais como desperdício de recursos computacionais e diminuição de efetividade na busca. Dessa forma, a detecção de sítios web replicados é atualmente um pré-requisito para o desenvolvimento de máquinas de busca modernas. No entanto, a grande quantidade de conteúdo dinâmico e o imenso volume de potenciais réplicas tornam a tarefa de detecção de sítios replicados extremamente difícil.

O objetivo deste trabalho é a elaboração de novas técnicas de detecção de sítios replicados. Características intrínsecas de sítios replicados foram estudadas e, dependendo do valor que essas características assumem, podemos discriminar pares de sítios replicados e não-replicados. No entanto, (i) há uma grande dificuldade em encontrar o valor ideal para o qual temos a melhor discriminação, e (ii) há uma grande quantidade de características a serem combinadas de forma a se obter a melhor discriminação. Sendo assim, avaliamos a utilização de técnicas de aprendizado de máquina para a detecção de sítios web replicados e propomos um algoritmo denominado DREAM (Detecção de RÉplicas usando Aprendizado de Máquina), que combina e pondera de forma automática as características discriminatórias propostas.

O algoritmo DREAM foi avaliado em uma coleção real de sítios coletados da Web brasileira. Quatro algoritmos de aprendizado de máquina foram avaliados: árvores de decisão (C4.5), classificação associativa (LAC), combinações de árvores (*Random Forests*) e máquina de vetor de suporte (SVM). Resultados experimentais mostram que o algoritmo DREAM, usando combinações aleatórias de árvores, supera em 35,1% o melhor método da literatura em termos da métrica F1. Os resultados experimentais indicam que a utilização de técnicas de aprendizado de máquina leva a resultados superiores aos obtidos pelos algoritmos existentes na literatura.

Palavras-chave: Detecção de Réplicas de Sítios, Recuperação de Informação, Aprendizado de Máquina.

Abstract

Earlier work estimate that at least 30% of all content available on the Web is replicated content, posing serious challenges to search engines, such as waste of computational resources and decrease in the search effectiveness. Thus, detection of replicated websites is currently a prerequisite for the development of modern search engines. However, the large amount of dynamic content and the huge amount of potential replicas make the detection of replicated website an extremely difficult task.

This work focus on the development of new techniques for detecting replicated websites. Intrinsic features of replicated sites were studied and, depending on the value assumed by these features, we can discriminate pairs of replicated and non-replicated sites. However, (i) there is great difficulty in finding the optimal values that lead to the best discriminative performance, and (ii) there is a lot of features to be combined in order to obtain the best discrimination. Therefore, we evaluated the use of machine learning techniques for detecting replicated websites and we propose an algorithm called DREAM (Detecção de RÉplicas usando Aprendizado de Máquina), which automatically combines and weights the proposed discriminative features.

The algorithm DREAM was evaluated through the use of a real collection of sites, collected from the Brazilian Web. Four machine learning algorithms were evaluated: decision trees (C4.5), lazy associative classification (LAC), combinations of trees (random forests) and support vector machine (SVM). Experimental results show that the algorithm DREAM outperforms by 35.1% the best method in the literature in terms of F1 measure. The experimental results indicate that the use of machine learning techniques lead to superior results comparing to the known algorithms in the literature.

Keywords: Web Site Replica Detection, Information Retrieval, Machine learning.

Sumário

Capítulo 1

Introdução

A Web contém o maior repositório de informação já construído pelo homem. Seja pelo computador, celular, *palmtop* ou qualquer outro aparelho, móvel ou não, hoje podemos realizar, por meio dela, tarefas como compras, *marketing*, correio, ler livros e jornais, ver fotos, assistir a filmes, saber de fatos que acabam de acontecer ou até mesmo acompanhá-los enquanto acontecem. Tal sucesso deve-se, em grande parte, à facilidade com que pessoas anônimas conseguem publicar conteúdo na Web. Antes da Web, o controle sobre as publicações estava nas mãos das editoras, mas hoje, qualquer pessoa pode criar um documento eletrônico de qualquer natureza.

A facilidade de publicação torna a Web um ambiente bastante volátil e capaz de abrigar uma quantidade incalculável de informação. Mas com tanta informação disponível certamente é preciso ajuda para encontrar o que necessitamos. Nesse ponto entram as máquinas de busca. Uma máquina de busca é um sistema que recebe uma consulta (que expressa uma necessidade de informação do usuário), pesquisa um índice (que contém informações extraídas de documentos da Internet) e apresenta ao usuário uma lista ordenada de objetos (páginas, imagens, vídeos, entre outros) que atendem a necessidade do usuário.

A motivação para este trabalho tem origem na construção de uma máquina de busca dentro do Instituto Nacional de Ciência e Tecnologia para a Web[1] (InWeb). Um dos grandes problemas enfrentados no desenvolvimento desse tipo de sistema é a presença de sítios replicados em sua base de dados. Tal problema ocorre devido à existências de informação replicada na Web. Existem estudos na literatura que estimam que cerca de 30% da informação na Web é replicada [Broder et al., 1997; Fetterly et al., 2003; Henzinger, 2006]. As principais causas da replicação de sítios são:

[1] http://www.inweb.org.br/

- Múltiplos nomes de domínio: um mesmo sítio é cadastrado sob vários nomes diferentes no serviço de resolução de nomes de domínio. Esse artifício é geralmente utilizado no intuito de facilitar o acesso do usuário por meio de URLs diferentes, como http://www.bb.com.br e http://www.bancodobrasil.com.br, que apontam para o sítio do Banco do Brasil.

- Balanceamento de carga: alguns sítios possuem vários servidores para atender à demanda de acessos. Nessa estratégia, um desses servidores, o mais próximo ao usuário ou o menos sobrecarregado, é selecionado para responder a uma determinada requisição HTTP. Algumas vezes cada um desses servidores responde sob uma URL diferente. Esse comportamento faz com que os coletores web coletem e armazenem cópias do mesmo sítio sob URLs diferentes representando sítios diferentes.

- Franquia: um mesmo sítio é mantido por dois parceiros diferentes com duas URLs diferentes, mas com conteúdo semelhante.

- Razões Sociais: acontece quando um determinado grupo social mantém bases de conhecimentos distintas contendo a mesma informação. É o caso da base de dados *Protein Data Bank* (PDB), compartilhada e mantida por vários pesquisadores. O mesmo ocorre com o *Linux Documentation Project* (LDP).

No caso das máquinas de busca, a replicação de informação acarreta uma série de problemas, entre eles o desperdício de tempo durante o processo de coleta e durante o processamento da informação coletada, o desperdício de banda de Internet, de recursos de armazenamento e de processamento, entre outros. A informação replicada também afeta a qualidade das respostas exibidas ao usuário, uma vez que quando um sítio é replicado, a informação a respeito da conectividade desse sítio também é replicada. Isso interfere diretamente no funcionamento de algoritmos de *ranking*, tais como o PageRank [Brin & Page, 1998], CLEVER [Chakrabarti et al., 1998; Kleinberg, 1998] e Topic Distillation [Bharat & Henzinger, 1998], usados nas máquinas de busca mais importantes da atualidade. Outro problema causado pela replicação de sítios é a exibição de respostas que, do ponto de vista da máquina de busca, são diferentes, mas que do ponto de vista do usuário são iguais. Essa exibição de respostas muito similares pode aborrecer o usuário e passar a sensação de estar recebendo respostas repetidas.

O problema que estudamos neste trabalho é a detecção de réplicas de sítios web em bases de máquinas de busca. Detectar réplicas de sítios é uma tarefa importante, pois evita os problemas citados anteriormente, a saber, o desperdício de recursos, as anomalias nas funções de *ranking* e as repetições nas respostas exibidas ao usuário.

Além disso, uma coleta livre de réplicas geralmente possui uma diversidade maior de sítios cobertos do que uma coleta sem a remoção de réplicas, demandando os mesmos custos.

Detectar réplicas de sítios e detectar réplicas de páginas web [Henzinger, 2006; Manku et al., 2007; Radlinski et al., 2011] são tarefas similares, porém distintas quanto aos custos e benefícios. A detecção de réplicas de sítios pode ser considerada mais barata que a de páginas, pois, uma vez que um único sítio é composto por várias páginas, há bem menos sítios do que páginas na base de uma máquina de busca. Por outro lado, os objetivos da detecção de réplicas de sítios não são necessariamente alcançados pela detecção de réplicas de páginas. Por exemplo, se apenas a detecção de réplicas de páginas web é executada, a máquina de busca continuará recebendo (novas) páginas repetidas de um sítio replicado qualquer. Entretanto, se o sítio é identificado como réplica evita-se que o coletor continue visitando-o e suas páginas (replicadas) não serão coletadas novamente.

Vários fatores tornam a detecção de réplicas uma tarefa difícil. Uma ideia inicial para resolver este problema seria, por exemplo, verificar se os sítios suspeitos possuem o mesmo conjunto de páginas. No entanto, realizar essa verificação na base de dados da máquina de busca não é possível, devido à estratégia de cobertura por sítios adotada pelos coletores. Essa estratégia faz com que os coletores priorizem obter um maior número de sítios ao invés de sítios completos. Isso torna difícil encontrar, nas bases das máquinas de busca, interseções entre os conjuntos de páginas de sítios replicados.

Outra abordagem seria verificar se os sítios suspeitos possuem o mesmo conjunto de páginas diretamente na Web. Essa solução não é viável devido ao esforço exigido pela coleta das páginas dos sítios. Outra ideia para resolver o problema seria usar o endereço IP como um identificador único para cada sítio. Essa abordagem funcionaria muito bem, não fosse pelo fato de que um mesmo sítio pode responder sob vários IPs diferentes e um mesmo IP pode estar relacionado a vários sítios distintos.

Outro fator que torna a detecção de pares de sítios replicados em bases de máquinas de busca uma tarefa difícil é a dimensionalidade envolvida no problema. A solução por força bruta, por exemplo, requer a comparação entre todos os pares de sítios da base, o que leva a uma complexidade quadrática. Para evitar esse custo, a tarefa de detecção é comumente dividida em duas etapas: seleção de pares candidatos a réplica, que reduz a dimensionalidade do problema, e validação dos pares candidatos, que visa verificar com alta precisão se os pares selecionados são realmente réplicas. Esta dissertação contribui com novos resultados para a etapa de seleção de pares candidatos a réplica.

1.1 Objetivos

O objetivo principal desta dissertação é propor e avaliar abordagens baseadas em técnicas de aprendizado de máquina para serem aplicadas à etapa de seleção de pares de sítios web candidatos a réplica. Mais especificamente, pretendemos verificar se a utilização desse tipo de técnica pode melhorar os resultados obtidos até então pela utilização de heurísticas fixas na etapa de seleção de pares candidatos.

Nossa hipótese é que técnicas de aprendizado de máquina podem ajudar a selecionar as melhores características para a detecção de réplicas e combiná-las com o objetivo de maximizar a efetividade na tarefa de detecção de réplicas. Além disso, a base teórica por trás dessas técnicas pode nos ajudar a compreender melhor as soluções, ajudando-nos a obter uma melhor visão do problema.

1.2 Trabalhos Relacionados

A grande quantidade de dados replicados na Web tem motivado pesquisas com o objetivo de caracterizar e entender como tratar o problema. Nesse sentido, Bharat & Broder [1999] realizaram um estudo com o objetivo de apresentar uma imagem clara do nível de replicação de sítios na Web e propuseram um método automático para a classificação de réplicas de sítios em vários níveis de replicação. Tal método é composto por um estágio de seleção de pares de sítios candidatos a réplica, que se baseia em características estruturais dos sítios, e um estágio de classificação automática desses pares, que se baseia na comparação entre a estrutura e o conteúdo dos sítios em questão.

O argumento principal dos autores é que seu método tem um bom desempenho, pois depende principalmente da análise sintática de cadeias de URL, porém a precisão máxima relatada foi de 80%. A precisão tem papel importante na detecção de réplicas de sítios web, pois um sítio classificado erroneamente como réplica será removido das respostas da máquina de busca. Sendo assim, a precisão de 80% relatada pelos autores pode ser considerada baixa.

A maioria dos trabalhos existentes na literatura trata da seleção de pares candidatos, que corresponde à primeira etapa da tarefa detecção de réplicas de sítios web. Por exemplo, Bharat et al. [2000] apresentam uma família de métodos para seleção de pares candidatos a réplica baseados em evidências derivadas de estruturas dos sítios Web tais como URL, endereços IP e informações sobre apontadores. Os melhores resultados individuais foram Alcançados por um método baseado em endereço IP, o IP4, que analisa os quatro octetos do endereço IP. No entanto, o melhor resultado geral

foi alcançado por uma abordagem híbrida, que combina cinco algoritmos e alcançou uma precisão de 57% para uma revocação de 86%. Para executar a validação dos pares candidatos, que corresponde à segunda etapa da detecção de réplicas, os autores propuseram um método efetivo para a validação de pares candidatos, descrito na Seção 2.6, também adotado em outros trabalhos na literatura e por esta dissertação.

Novamente os autores se apóiam na afirmação de que seu método possui bom desempenho, mas, como alertado anteriormente, a precisão baixa pode causar problemas às respostas do sistema de busca. Vale ressaltar que nesse trabalho os resultados podem ter sido influenciados pela metodologia adotada pelos autores. Somente sítios com pelo menos 100 URLs foram considerados em seus experimentos e, na abordagem combinada, apenas os 100.000 resultados do topo das listas retornadas foram combinados. Fazendo isso, os autores podem ter analisado uma pequena parte do universo de possíveis réplicas e, provavelmente, uma parte mais fácil de classificar.

Em [Cho et al., 2000] os autores apresentam um trabalho cujo objetivo é encontrar coleções web duplicadas ao invés de réplicas de sítios web. A diferença entre esses objetivos é que encontrar coleções duplicadas também compreende a tarefa de encontrar pedaços replicados de sítios. Primeiramente o método proposto agrupa páginas com conteúdo similar e posteriormente expande esses grupamentos até que eles representem uma coleção completa. A principal desvantagem desse método é a necessidade de processar o conteúdo das páginas enquanto os métodos anteriores se baseiam principalmente na estrutura dos sites (URLs, endereço IP, etc.).

Vale ressaltar o estudo de caso relatado usando o coletor Web da máquina de busca Google[2]. Nele os autores comparam a quantidade de páginas similares resultantes de duas coletas: uma sem e outra com seu filtro de coleções duplicadas. A quantidade de páginas similares caiu de 48% na primeira coleta para 13% na segunda coleta, onde o filtro foi aplicado.

Em [da Costa Carvalho et al., 2007] os autores se basearam num dos métodos apresentados por Bharat et al. [2000] para propor um novo método capaz de utilizar eficientemente informações acerca do conteúdo das páginas. Esse método é chamado de NormPaths. O NormPaths usa a norma da página como uma assinatura para seu conteúdo. O uso da norma não afeta o desempenho do método, pois essa informação já se encontra disponível nas bases das máquinas de busca como um subproduto do processo de indexação de documentos.

Devido ao uso de informações sobre o conteúdo dos sítios, o método proposto é capaz de alcançar níveis de precisão maiores do que os métodos anteriores sem, con-

[2]http://www.google.com

tudo, consumir recursos computacionais adicionais. Segundo os autores, o NormPaths conseguiu uma melhoria de 47,23% na métrica F1 comparado com métodos propostos anteriormente. Por se tratar do estado-da-arte em detecção de réplicas de sítios web, o NormPaths será usado como *baseline* neste trabalho.

Não foi possível encontrar na literatura um trabalho anterior a este que aplicasse técnicas de aprendizado de máquina ao problema de detecção de réplicas de sítios web. Porém, aprendizado de máquina tem sido aplicado com sucesso em várias outras áreas. Por exemplo, em [Hajishirzi et al., 2010] os autores apresentam um método para detecção de quase duplicatas de páginas web que pode ser facilmente configurado para um domínio em particular.

Essa facilidade de configuração se deve ao uso de técnicas de aprendizado de máquina para aprender características a partir de coleções rotuladas de documentos, de forma otimizar o método para o domínio em questão. Os autores relatam que o valor máximo obtido para a métrica F1 foi 0,953 a partir de uma coleção rotulada de 1005 documentos, porém, com apenas 60 documentos, o método foi capaz de alcançar 0,856 de F1.

1.3 Contribuições

O principal resultado deste trabalho é a apresentação de um novo algoritmo, baseado em técnicas de aprendizado de máquina, para a detecção de pares de sítios web candidatos a réplica. O uso de aprendizado de máquina é uma abordagem inédita nesse contexto uma vez que os métodos existentes na literatura até então se baseiam apenas no uso de heurísticas para selecionar pares de sítios candidatos a réplica dentro das bases das máquinas de busca. Podemos citar como contribuições específicas as seguintes:

1. O algoritmo DREAM (Capítulo 3), que provê ganhos significativos em relação ao estado-da-arte [da Costa Carvalho et al., 2007].

2. Um estudo comparativo de várias técnicas de aprendizado de máquina aplicadas ao problema de detecção de réplicas de sítios web (Seção 4.7).

3. A proposição de novas evidências de replicação de sítios para o problema (Seções 3.2.3 e 3.2.4).

4. A proposição de modificações ao validador automático proposto por Bharat et al. [2000], que o habilitam a lidar com as novas características da Web (Seção 4.3).

5. Um estudo sobre classes de sítios que possuem propriedades capazes de afetar o desempenho de sistemas de detecção de réplicas de sítio web (Seção 4.2).

6. A criação de uma nova coleção de dados para detecção de réplicas de sítios web a qual facilitará a comparação com nossos resultados e que também pode ajudar no avanço dessa área (Seção 4.1).

7. A avaliação do algoritmo DREAM, utilizando algoritmos de aprendizado de máquina, sobre a coleção criada. Os resultados mostram que o algoritmo DREAM, utilizando o algoritmo de aprendizado de máquina *Random Forest*, supera em 35.1% o melhor método da literatura em termos da métrica F1 (Capítulo 4).

1.4 Organização da Dissertação

Esta dissertação está organizada da seguinte forma: no Capítulo 2 são introduzidos os conceitos básicos referentes ao problema de detecção de réplicas de sítios web. No Capítulo 3 descrevemos o algoritmo proposto e apresentamos as características utilizadas. O Capítulo 4 apresenta e discute o estudo realizado, bem como seus resultados. Finalmente, no Capítulo 5, apresentamos as conclusões e as direções vislumbradas como possíveis trabalhos futuros sobre esse assunto.

Capítulo 2

Conceitos Básicos

O objetivo deste capítulo é apresentar conceitos básicos a serem utilizados no restante do trabalho. Entre eles, os algoritmos de aprendizado de máquina estudados, os componentes de uma máquina de busca e a estrutura da URL.

2.1 Máquinas de Busca

Uma máquina de busca é a aplicação prática de técnicas de recuperação de informação em grandes coleções de texto [Croft et al., 2009]. Apesar de as máquinas de busca para Web serem as mais conhecidas, elas não são as únicas. Esse tipo de sistema está presente em muitas outras áreas e aplicações: computadores, *smartfones*, emails aplicações empresariais, entre outros. Quando usamos a funcionalidade *pesquisar* do Microsoft Windows 7, por exemplo, estamos usando na verdade uma máquina de busca que age sobre a coleção de documentos existente em nosso computador. No restante da dissertação focaremos nas máquinas de busca para Web, as quais iremos nos referir como máquinas de busca apenas.

Na prática, máquinas de busca comparam consultas com documentos e os organizam em listas ordenadas que são apresentadas aos usuários em forma de *ranking*, porém há muito mais por trás desse tipo de sistema do que apenas algoritmos de *ranking*. É preciso obter documentos úteis da Web, processá-los de forma eficiente, montar estruturas de dados otimizadas para vários tipos de consultas, processar consultas de forma quase instantânea e apresentá-las de forma clara e atraente ao usuário, entre outros. Para realizar todas essas tarefas é necessário haver uma série de componentes trabalhando em conjunto. Apresentamos a seguir os três principais.

2.1.1 Coletor Web

Embora a mais nova e sofisticada tecnologia aplicada a uma máquina de busca possa torná-la ainda melhor, é a informação de boa qualidade contida em seus documentos que a torna útil [Croft et al., 2009]. Por isso, o coletor de páginas da Web pode ser considerado o componente mais importante de uma máquina de busca.

O coletor Web é o componente responsável por encontrar e coletar páginas web automaticamente. Também é dele a tarefa de responder à questões como "quando e o que coletar". Porém, como responder a essas duas questões? Certamente, a melhor resposta para "o que" seria "tudo o que for útil pra alguém", e a melhor resposta para quando seria "sempre que algo novo aparecer ou for atualizado". Contudo, fazer com que um programa de computador saiba o que é útil e saiba quando um documento é criado ou atualizado na Web é uma tarefa difícil.

Com relação ao seu funcionamento, geralmente os coletores trabalham em ciclos compostos por três etapas:

i. *Fetching*: o sistema que recupera os documentos na Internet, chamado *fetcher*, recebe uma lista de URLs a serem coletadas, dispara várias requisições HTTP paralelas e armazena os documentos recuperados em um repositório de dados. No primeiro ciclo, o conjunto de URLs recebido é montado externamente ao coletor e é chamado de semente.

ii. Descobrimento de novas URLs: os documentos coletados são processados a fim de identificar e extrair seus apontadores, entre outras informações. Esses apontadores farão parte do conjunto de URLs conhecidas pela máquina de busca.

iii. Escalonamento: as informações extraídas na etapa anterior são usadas para escolher, dentre o conjunto de URLs conhecidas pela máquina de busca, quais URLs devem ser enviadas ao *fetcher* para serem coletadas. Isso é feito por um módulo chamado escalonador. Por fim, volta-se a etapa (i) e o ciclo continua.

O coletor deve ser seguro e robusto o bastante para lidar com os problemas encontrados na Web, pois ele é porta de entrada para tudo que vem dela. Entre esses problemas é possível citar: URLs mal formadas, documentos com estrutura inválida, informação replicada, sítios fraudulentos, vírus, entre outros. O sistema para detecção de réplicas de sítios web deve trabalhar próximo desse módulo, a fim de evitar que réplicas de um mesmo sítio continuem sendo coletadas.

2.1.2 Indexador

Uma vez que todos os documentos necessários foram coletados, é preciso capturar sua informação e transformá-la em um formato que o computador entenda e que seja possível pesquisar quase que instantaneamente. Para isso, as máquinas de busca utilizam estruturas de dados conhecidas como índices. Índices são muito comuns em nossas vidas: estão presentes no final dos livros, nas listas telefônicas, cardápios, bibliotecas e em muitos outros lugares.

Geralmente os índices são representados em máquinas de buscas como listas invertidas. Listas invertidas são um tipo de lista de tuplas ordenadas do tipo <*chave-valor*>, onde as *chaves* são termos (ou grupos deles) do vocabulário da coleção e os *valores* são listas de referências para documentos (e posições) onde os termos ocorrem na coleção. O responsável pela construção dessas listas é o indexador. Ele realiza a análise sintática dos documentos, extrai seus termos e atualiza as estruturas de dados pesquisadas durante o processamento das consultas.

Um dos principais desafios na construção do índice é o seu tamanho. Em coleções enormes como a Web, o elevado número de termos pode fazer com que seu processamento se torne muito difícil. Técnicas de redução do tamanho do índice, tais como compressão [Ziviani et al., 2000], podem ser aplicadas para facilitar o processamento de tais coleções. Outra forma é transformar o texto a fim de reduzir o número de termos distintos no vocabulário final. As transformações mais usadas são: transformação de maiúsculas em minúsculas, *stemming* e remoção de *stop words*. Um estudo aprofundado sobre criação e compressão de índices pode ser encontrado em [Witten et al., 1999].

2.1.3 Processador de Consultas

Uma vez que os documentos já foram coletados e indexados, a máquina de busca já é capaz de responder às consultas dos usuários. Assim, quando o usuário necessita de alguma informação na Web, ele submete à máquina de busca uma consulta em forma de texto que representa a informação a qual ele necessita. Essa consulta é então modificada por meio de transformações semelhantes àquelas aplicadas aos textos dos documentos no processo de indexação. Por fim, os documentos recuperados são ordenados de acordo com uma estimativa que tenta predizer sua relevância com relação à consulta submetida pelo usuário.

Um problema encontrado ao se processar consultas é predizer a relevância dos documentos. Isso porque quem decide o que é ou não relevante é o usuário e essa decisão varia de usuário para usuário. Por exemplo, alguém pode formular a consulta

"Brasília" procurando por informações sobre como chegar à capital do Brasil. Outro usuário, porém, pode formular a mesma consulta procurando por informações turísticas sobre a cidade, ou ainda pode haver um terceiro usuário que estaria interessado em informações sobre o automóvel de mesmo nome.

Outro problema que dificulta bastante o trabalho da máquina de busca é o tamanho da consulta que, em geral, gira em torno de apenas duas ou três palavras [Croft et al., 2009]. Você diria a uma pessoa "necessitar" se quisesse que a mesma lhe dissesse qual a regência verbal desse verbo? Pois é, essa é uma situação muito comum e que as máquinas de busca enfrentam milhares de vezes por dia: os usuários, geralmente, não estão dispostos a digitar grandes frases para explicar o que procuram.

Por esses e outros motivos, existem vários modelos de recuperação de informação que as máquinas de busca podem utilizar para tentar predizer a relevância dos documentos, escolhendo, por exemplo, aquele mais adequado à classe de consultas em questão. Podemos citar como os mais importantes o modelo Booleano [Wartick, 1992] (um dos primeiros), o modelo vetorial [Salton & Lesk, 1968; Salton & Yang, 1973] (considerado o mais popular) e o modelo probabilístico [Robertson & Jones, 1976]. O algoritmo BM25 [Robertson & Walker, 1994], da família dos probabilísticos, merece ser ressaltado por haver um consenso de que ele supera o modelo vetorial em coleções de documentos gerais. [Baeza-Yates & Ribeiro-Neto, 2011] é uma boa leitura para os interessados em saber mais sobre os modelos de recuperação de informação.

2.2 Aprendizado de Máquina

Mitchell [1997] define aprendizado de máquina como o estudo de algoritmos computacionais capazes de melhorarem automaticamente a execução de alguma tarefa através da experiência. Algoritmos de aprendizado de máquina se baseia principalmente em probabilidade e estatística para aprender padrões complexos a partir de algum *corpus* e usá-los na tomada "inteligente" de decisões sobre algum assunto.

A aplicabilidade desses algoritmos é vasta e interdisciplinar. Podemos citar como exemplos áreas que vão desde correção ortográfica [Schmid, 1994] até o diagnóstico de doenças [Bair & Tibshirani, 2003]. Além da aplicabilidade, o fato de que esse tipo de algoritmo tem sido aplicado com sucesso em várias tarefas de natureza diferenciada evidencia sua capacidade de render bons resultados. Lacerda et al. [2006] propuseram um arcabouço baseado em programação genética capaz de associar propagandas à páginas web, alcançando ganhos de 67% em termos de precisão média sobre o *baseline*. Hajishirzi et al. [2010] apresentam um método para detecção de quase duplicatas de páginas web que foi capaz de reduzir em 40% o trabalho de um coletor Web.

Os casos de sucesso de aprendizado de máquina nos levaram a acreditar que esse tipo de algoritmo tem potencial para melhorar os resultados do estado-da-arte em detecção de réplicas de sítios web, uma vez que não encontramos relatos de aplicação dessa abordagem nesse contexto. A seguir, descrevemos os princípios básicos e diferenças entre os métodos mais populares da literatura estudados neste trabalho.

2.2.1 Árvores de Decisão

Métodos de classificação por árvores de decisão modelam o relacionamento entre uma entrada x_i e uma saída y_i por meio de uma árvore. Cada nó interno da árvore corresponde a um atributo e cada conexão entre esse e um nó subsequente (ou nó filho) representa um valor possível para o atributo correspondente. Um nó folha representa uma saída predita para uma entrada cujos valores dos atributos são os representados pelo caminho que liga a raiz até a folha em questão. Dessa forma, um caminho na árvore pode ser interpretado como um conjunto de pares atributo-valor que leva a uma predição.

Algoritmos para construção de árvores de decisão geralmente seguem a estratégia recursiva dividir-para-conquistar [Quinlan, 1986, 1993], os quais dividem um nó em outros nós que virão a ser seus filhos. Esse processo é repetido de maneira recursiva em cada nó derivado até que a divisão não seja mais viável ou que a discriminação perfeita tenha sido atingida. Nesse processo, o passo crucial é a seleção de atributos, isto é, a qual atributo um dado nó interno da árvore deve corresponder.

Vários critérios para a seleção de atributos já foram propostos, entre eles, o ganho de informação é um dos mais populares. O ganho de informação mede a redução na incerteza em uma variável aleatória V dado o valor de uma segunda variável aleatória Q. Na prática o ganho de informação pode ser computado por meio de probabilidades empíricas, com V representando as saídas e Q representando os valores possíveis para um determinado atributo. O passo de seleção de atributos mencionado anteriormente é realizado testando o ganho de informação para cada atributo Q com as saídas V. Escolhe-se então aquele atributo que render o maior ganho de informação.

2.2.2 Combinações de Árvores

O classificador Combinações de Árvores (*Random Forest*) é uma combinação de classificadores baseados em árvores de decisão tal que cada árvore depende dos valores de um vetor aleatório, amostrado independentemente e com a mesma distribuição para todas as árvores na floresta. Para classificar um novo objeto, seu vetor de características é

apresentado a cada árvore na floresta. Cada árvore dá uma classificação, ou um voto para uma classe. A classificação final é dada pela classe que recebeu o maior número de votos entre todas as árvores da floresta.

Cada árvore é construída da seguinte forma:

1. Seja N o número de instâncias no conjunto de treino. Selecione aleatoriamente, com reposição, N instâncias do conjunto de dados original. Esse será o conjunto de treinamento utilizado na construção das árvores.

2. Seja M o número de variáveis de entrada. Especifique um número $m \ll M$ tal que, em cada nó, m variáveis sejam selecionadas aleatoriamente (a partir das M variáveis de entrada) e o nó seja subdividido usando a melhor subdivisão entre essas m variáveis.

3. Cada árvore cresce até a maior extensão possível. Não há poda.

Em árvores de decisão padrão, cada nó é dividido usando a melhor divisão entre todas as variáveis. No Random Forest, cada nó é dividido usando a melhor divisão entre um subconjunto de varáveis escolhido aleatoriamente naquele nó. Apesar de contra-intuitiva, essa estratégia funciona bem quando comparada a outros classificadores.

Em [Breiman, 2001], trabalho em que o Random Forest foi proposto, os autores mostram que a taxa de erro da floresta depende de duas propriedades:

- A correlação par a par entre as árvores na floresta. Aumentando a correlação aumenta-se a taxa de erro da floresta.

- A robustez de cada árvore individual na floresta. Uma árvore com uma baixa taxa de erro é um classificador robusto. Aumentando a robustez de cada árvore, reduz-se a taxa de erro da floresta.

Reduzindo o valor de m reduzem-se ambas, correlação e robustez. Aumentando o valor de m aumentam-se ambas, correlação e robustez. Esse é o único parâmetro ajustável ao qual o Ramdom Forest é, de alguma forma, sensível.

2.2.3 Máquinas de Vetor de Suporte (SVM)

Máquinas de Vetor de Suporte (Support Vector Machines - SVM) [Platt, 1999] consistem de métodos de espaço vetorial para problemas de classificação binária, isto é, onde existem apenas duas classes possíveis. A ideia por trás do SVM é encontrar um hiperplano de decisão que melhor separe os elementos das duas classes. Apesar de

ter sido concebido para resolver problemas binários (duas classes) onde as classes são separáveis por um hiperplano, o SVM pode ser estendido para problemas multi-classes e/ou de classes não separáveis linearmente.

Para tornar o conceito de "melhor separar" mais claro, é preciso definir o conceito de margem entre duas classes. A Figura 2.1 ilustra dois casos de escolha de hiperplano de decisão diferentes, embora resolvam o mesmo problema. As linhas contínuas representam os hiperplanos de decisão escolhidos enquanto que as linhas tracejadas, paralelas aos hiperplanos de decisão, representam os limites extremos de cada classe. As linhas tracejadas são chamadas de hiperplanos delimitadores. A distância entre os hiperplanos delimitadores é chamada de **margem** e representa o quanto o hiperplano de decisão pode ser movido sem causar erros.

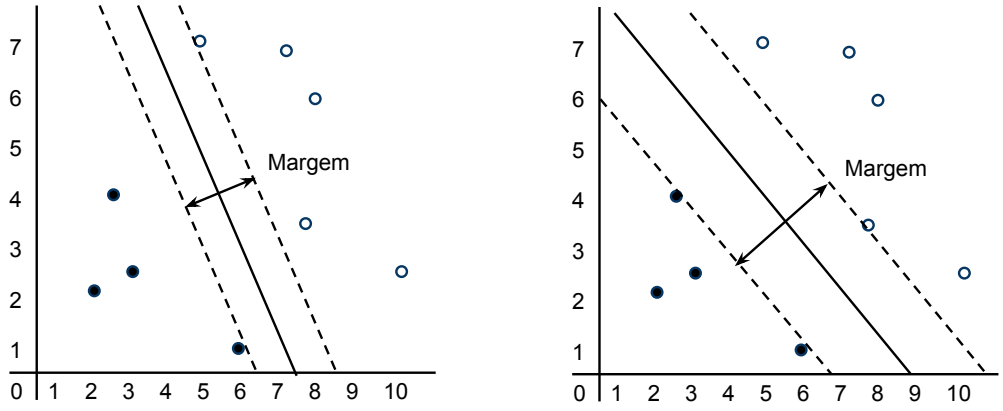

Figura 2.1. Exemplos de hiperplanos diferentes aplicados ao mesmo problema.

O gráfico da esquerda apresenta um hiperplano que rende uma margem menor enquanto que gráfico da direita apresenta o hiperplano que maximiza a margem. A ideia por trás do problema é que quanto maior a margem maior será a generalização. Dessa forma, o SVM resolve o problema de otimização de encontrar o hiperplano de decisão que maximiza a margem entre as instâncias das classes na coleção de treino.

Como afirmado anteriormente, o SVM pode ser estendido para resolver problemas não separáveis linearmente. Essa funcionalidade pode ser alcançada tanto por meio do uso de hiperplanos que permitem algum erro de treino [Cortes & Vapnik, 1995] quanto por modificações no algoritmo que mapeiam as entradas para um espaço de dimensionalidade mais alta [Aizerman et al., 1964], onde a instâncias do problema podem se tornar linearmente separáveis.

O SVM possui uma propriedade interessante que merece ser ressaltada: o hiperplano de decisão é determinado apenas pelas instâncias das classes que se sobrepõem aos hiperplanos delimitadores. Estas instâncias são chamadas de vetores de suporte.

Mesmo que todos os outros exemplos de treino sejam descartados, o mesmo hiperplano de decisão será obtido, gerando assim o mesmo modelo de classificação.

2.2.4 Classificador Associativo Sob Demanda (LAC)

O Classificador associativo sob demanda (Lazy Associative Classifier - LAC) é um tipo de classificador baseado em regras de associação que, como o próprio nome já diz, constrói seu modelo de classificação utilizando uma estratégia de extração sob demanda. Outros tipos de classificadores associativos extraem um mesmo conjunto global de regras de associação, a partir dos dados de treinamento, para ser aplicado a todas as instâncias de teste. Os classificadores baseados na estratégia LAC induzem um conjunto específico de regras para cada instância de teste. Esse processo, geralmente, reduz consideravelmente o número de regras geradas [Veloso, 2009].

A ideia por trás da classificação associativa sob demanda é a de que o problema pode ser decomposto em subproblemas mais simples, os quais podem ser resolvidos de forma independente. A decomposição do problema é alcançada encarando a classificação de cada instância x_i do conjunto de teste $\mathcal{T}$ como um problema independente. Assim, ao invés de gerar um único modelo de classificação f a ser utilizado para todas as instâncias x_i, o classificador gera um modelo específico f^{x_i} para cada x_i. Tal modelo mapeia x_i em uma classe c_i predita.

A decomposição se dá por meio de projeções do conjunto de treinamento $\mathcal{S}$ sobre uma instância $x_i \in \mathcal{T}$, denotadas por $\mathcal{S}^{x_i}$. Mais especificamente, sempre que uma entrada $x_i \in \mathcal{T}$ é processada, o classificador usa x_i como um filtro para remover de $\mathcal{S}$ atributos e exemplos que não são úteis para gerar o modelo de classificação para x_i. Uma vez que $\mathcal{S}^{x_i}$ contém apenas valores presentes em x_i, todas as regras de associação geradas a partir de $\mathcal{S}^{x_i}$ devem se encaixar em x_i. Dessa forma, apenas regras do tipo $\mathcal{X} \rightarrow c_i$, com $\mathcal{X} \subseteq x_i$ (x_i contém todos os valores de $\mathcal{X}$), poderão ser extraídas.

Outros tipos de classificadores associativos extraem suas regras a partir de grandes conjuntos de treinamento. Enquanto esse processo gera grandes conjuntos de regras globais, regras para instâncias de teste específicas podem não ser extraídas. O classificador LAC, por outro lado, foca sua extração de regras em um conjunto de treinamento muito menor, que é induzido pelos valores da própria instância de teste. O classificador LAC também é mais apropriado quando o conjunto de treino é complexo e há várias formas de generalizar um caso, uma vez que ele generaliza seus exemplos de treino exatamente da forma necessária para cobrir a instância de teste [Veloso, 2009].

Existem algumas variações do classificador LAC com relação à forma como a classe de saída é escolhida. A versão que usamos neste trabalho é o LAC-MR (LAC Multiple Rules). Essa versão do algoritmo se baseia em múltiplas regras para definir a

classe de saída. Cada regra contribui com um voto ponderado e a classe mais votada é a escolhida.

2.3 A Estrutura da URL

Cada documento na Web é identificado por uma URL (*Uniform Resource Locator*). Uma URL é única e identifica um único recurso. Ela é composta por três campos: método de acesso, nome do servidor e caminho. Por exemplo, na URL da Figura 2.2 a sequência de caracteres *http* define o método de acesso, *www.dcc.ufmg.br* é o nome do servidor e *pos/programa/historia.php* é o caminho.

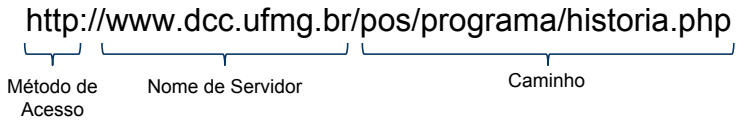

Figura 2.2. Os três campos da URL.

O nome do servidor identifica o servidor web no qual o documento está armazenado. Um servidor web guarda uma coleção de documentos os quais compartilham o mesmo nome de servidor. O nome de servidor é composto por um nome de domínio acrescido ou não de um prefixo. O nome de domínio identifica um conjunto de servidores enquanto que o prefixo identifica um servidor específico sob esse domínio. No exemplo da Figura 2.2, o nome de domínio é *ufmg.br* e o prefixo é *www.dcc*.

Cada documento é identificado pelo seu próprio caminho, que é único dentro do seu servidor. O caminho reflete a estrutura de diretórios do servidor web. Por exemplo, na URL da Figura 2.2 o documento *historia.php* está dentro do diretório *programa* que por sua vez está dentro do diretório *pos*. O número de diretórios de um caminho é chamado de nível do caminho e pode ser definido a partir do número de "/" (barras) do caminho.

2.4 Definição de Sítio Web

O conceito de sítio web não está claramente definido na literatura. Os trabalhos no tópico de detecção de réplicas de sítios web [Bharat & Broder, 1999; Bharat et al., 2000; Cho et al., 2000; da Costa Carvalho et al., 2007] têm usado a definição de que um sítio é o conjunto de páginas que compartilham um mesmo nome de servidor. Em [da Costa Carvalho et al., 2007], por exemplo, essa definição é adotada sob a

alegação de que ela é simples de usar e representa um bom equilíbrio entre conjuntos de páginas de alta granularidade e de baixa granularidade. Um conjunto de páginas de alta granularidade poderia ser obtido assumindo que cada nome de domínio constitui um sítio diferente. Por outro lado, um conjunto de páginas de baixa granularidade poderia ser obtido considerando que o sítio de uma página web é dado pelo nome de servidor mais um pedaço de seu caminho.

Portanto, em concordância com a literatura de detecção de réplicas de sítios web, adotamos a seguinte definição para descrever um sítio:

Definição 1. *Um sítio web é o conjunto de páginas web que compartilham um mesmo nome de servidor.*

2.5 Definição de Réplicas de Sítios Web

É difícil obter uma definição clara de o que é réplica quando se trata de sítios web. Por um lodo, pode-se dizer que dois sítios são réplicas se seu conteúdo é idêntico, mas na prática essa definição é restritiva de mais. Por exemplo, mesmo que a URL www.waz.com.br seja coletada duas vezes e ao mesmo tempo o conteúdo coletado irá diferir. Esse fato não é raro e ocorre devido à presença de conteúdo dinâmico nas páginas, tais como anúncios, identificadores de seção, entre outros. Por outro lado, pode-se dizer que dois sítios são réplicas se um número suficiente de páginas de um deles possui conteúdo similar ao conteúdo de páginas do outro sítio. Porém, essa definição pode gerar problemas por não tratar de casos onde a estrutura dos sítios difere. Considere, por exemplo, os sítios www.cifraclub.com.br e www.cifras.com.br, dois dos maiores sítios de letras e cifras de músicas no Brasil. Eles apresentam uma grande quantidade de páginas com conteúdo similar (letras ou cifras das mesmas músicas), entretanto os dois sítios dificilmente podem ser chamados de réplicas. Em [Bharat & Broder, 1999] os autores propõem a seguinte definição de réplicas de sítio web:

Definição 2. *Dois sítios A e B são ditos serem réplicas se (i) uma alta porcentagem dos caminhos de A são válidos em B e vice-versa, e (ii) esses caminhos comuns designam documentos com conteúdo similar entre os dois sítios.*

Essa é a definição que tem sido adotada nos trabalho que tratam de replicação de sítio web e, por isso, é a definição que adotamos neste trabalho.

2.6 Validação de Pares Candidatos a Réplica

Marcar como réplica um sítio e eliminá-lo das respostas das máquinas de busca exige certeza no julgamento de que o sítio em questão é, de fato, uma réplica. A remoção de um sítio não replicado das respostas das máquinas de busca causa a perda de informação de qualidade que poderia ser de interesse do usuário. O ideal seria que, antes de remover um sítio candidato a réplica das respostas, um avaliador humano validasse o sítio em questão a fim de evitar a perda de informação. Porém, a validação manual não é viável na prática, devido ao esforço humano exigido.

Para contornar esse problema, Bharat et al. [2000] propuseram um método efetivo para a validação de pares candidatos, também adotado por da Costa Carvalho et al. [2007]. Dado um par de sítios (s_1, s_2), o método proposto verifica se uma parte dos caminhos de s_1 é válida em s_2 e vice-versa, e verifica se os conteúdos das páginas designadas por esses caminhos comuns são similares. Primeiramente seleciona-se aleatoriamente uma amostra dos caminhos de s_1 e, da mesma forma, uma amostra dos caminhos de s_2. Em seguida, cada caminho selecionado é coletado paralelamente nos dois sítios: s_1 e s_2. Se os pares de conteúdos coletados nos dois sítios são similares, o par é considerado réplica.

A similaridade entre as páginas é obtida por meio da métrica *resemblance* [Broder, 1997] que consiste em medir qual é a proporção de conteúdo textual igual entre os documentos comparados. Para tanto, cada texto é dividido em *k-shingles* (conjuntos de tamanho fixo de palavras). Por exemplo, na frase "detectar sítios replicados é importante" há três *3-shigles*: "detectar sítios replicados", "sítios replicados é" e "replicados é importante". Dados dois documentos $d1$ e $d2$ e seus respectivos conjuntos de *k-shingles* $S(d1)$ e $S(d2)$, a similaridade entre eles é dada pela Equação 2.1.

$$r(d_1, d_2) = \frac{|S(d_1) \cap S(d_2)|}{|S(d_1) \cup S(d_2)|} \tag{2.1}$$

Apesar de determinar com bastante eficácia se dois sítios são ou não réplicas, o método de validação automática não pode ser considerado uma solução para o problema de detectar pares de sítios replicados em bases de máquinas de busca. Isso porque esse método é demasiadamente caro, devido à necessidade de coleta extra de páginas e comparações de conteúdos textuais. Dessa forma, é muito importante se obter, do algoritmo de seleção de pares candidatos a réplica, uma lista precisa de candidatos a fim de reduzir o número de pares a serem validados.

Capítulo 3

Algoritmo Proposto para Detecção de Réplicas

Este capítulo apresenta um novo algoritmo para detectar pares de sítios web possivelmente replicados em bases de documentos de máquinas de busca, chamado DREAM (Detecção de RÉplicas usando Aprendizado de Máquina). O algoritmo DREAM faz uso de técnicas de aprendizado de máquina para combinar várias características obtidas da base de documentos com o objetivo de maximizar a efetividade na tarefa de detecção de réplicas.

Antes de passarmos à descrição do algoritmo DREAM vamos repetir a seguir o que queremos dizer por sítios replicados, conforme discussão apresentada na Seção 2.5. Dois sítios são réplicas se: (i) uma alta porcentagem dos caminhos (isto é, as partes da URL depois do nome do servidor) são válidos em ambos os sítios web, e (ii) esses caminhos comuns apontam para documentos que têm conteúdo similar. Os principais componentes que constituem uma URL são discutidos na Seção 2.3.

3.1 Descrição do Algoritmo DREAM

A partir da relação de sítios de uma base de uma máquina de busca, a tarefa de detectar sítios replicados pode ser executada por meio de um algoritmo ingênuo que utiliza força bruta, comparando todos os pares de sítios na base. Para uma base contendo s sítios, se cada sítio é comparado com todos os outros, então o algoritmo tem complexidade $O(s^2)$. Entretanto, o tamanho do problema a ser resolvido torna proibitivo o uso de algoritmos de complexidade quadrática como o que acabamos de descrever. Por exemplo, a coleção original que usamos nos experimentos possui 2.002.955 sítios sendo o número de pares a serem avaliados igual a $\frac{(n^2-n)}{2} = 2.005.913.364.535$, o que já tornaria proibitiva a solução por força bruta.

O algoritmo DREAM resolve o problema apresentado pela solução ingênua por meio de um processo de redução do número de pares candidatos a réplicas. O algoritmo DREAM é constituído de duas fases:

1. Uma primeira fase onde características mais gerais são utilizadas como uma espécie de filtro para selecionar um conjunto de pares de sítios que tem algum potencial para serem réplicas.

2. Uma segunda fase onde características mais específicas são obtidas para o conjunto de pares gerados na fase anterior e submetidas ao modelo aprendido pelos algoritmos de aprendizado de máquina, obtendo uma lista refinada de pares de sítios candidatos a réplica.

O Programa 1 apresenta o primeiro refinamento do algoritmo DREAM. Inicialmente, o algoritmo havia sido projetado para executar em apenas uma fase: a aplicação de modelos de aprendizado de máquina diretamente sobre o conjunto de todos os pares de sítios possíveis. Contudo, o desbalanceamento entre o número de réplicas e não réplicas nesse conjunto prejudica o treinamento dos algoritmos, o que foi resolvido por uma filtragem prévia de pares de não réplicas.

Programa 1 DREAM (Detecção de RÉplicas usando Aprendizado de Máquina).

Entrada: base de dados B de uma máquina de busca.
Saída: conjunto C de pares de sítios candidatos a serem réplicas.

{–***1ª Fase:*** *Seleção de pares candidatos a serem réplicas*–}
1: **para todo** par de sítios (s_1, s_2) em B **faça**
2: extraia o conjunto básico de características $\mathcal{F}_b$ de s_1 e s_2
3: **se** s_1 e s_2 possuem alguma das características $\mathcal{F}_b$ em comum **então**
4: insira (s_1, s_2) no conjunto C de candidatos a réplica
{–***2ª Fase:*** *Refinamento do conjunto de pares candidatos a serem réplicas*–}
5: **para todo** par de sítios (s_1, s_2) em C **faça**
6: extraia o conjunto de características para refinamento $\mathcal{F}_r$ de s_1 e s_2
7: **se** o algoritmo de aprendizado de máquina diz que $\mathcal{F}_r$ corresponde a um par não replicado **então**
8: remova (s_1, s_2) de C

Na fase de seleção de pares candidatos a serem réplicas um conjunto básico de características é utilizado como uma espécie de filtro sobre toda a base da máquina de busca. Esse conjunto possui duas propriedades que o tornam adequado a filtragem inicial sobre uma grande quantidade de pares: garante uma alta revocação e possui baixo custo de obtenção. Durante a fase de seleção de candidatos a réplica, todo par

de sítios que compartilhe alguma característica do conjunto básico é adicionado a um conjunto C de candidatos a réplica.

Na fase de refinamento do conjunto de pares candidatos a serem réplicas, algoritmos de aprendizado de máquina baseados em um novo conjunto de características refinam os pares do conjunto C selecionados na fase anterior. Chamaremos esse novo conjunto de conjunto de refinamento. Inicialmente, os algoritmos de aprendizado de máquina são treinados a partir de uma base marcada, isto é, cujos pares estão classificados como réplica ou não réplica. Mais detalhes sobre o treinamento dos algoritmos serão discutidos na Seção 3.3.2.

Ao final da segunda fase o conjunto C de pares candidatos a réplica é reduzido significativamente, restando apenas os pares que tenham uma probabilidade maior de serem réplicas. O processo de detecção de réplicas se completa com a aplicação do método de validação automática a todos os pares do conjunto refinado C retornados pelo algoritmo DREAM. Esse método é descrito na Seção 2.6. A coleta extra de páginas e comparação entre conteúdos textuais tornam cara a validação automática, daí a importância da redução do conjunto de pares candidatos a serem réplicas.

Uma etapa importante nas duas fases do algoritmo DREAM está relacionada com a extração de características a partir da base de documentos da máquina de busca. Consequentemente, na seção seguinte vamos apresentar as características utilizadas antes de apresentar um refinamento de cada uma das fases.

3.2 Características Utilizadas

Esta seção trata da descrição e da obtenção da características utilizadas pelo algoritmo DREAM apresentado no Programa 1. O fato de o algoritmo fazer uso de técnicas de aprendizado de máquina possibilita um melhor aproveitamento da combinação das capacidades discriminativas das características, baseando-se em propriedades aprendidas do próprio problema.

A seguir são apresentadas as características utilizadas como evidência de replicação neste trabalho, algumas delas propostas por nós e outras já existentes na literatura. As duas primeiras características apresentadas, caminho da URL e assinatura de conteúdo, compõem o conjunto básico de características utilizado na primeira fase do algoritmo DREAM. As sete características restantes fazem parte do conjunto de refinamento, utilizado na segunda fase do algoritmo.

3.2.1 Caminho da URL

O caminho de uma URL pode ser baseada em fragmentos [Bharat & Broder, 1999] ou no caminho completo [Bharat et al., 2000]. Neste trabalho propomos usar diretamente o caminho completo. Nesse caso um par de sítios é considerado ter algum potencial para ser réplica se eles possuem pelo menos um caminho em comum.

3.2.2 Assinatura do Conteúdo

Neste trabalho utilizamos uma função *hash* para obter a assinatura para o conteúdo das páginas dos sítios. Na máquina de busca utilizada para obter a coleção que utilizamos nos experimentos, a assinatura de cada página é gerada na fase de coleta. Um dado par de sítios é selecionado como candidato a réplica se seus sítios possuem pelo menos uma página em comum com a mesma assinatura. Outros tipos de assinaturas também podem ser utilizados. O trabalho de da Costa Carvalho et al. [2007], por exemplo, utiliza a norma das páginas ao invés de uma função *hash*.

3.2.3 Distância de Edição

A distância de edição [Levenshtein, 1965, 1966] é uma das novas características (no contexto de detecção de réplicas sítios) que estamos propondo. Ela é dada pelo número mínimo de operações de remoção, inserção ou substituição necessárias para transformar uma cadeia de caracteres em outra. Por exemplo, www.esporte.ce.gov.br e www.**s**esporte.ce.gov.br se referem ao mesmo sítio, que pertence à secretaria de esporte do Governo do Estado do Ceará, e sua distância de edição é um, relativa a remoção (ou inserção) do caractere *s* antes de *esporte*.

3.2.4 Diferença de Segmentos

Outra nova característica que propomos é a diferença de segmentos, que computa a probabilidade de um par de sítios ser réplica com base na ausência ou troca de segmentos em um dos nomes de servidor. Por exemplo, os nomes de servidor www.ippex.com.br e www.ippex.org.br diferem apenas pela troca entre os segmentos *com* e *org*. Cada evento de troca ou ausência de segmentos específicos recebe uma probabilidade de replicação, computada a partir da base de treinamento. Tal probabilidade é dada pela relação entre número de vezes em que o evento ocorreu e em quantas das ocorrências o par foi marcado como réplica.

3.2.5 Correspondência de Nomes de Servidor

A característica correspondência de nomes de servidor, proposta por Bharat et al. [2000], representa cada nome de servidor como um vetor de termos. Neste trabalho, os termos de um nome de servidor são seus segmentos. Por exemplo, o nome de servidor www.latin.dcc.ufmg.br contém os termos www, latin, dcc, ufmg e br. O peso de um termo t é dado pela Equação 3.1, onde $len(t)$ é número de caracteres de t e $df(t)$ é número de nomes de servidor que possuem o termo t.

$$peso(t) = \frac{\log(len(t))}{1 + \log(df(t))} \tag{3.1}$$

Dados dois sítios A e B e seus respectivos vetores de termos $\vec{a}$ e $\vec{b}$, a correspondência entre seus nomes de servidor é dada pelo cosseno do ângulo entre $\vec{a}$ e $\vec{b}$, como descrito na Equação 3.2.

$$cosseno(\vec{a}, \vec{b}) = \frac{\vec{a} \bullet \vec{b}}{|\vec{a}| \times |\vec{b}|} \tag{3.2}$$

Essa é uma das características que necessitam de dados pré-processados para serem utilizadas. Os dados necessários são as frequências $df(t)$ de todos os termos.

3.2.6 Quatro Octetos

Essa característica, proposta por Bharat et al. [2000], se baseia nos quatro octetos do endereço IP (IP4) e também precisa de dados pré-processados para seu funcionamento. Nesse caso, os dados necessários são os IPs de todos os sítios da base da máquina de busca. Após obter esses dados, sítios com mesmo endereço IP são agrupados. Dados dois sítios A e B, o valor da característica é dado pela Equação 3.3, onde $\mathcal{G}$ é um grupamento qualquer de sítios. A ideia é que quanto maior o grupo, menor a possibilidade de os sítios do grupo serem réplicas.

$$ip(A, B) = \begin{cases} \frac{1}{|\mathcal{G}|-1} & \text{se A e B pertencem a um mesmo grupo } \mathcal{G} \\ 0 & \text{caso contrário} \end{cases} \tag{3.3}$$

3.2.7 Três Octetos

A característica três octetos (IP3), proposta por Bharat et al. [2000], difere da característica quatro octetos (IP4) por se basear nos três primeiros octetos do endereço IP. Obtidos os endereços IP de todos os sítios da base da máquina de busca, os sí-

tios que possuírem os três primeiros octetos idênticos são agrupados. O valor dessa característica também é dado pelo Equação 3.3.

3.2.8 Correspondência entre Caminhos Completos

A característica Correspondência entre Caminhos Completos, proposta por [Bharat et al., 2000], representa cada sítio web como um vetor de termos, onde os caminhos completos das páginas do sítio representam os termos do vetor. O peso de um termo t é dado pela Equação 3.4, onde $df(t)$ é o número de sítios que contêm t e $maxdf$ é o maior $df(t)$ entre todos o termos da coleção. Em nossos experimentos todos os termos com frequência acima de 100 são descartados, assim $maxdf$ é efetivamente 100.

$$peso(t) = 1 + \log\left(\frac{maxdf}{df(t)}\right) \tag{3.4}$$

Dados dois sítios A e B e seus respectivos vetores de termos $\vec{a}$ e $\vec{b}$, o valor da Correspondência entre Caminhos Completos entre eles é dado pela Equação 3.2. As informações df e $maxdf$ devem ser obtidas antes que a característica possa ser utilizada.

3.2.9 Caminho e Conteúdo

A característica Caminho e Conteúdo, proposta por [da Costa Carvalho et al., 2007], consiste de uma pontuação entre um par de sítios: quanto maior essa pontuação, maiores as chances de replicação. Antes de utilizar esta característica é necessário extrair, de cada página da coleção, seu caminho e uma assinatura para seu conteúdo. Cria-se então uma tupla no formato <caminho, assinatura> e, para cada tupla, é montada uma lista L de todos os sítios que possuem uma página com aquela tupla.

Dado um par de sítios A e B da base da máquina de busca, a característica Caminho e Conteúdo para ele é dada pela Equação 3.5, onde SL é o conjunto de todas as listas L e $|L|$ é número de sítios na lista L.

$$rsim(A, B) = \sum_{\forall L \in SL | A \in L, B \in L} \frac{1}{|L|} \tag{3.5}$$

3.3 Refinamento das Duas Fases do Algoritmo DREAM

A partir do conhecimento das características utilizadas no algoritmo DREAM, podemos passar ao refinamento de suas duas fases. Antes disso, é importante relembrar

resumidamente as nove características discutidas nas seções anteriores e suas propriedades. A Tabela 3.1 apresenta, para cada característica, seu nome, custo de utilização e em qual fase do algoritmo DREAM ela é empregada.

Característica	Custo	Fase
Caminho da URL	Baixo	1ª Fase
Assinatura do conteúdo	Baixo	1ª Fase
Distância de edição	Baixo	2ª Fase
Diferença de segmentos	Baixo	2ª Fase
Correlação nomes de servidor	Alto	2ª Fase
Quatro octetos	Alto	2ª Fase
Três octetos	Alto	2ª Fase
Correlação entre caminhos completos	Alto	2ª Fase
Caminho e conteúdo	Alto	2ª Fase

Tabela 3.1. Lista das características utilizadas no algoritmo DREAM, seus custos e fase onde é empregada.

3.3.1 Seleção de Pares Candidatos

A maior parte dos pares de sítios nas bases das máquinas de busca é formada por sítios que não são réplicas. O objetivo da fase de seleção de pares candidatos é remover do conjunto inicial pares de sítios que não tem nenhum potencial para serem réplicas. Isso é feito por meio da aplicação de filtros baseados no conjunto básico de características, que é composto pela característica caminho da URL e assinatura do conteúdo.

Tipicamente, encontrar pares de sítios replicados em bases de máquinas de busca é um problema quadrático, uma vez que é necessário comparar todos os sítios entre si. Porém, a forma como o algoritmo de seleção de pares candidatos foi construído torna o custo desse processo linear no número de páginas na base da máquina de busca. O Programa 2 mostra um primeiro refinamento do algoritmo em questão.

Programa 2 Seleção de pares de sítios candidatos a serem réplica - 1º refinamento.

Entrada: base de dados B de uma máquina de busca.
Saída: conjunto C de pares de sítios candidatos a réplica.

1: **para todo** documento d_i em B **faça**
2: extraia o conjunto básico de características $\mathcal{F}_b$ de d_i
3: adicione o sítio de d_i as listas L_j de sítios relacionados correspondentes as características $\mathcal{F}_b$
4: **para todo** Lista L_j **faça**
5: monte todos os pares de sítio possíveis e os guarde no conjunto C de candidatos a réplica

A complexidade linear é alcançada ao se fazer apenas uma varredura na base de entrada, criando-se listas de sítios relacionados, isto é, que possuam ao menos uma página com determinada característica do conjunto básico em comum. O primeiro passo é, para cada página da base, inserir seu sítio nas listas correspondentes a suas características. O segundo passo é montar, para cada lista de sítios relacionados, todos os pares possíveis e, com isso, se obter o conjunto dos pares de sítios candidatos a réplica.

Um ponto importante dessa fase é a estrutura de dados usada para armazenar os sítios relacionados, ilustrada na Figura 3.1. A estrutura é composta por uma tabela de valores para uma determinada característica. A cada entrada da tabela está ligada uma lista de sítios relacionados que compartilham alguma página com aquele valor para aquela característica. É importante ressaltar que é mantida uma estrutura como a da Figura 3.1 para cada característica do conjunto básico.

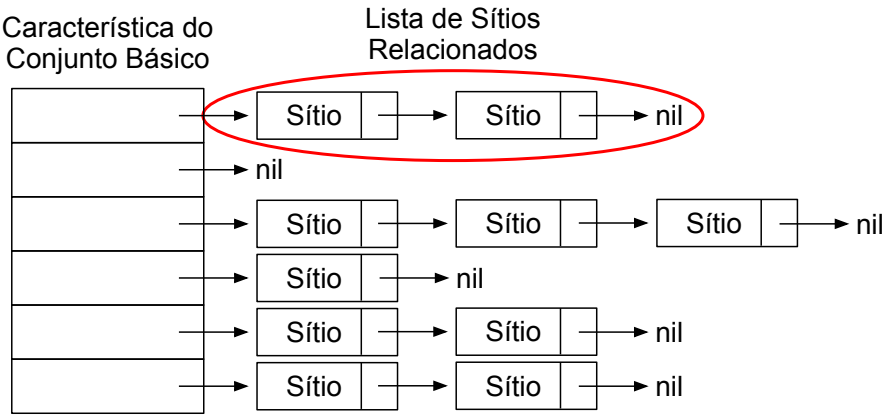

Figura 3.1. Estrutura de dados da Tabela de Sítios Relacionados.

Após a apresentação da estrutura de dados usada, é possível obter-se um novo refinamento para o algoritmo de seleção de pares candidatos, apresentado no Programa 3. O novo refinamento torna claro que é mantida uma tabela de listas de sítios relacionados para cada característica do conjunto básico. Vale ressaltar que a varredura da base é orientada a documentos (páginas web) e não a pares de sítios.

3.3.2 Refinamento do Conjunto de Pares Candidatos

O conjunto de pares candidatos a serem réplicas obtido pela fase anterior não é preciso o bastante para que o validador automático seja efetivamente aplicado a ele. Entretanto, o desbalanceamento entre o número de pares de réplicas e de não réplicas no conjunto atual já não prejudica o treinamento dos algoritmos de aprendizado de má-

Programa 3 Seleção de pares de sítios candidatos a serem réplica - 2º refinamento.

Entrada: base de dados B de uma máquina de busca.
Saída: conjunto C de pares de sítios candidatos a réplica.

```
1: para todo característica f_j do conjunto básico F_b faça
2:    inicialize uma tabela T_j de listas L de sítios relacionados
3: para todo documento d_i em B faça
4:    para todo característica f_j do conjunto básico F_b faça
5:       extraia f_j de d_i
6:       encontre em T_j a lista L correspondente ao valor de f_j em d_i
7:       insira o sítio de d_i em L, se ele ainda não existir lá
8: para todo tabela T_j faça
9:    para todo lista L em T_j faça
10:      para todo sítio s_k em L faça
11:         para todo sítio s_l diferente de s_k em L faça
12:            adicione o par (s_k, s_l) ao conjunto C de candidatos a réplica
```

quina. Além disso, o número atual de pares candidatos já viabiliza o pré-processamento de dados exigidos para a utilização de características que precisam de informações pré-processadas para seu funcionamento. Dessa forma, o conjunto de pares candidatos pode ser refinado por um processo onde algoritmos de aprendizado de máquina baseados em um conjunto de novas características, chamado de conjunto de refinamento, removem aqueles pares com baixa probabilidade de serem réplicas.

O Programa 4 descreve o algoritmo para o refinamento do conjunto de pares candidatos. O primeiro passo é processar os dados das características que necessitam de informação pré-processada. De posse desses dados, todas as características já podem ser utilizadas. Inicia-se então o processo de refinamento propriamente dito. Nele, são obtidos os valores das características do conjunto de refinamento para cada par de sítios candidatos. Em seguida, com base nesses valores, o algoritmo de aprendizado de máquina, já com um modelo aprendido, fornece uma probabilidade de o par ser ou não réplica. Se essa probabilidade estiver abaixo de um limite pré-estabelecido, o par de sítios deve ser removido do conjunto dos pares candidatos a réplica.

Antes que os algoritmos de aprendizado de máquina possam ser utilizados na fase de refinamento do conjunto dos pares candidatos a réplica eles devem ser treinados a partir de um gabarito. O gabarito é uma coleção de pares de sítios web onde os pares de réplicas são conhecidos. As informações necessárias para o treinamento dos algoritmos são as características do conjunto de refinamento, referentes de cada par do gabarito, além de sua classe. Uma vez obtido, o mesmo gabarito pode ser usado para o treinamento de vários algoritmos. A Seção 4.1 descreve a coleção de dados e criação do gabarito utilizado nos experimentos realizados neste trabalho.

Programa 4 Refinamento do conjunto de pares candidatos a réplica.

Entrada: conjunto C de pares de sítios candidatos a réplica.
Saída: conjunto reduzido C de pares de sítios candidatos a réplica.

1: **seja** $\mathcal{F}_r$ o conjunto das características para refinamento
2: **seja** f^p uma característica de $\mathcal{F}_r$ que precisa de dados pré-processados
3: **para todo** sítio s_i em C **faça**
4: **para todo** característica f_j^p em $\mathcal{F}_r$ **faça**
5: processe os dados necessário para a utilização de f_j^p em s_i
6: **para todo** par p_i em C **faça**
7: **para todo** característica f_j em $\mathcal{F}_r$ **faça**
8: obtenha o valor de f_j para p_i
9: **seja** $\mathcal{V}$ o conjunto dos valores das características em $\mathcal{F}_r$
10: com base em $\mathcal{V}$, obtenha, do algoritmo de aprendizado de máquina, a probabilidade pr de p_i ser réplica
11: **se** $pr <$ probabilidade mínima **então**
12: remova p_i de C

Capítulo 4

Resultados Experimentais

Este capítulo apresenta os resultados experimentais referentes à utilização do algoritmo DREAM. A Seção 4.1 descreve a coleção de páginas web utilizada nos experimentos. A Seção 4.2 apresenta um estudo sobre classes de sítios web que afetam o desempenho de sistemas de detecção de réplicas. A Seção 4.3 discute modificações aplicadas ao validador automático descrito na Seção 2.6. As métricas utilizadas na avaliação dos algoritmos são apresentadas na Seção 4.4. A Seção 4.5 discute a parametrização automática dos algoritmos. Finalmente, a Seção 4.7 apresenta uma comparação entre os resultados obtidos pelos algoritmos.

4.1 Coleção de Dados

Um dos desafios enfrentados durante o desenvolvimento deste trabalho foi o fato de não existirem coleções públicas de dados com informações sobre replicação de sítios web. A falta de coleções para treinar e avaliar algoritmos nos levou a construir uma coleção própria, a WBR2010B, e seu conjunto de informações sobre replicação de sítios.

A WBR2010B é uma coleção de mais de 150 milhões de páginas web, obtidas a partir da coleta de uma máquina de busca real brasileira. Seu escopo está restringido a páginas de sítios brasileiros (nomes de domínio terminados em .br) com conteúdo textual. Sua coleta aconteceu entre 22 de setembro e 05 de outubro de 2010, porém a duração efetiva foi de sete dias. Essa coleta foi realizada sem qualquer filtro ou tratamento de informação de má qualidade, o que permite encontrar na coleção todos os problemas presentes na Web real, incluindo replicação de informação. Por se tratar de um retrato fiel da Web, a WBR2010B é bastante adequada ao estudo de replicação e, por isso, foi escolhida para nossos experimentos. A Tabela 4.1 resume as informações sobre a WBR2010B.

Sítios conhecidos		
Total	2.002.955	
Média de páginas conhecidas	188,65	
Média de páginas coletadas	94,54	
URLs		
Conhecidas	377.856.944	
Coletadas	189.355.270	50,11%
Não coletadas	188.501.674	49,89%
URLs Coletadas com Sucesso (79,74%)		
Total	150.984.371	
Estáticas	62.465.536	41,37%
Dinâmicas	88.518.835	58,63%
URLs com Falhas (20,26%)		
Total	38.370.899	
Timeouts	19.320.889	50,35%
Erros HTTP (código >= 400)	11.009.752	28,69%
Errors gerais	8.040.258	20,95%
Coleta		
Período	de 22/09/10 a 05/10/10	
Duração efetiva	7 dias	

Tabela 4.1. Resumo das informações a respeito da WBR2010B.

4.1.1 Geração do Gabarito

Algoritmos de aprendizado de máquina exigem uma etapa de treinamento para que possam ser utilizados. Esse treinamento deve ser realizado sobre um gabarito, isto é, uma coleção de pares de sítios onde as réplicas sejam conhecidas. A forma ingênua de obter esse gabarito seria aplicando o validador automático, descrito na Seção 2.6, diretamente sobre os pares da coleção. Porém, para os mais de 2 milhões de sítios da WBR2010B, o número total de pares possíveis ultrapassa a marca dos 2×10^{12} pares.

Desta forma, optamos por eliminar do conjunto de pares possíveis aqueles que não têm potencial nenhum para serem réplicas, de forma a reduzir os custos do processo de validação automática. A eliminação foi feita por meio da aplicação da primeira fase do algoritmo DREAM sobre os sítios da coleção. Note que não há problemas se algum par de réplicas for eliminado nesse processo, pois o principal objetivo deste trabalho não é encontrar todos os pares de réplicas da WBR2010B, e sim avaliar técnicas de aprendizado de máquina como solução alternativa para o problema da replicação de sítios.

Além disso, o *baseline* (NormPaths) não foi prejudicado, pois, assim como a primeira fase do algoritmo DREAM, ele também se baseia no conjunto básico de ca-

racterísticas para selecionar sua lista de pares candidatos. O NormPaths seleciona pares de sítios que compartilham páginas com caminho idêntico **e** conteúdo similar, ao passo que a primeira fase do algoritmo DREAM seleciona pares de sítios que compartilham páginas com caminho idêntico **ou** conteúdo similar. Ou seja, o conjunto de pares selecionados pelo *baseline* é um subconjunto do conjunto de pares selecionados pela primeira fase do algoritmo DREAM.

A primeira fase do algoritmo DREAM recuperou 8.906.742 de pares de sítios candidatos da WBR2010B. O próximo passo seria aplicar o validador automático a esse conjunto de pares e considerá-lo como um gabarito. Porém, devido ao alto custo do processo de validação automática, optamos por montar um gabarito menor, mas que ainda assim desse suporte estatístico aos nossos resultados. Vale ressaltar que essa decisão não interfere na avaliação de técnicas de aprendizado de máquina como solução alternativa para o problema da replicação de sítios.

O tamanho escolhido para o gabarito foi de dez mil pares. Os dez mil pares foram selecionados aleatoriamente a partir dos 8.906.742 pares retornados pela primeira fase do algoritmo DREAM. Por fim, esse conjunto foi submetido ao validador automático com o objetivo de gerar o gabarito final, onde os algoritmos seriam treinados e testados. A tabela 4.2 apresenta o número total de sítios na WBR2010B, o número de pares possíveis de serem gerados com essa quantidade se sítios, a quantidade de pares de sítios candidatos a réplica retornados pela primeira fase do algoritmo DREAM, o tamanho escolhido para o gabarito e o número total de réplicas encontradas pelo validador automático dentro do gabarito.

Total de sítios na WBR2010B	2.002.955
Total de pares possíveis	2.005.913.364.535
Candidatos recuperados pela 1ª fase do DREAM	8.906.742
Tamanho do gabarito	10.000
Réplicas encontradas no gabarito	583

Tabela 4.2. Resumo das informações a respeito do gabarito.

4.2 Classes de Sítios

A primeira tentativa de obtenção do conjunto de réplicas do gabarito descrito na Seção 4.1 foi realizada com o uso do validador automático original proposto em [Bharat & Broder, 1999]. Porém esse validador não se comportou como esperado, cometendo falhas em situações em que a reposta parecia ser óbvia. Tal comportamento nos levou

a conduzir um estudo para classificar os pares de sítios problemáticos para o validador automático original.

O objetivo desse estudo foi conhecer e entender as causas das falhas cometidas pelo validador e, com isso, realizar modificações para evitá-las. As falhas identificadas estão relacionadas principalmente às mudanças que a Web sofreu desde que o validador automático original foi proposto, há 11 anos. O estudo realizado gerou três classes de sítios: conteúdo volátil, caminhos trocados e conteúdo regional.

4.2.1 Conteúdo Volátil

Dentre os sítios que causaram problemas ao validador automático original, os desta classe foram os mais frequentes. A principal característica deste tipo de sítio é apresentar páginas com conteúdo volátil, isto é, aquele que muda a cada acesso à página. Nesses casos, mesmo que sejam realizados acessos simultâneos à mesma página, o conteúdo (volátil) recebido irá diferir.

Dentre os sítios que se encaixam nesta classe, os sítios de ofertas figuram como principais. Neste tipo de sítio, o conteúdo principal de sua página raiz muda a cada acesso à página a fim de não apresentar ao usuário sempre os mesmos produtos. Como o validador baseia-se no conteúdo das páginas dos sítios para identificá-los como réplicas, quando uma mesma página com conteúdo volátil era recuperada de sítios replicados, seu conteúdo divergia e o par era marcado como não réplica erroneamente. Exemplos desse tipo de par são listados na Tabela 4.3.

Par de sítios	localização do conteúdo volátil
www.ksesporte.com.br www.materialdenatacao.com.br	página raiz
www.acaocompras.com.br www.agrotama.com.br	página raiz
www.chelsom.com.br www.chelsomprofissional.com.br	página raiz
www.alaserland.com.br www.laserland.com.br	páginas internas
www.guiaspaginasamarelas.com.br www.passeiopublico.com.br	páginas internas conteúdo secundário
www.doegratis.com.br www.newsline.com.br	páginas internas conteúdo secundário

Tabela 4.3. Exemplos de pares de sítios que possuem páginas com conteúdo volátil e localização desse conteúdo.

Em alguns casos, páginas internas dos sítios também possuem conteúdo volátil. Nesses casos, geralmente, tal conteúdo constitui uma informação não central, tal como recomendações de produtos, notícias ou patrocinadores, podendo ser caracterizado como um conteúdo secundário. Quando o conteúdo secundário é volátil e muito maior que o conteúdo principal da página, o mesmo problema das páginas raízes acontece.

4.2.2 Caminhos Trocados

Para verificar se dois sítios A e B são réplicas, o validador automático seleciona aleatoriamente caminhos de A e verifica se eles existem em B e vice-versa. Se os caminhos trocados entre A e B apontam para conteúdos similares aos originais, o sítios são ditos serem réplicas. Os pares de sítios desta classe confundem o validador automático ao permitirem o acesso a caminhos de um sítio A via um outro sítio B, mesmo que A e B não sejam réplicas.

Isso ocorre porque geralmente os dois sítios estão hospedados no mesmo servidor. Quando um caminho não é encontrado num determinado sítio, o servidor o busca num outro sítio em sua base de dados. A Tabela 4.4 mostra uma lista de pares de sítios e exemplos de caminhos que apresentam esse comportamento. Note que, geralmente, os pares de sítios que se comportam desta maneira possuem o mesmo nome de domínio.

Par de sítios	Exemplo de caminho
alexandre.mercadodobebe.com.br biatodarosa.mercadodobebe.com.br	/produto/berco-americano-veneza-branco-803
00005000.beltrano.com.br 2140.beltrano.com.br	/scripts/fotolog/fotolog_usuario.asp? idamigo=2097816&idfoto=3#
arte.centralblogs.com.br downloads.centralblogs.com.br	/post.php?href=porta+pijama+14+02+2011 & KEYWORD=19840&PO

Tabela 4.4. Exemplos de pares de sítios da classe caminhos trocados e seus caminhos que . .

4.2.3 Conteúdo Regional

Os sítios desta classe que causam problemas ao validador automático original geralmente pertencem ao mesmo domínio e apresentam informações de mesma natureza. Os melhores exemplos destes casos são os sítios de jornais regionais. Eles possuem várias páginas com conteúdo comum, que contêm notícias a nível nacional ou de maior repercussão. Porém, notícias de menor repercussão de uma determinada região não

aparecem nos sítios de outras regiões. As páginas com conteúdo comum fazem com que o validador marque esses pares como réplicas erroneamente. Outro bom exemplo são os sítios de ofertas que apresentam páginas com textos similares descrevendo produtos ou serviços diferentes, que variam de região para região. A Tabela 4.5 lista exemplos de pares de sítios desta classe e a natureza de seu conteúdo.

Par de sítios	Natureza do Conteúdo
ac.noticianahora.com.br ms.noticianahora.com.br	Jornais
abadia-de-goias.iclaz.com.br abelardo-luz.iclaz.com.br	Classificados
www.concursoce.com.br www.concursoto.com.br	Notícias de concursos

Tabela 4.5. Exemplos de pares de sítios da classe conteúdo regional e a natureza de seu conteúdo.

4.3 Modificações Aplicadas ao Validador Automático

O objetivo desta seção é propor melhorias para o validador automático (vide Seção 2.6) com base no estudo de classes de sítios realizado na seção anterior. Assim, para cada uma das três classes estudadas, propomos melhorias específicas no validador automático.

Para tornar o validador tolerante a pares de sítios da classe Conteúdo Volátil (vide Seção 4.2.1), propomos admitir que alguns dos caminhos comparados entre os dois sítios suspeitos de replicação possam possuir conteúdos não similares. Com isso possíveis páginas com conteúdo volátil selecionadas para comparação poderiam ser desconsideradas. Na maioria das vezes em que um sítio apresenta conteúdo volátil, tal conteúdo encontra-se apenas em sua página raiz. Assim, propomos considerar como réplicas pares que apresentem até dois pares de páginas não similares, assumindo que as páginas raízes dos dois sítios possam ter sido selecionadas para comparação. Nos experimentos realizados, comparamos 20 pares de páginas para cada par de sítios candidatos e exigimos que apenas 18 das 20 comparações fossem similares.

Para resolver o problema dos sítios da classe Caminhos Trocados (vide Seção 4.2.2), que respondem a caminhos que não existem, bastaria comparar suas páginas raízes. A página raiz é a única página que sempre existe em qualquer sítio e que pode funcionar como uma identidade para o mesmo. Assim, pares de sítios cujos conteúdos

das páginas raízes diferem não poderiam ser réplicas. Porém, isso é exatamente o que acontece com os sítios da classe conteúdo volátil: as páginas raízes diferem mesmo para o mesmo sítio. Uma possível solução para este problema é identificar páginas com conteúdo volátil e não utilizá-las na validação de pares candidatos. Uma forma de identificar páginas com conteúdo volátil é realizar uma série coletas da mesma página dentro de períodos curtos de tempo. Se seu conteúdo mudar de requisição pra requisição, então ele pode ser considerado volátil.

Com isso, resolvemos o problema da classe conteúdo volátil e parte do problema da classe caminhos trocados. A parte não resolvida do problema com a classe caminhos trocados são os pares cuja página raiz possui conteúdo volátil. Nesses casos, a validação pode ser feita com base nas informações de conectividade entre os sítios: sítios replicados tendem a possuir um conjunto bastante semelhante de apontadores externos (*outlinks*).

Todas as modificações propostas até então ajudam na solução dos problemas da classe Conteúdo Regional (vide Seção 4.2.3). No entanto, uma medida extra pode ser adotada nesses casos: buscar por elementos regionais nos nomes de servidor. Por fim, para os casos onde não foi possível decidir com segurança uma classificação para um determinado par, criamos uma terceira classificação: os pares suspeitos. Os pares suspeitos são avaliados por humanos para que possam receber uma classificação final como réplica ou não réplica. Nos experimentos, apenas 23 pares foram classificados como suspeitos dentre os 10.000 pares do gabarito.

De fato, as modificações implementadas no validador automático utilizado neste trabalho foram a flexibilização no número de comparações e a criação da classe de pares suspeitos. A adoção dessas duas modificações foi suficiente para nossos propósitos, como mostra a precisão de 100% obtida pelo novo validador nos experimentos realizados, relatada na Seção 4.8.

4.4 Métricas

Esta seção descreve as métricas precisão, revocação e F1 utilizadas na avaliação experimental do algoritmo DREAM. As três métricas foram escolhidas por serem utilizadas nos trabalhos da literatura no tópico de detecção de réplicas de sítios web.

4.4.1 Precisão

Em problemas de recuperação de informação, precisão é a fração dos elementos recuperados que é relevante. Para o problema de seleção de pares candidatos a serem réplicas,

por exemplo, elementos relevantes são pares de sítios replicados. Assim, a precisão dos métodos de seleção é dada pela Equação 4.1, onde R é conjunto dos pares conhecidos de réplicas e r é conjunto dos pares que o método diz serem réplicas.

$$precisão = \frac{|r \cap R|}{|r|} \tag{4.1}$$

Para o problema de detecção réplicas de sítios web, o mais importante é manter alto o nível de precisão. Isso porque, sítios erroneamente marcados como réplicas serão removidos das respostas da máquina de busca e não serão mais visitados pelo coletor.

4.4.2 Revocação

Em problemas de recuperação de informação, revocação é a fração dos elementos relevantes conhecidos que foi recuperada. Como citado anteriormente, para o problema de seleção de pares candidatos a serem réplicas, elementos relevantes são pares de sítios replicados. Assim, a revocação dos métodos de seleção é dada pela Equação 4.2, onde R é conjunto dos pares conhecidos de réplicas e r é conjunto dos pares que o método diz serem réplicas.

$$revocação = \frac{|r \cap R|}{|R|} \tag{4.2}$$

Vale lembrar que a revocação neste trabalho é relativa ao gabarito de dez mil pares, uma vez que o número total de réplicas da coleção não é conhecido.

4.4.3 F1

A métrica F1 é a média harmônica entre precisão e revocação. Além de combinar os valores de precisão e revocação em uma única métrica, ela tem a propriedade de ser bastante afetada quando um dois valores base é baixo. A métrica F1 é dada pela Equação 4.3.

$$F1 = 2 \times \frac{precisão \times revocação}{precisão + revocação} \tag{4.3}$$

Vale lembrar que, como a revocação é relativa ao gabarito de dez mil pares, a métrica F1 também é relativa ao mesmo gabarito.

4.5 Ajuste de Parâmetros

Os algoritmos de aprendizado de máquina estudados possuem um parâmetro ajustável que regula qual o limite mínimo de certeza necessário para que um par de sítios candidatos a réplica seja considerado réplica. Essa certeza é expressa em forma de probabilidade. No caso do NormPaths [da Costa Carvalho et al., 2007], a certeza de o par ser réplica é dada por uma pontuação ao invés de uma probabilidade. Para que essa pontuação pudesse ser encarada como uma probabilidade foi realizada uma transformação, apresentada mais adiante.

Um estudo foi realizado analisando as probabilidades onde cada algoritmo alcança o melhor desempenho. A métrica F1 foi adotada para medir esse desempenho por sumarizar o valor da precisão e da revocação em apenas uma curva no gráfico. No restante da dissertação esse limite mínimo de certeza será chamado de ponto de corte.

A seguir apresentamos a transformação aplicada sobre a pontuação retornada pelo NormPaths e, em seguida apresentamos, para cada um dos algoritmos estudados, as curvas da métrica F1 com relação à posição no *ranking*, para os pontos de corte 0,1, 0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8, e 0,9.

NormPaths

No caso do NormPaths, não há uma probabilidade de o par analisado ser réplica. Ao invés disso, o método computa uma pontuação chamada *rsim* onde, quanto maior seu valor, maiores as chances de o par em questão ser um par de réplicas. O *rsim* é um valor real maior ou igual a zero. Assim, para que fosse possível parametrizar também o NormPaths em termos de probabilidades, foi preciso realizar transformações nos valores do *rsim*. As transformações aplicadas têm o objetivo de suavizar a curva do *rsim* e normalizar os valores para um intervalo entre zero e um. Essas transformações são apresentadas na Equação 4.4, onde *minrsim* é o menor valor de *rsim* e *maxrsim* é o maior valor de *rsim*.

$$p = \frac{\log(rsim) - \log(minrsim)}{\log(maxrsim) - \log(minrsim)} \tag{4.4}$$

A Figura 4.1 apresenta as curvas de *rsim* antes (esquerda) e depois (direita) da aplicação das transformações. Como pode ser observado, antes da transformação a curva do *rsim* apresenta distribuição semelhante à Lei de Potência. Já a curva obtida após a transformação viabiliza o estudo dos pontos de corte 0,1, 0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8 e 0,9, definidos para o estudo dos algoritmos.

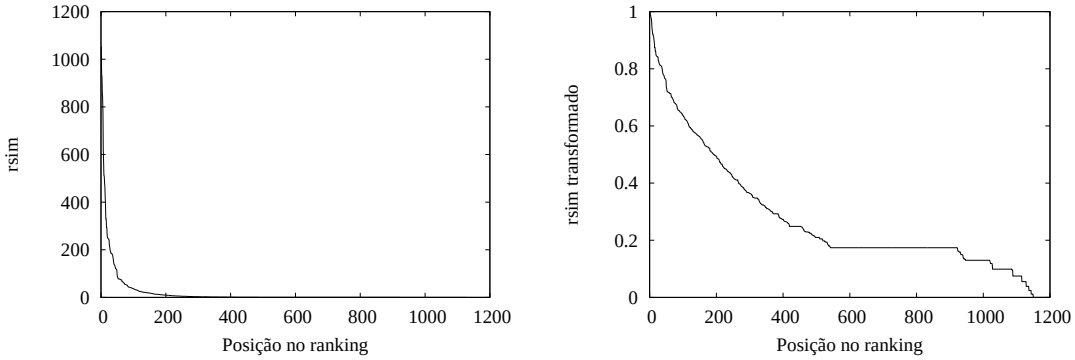

Figura 4.1. Curvas de *rsim* antes (esquerda) de depois (direita) da aplicação das transformações.

A parte reta da curva obtida após a transformação dos valores de *rsim* (gráfico da direita) corresponde aos pares formados por sítios que aparecem em apenas um par, ou seja, que só possuem uma suspeita de réplica em toda a base da máquina de busca. Esse comportamento já era esperado, uma vez que o mais comum é que um sítio seja replicado apenas uma vez.

Obtidos os novos valores de *rsim* é possível prosseguir com o estudo dos pontos de corte para o NormPaths. Como pode ser visto no gráfico da Figura 4.2, o ponto de corte que levou ao maior nível de F1 foi 0,1, seguido por 0,3. O nível mais baixo de F1 foi alcançado com o ponto de corte 0,9 provavelmente por ser muito restritivo e reduzir consideravelmente a revocação do método.

DREAM-DT – Aprendizado Utilizando Árvore de Decisão

Nos experimentos realizados com o algoritmo DREAM fazendo uso de um algoritmo de árvore de decisão (DREAM-DT), todos os pontos de corte obtiveram níveis de F1 acima de 82%, como mostra a Figura 4.3. O maior nível de F1 foi alcançado pelo ponto de corte 0,4, ao passo que 0,1 levou ao nível mais baixo. Isso acontece porque esse é um limiar muito tolerante, o que possibilita o aparecimento de falso-positivos. O segundo menor nível de F1 foi obtido pelo ponto de corte 0,9 que, por ser muito restritivo, favorece o aparecimento de falso-negativos.

Nos experimentos realizados com o algoritmo DREAM-DT o algoritmo de árvore de decisão utilizado foi do tipo C4.5 com fator de confiança de 25%, no mínimo 2 instâncias por folha, com poda e sem redução de erro na poda, como sugere o pacote [Markov & Russell, 2006].

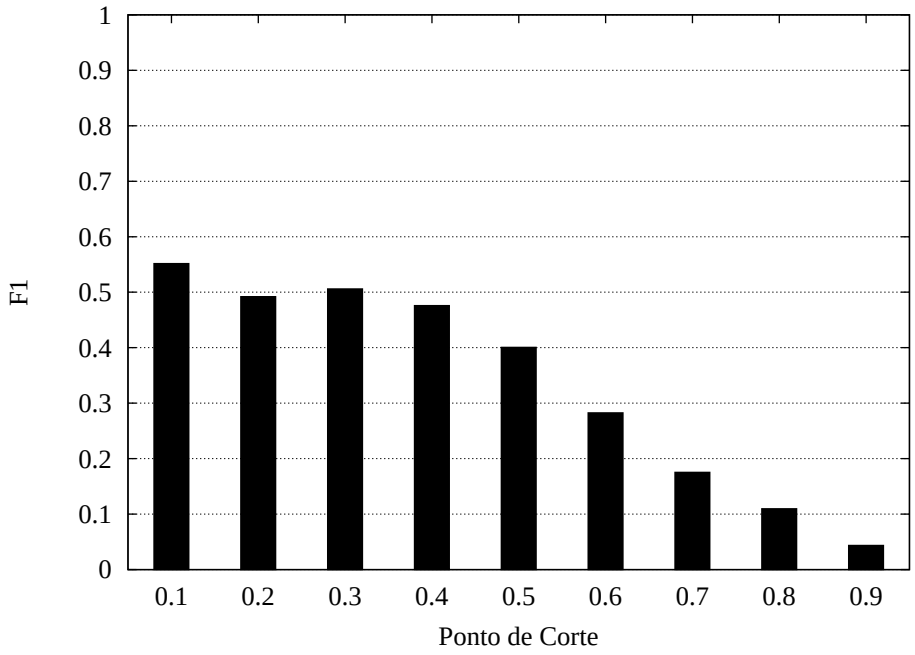

Figura 4.2. Níveis finais de F1 do NormPaths para cada ponto de corte analisado.

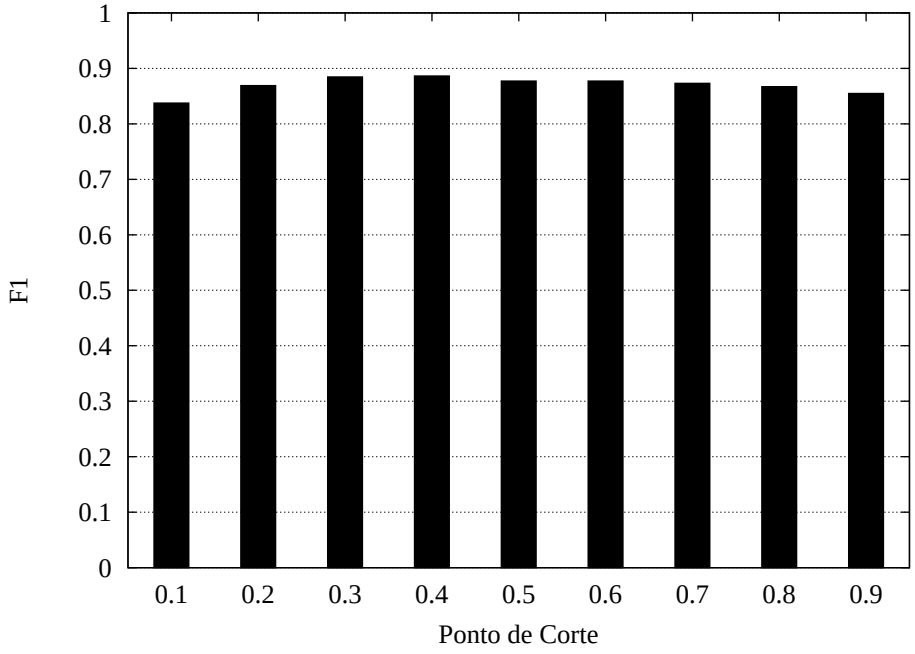

Figura 4.3. Níveis finais de F1 do algoritmo DREAM-DT para cada ponto de corte analisado.

DREAM-LAC – Aprendizado Utilizando Classificação Associativa

Nos experimentos realizados com o algoritmo DREAM fazendo uso de um algoritmo de Classificação Associativa (DREAM-LAC) o melhor ponto de corte também foi 0,4,

como mostra o gráfico da Figura 4.4. Como a maior probabilidade retornada pelo DREAM-LAC foi de 86,6%, nenhum par foi recuperado quando o ponto corte foi ajustado para 0,9, levando a métrica F1 à 0%. A distribuição das probabilidades do DREAM-LAC esteve concentrada em valores mais baixos. Por isso, os pontos de corte mais altos (mais restritivos) alcançaram os menores níveis de F1. Por outro lado, por ser muito permissivo, o ponto de corte 0,1 também esteve entre os que alcançaram baixos níveis de F1.

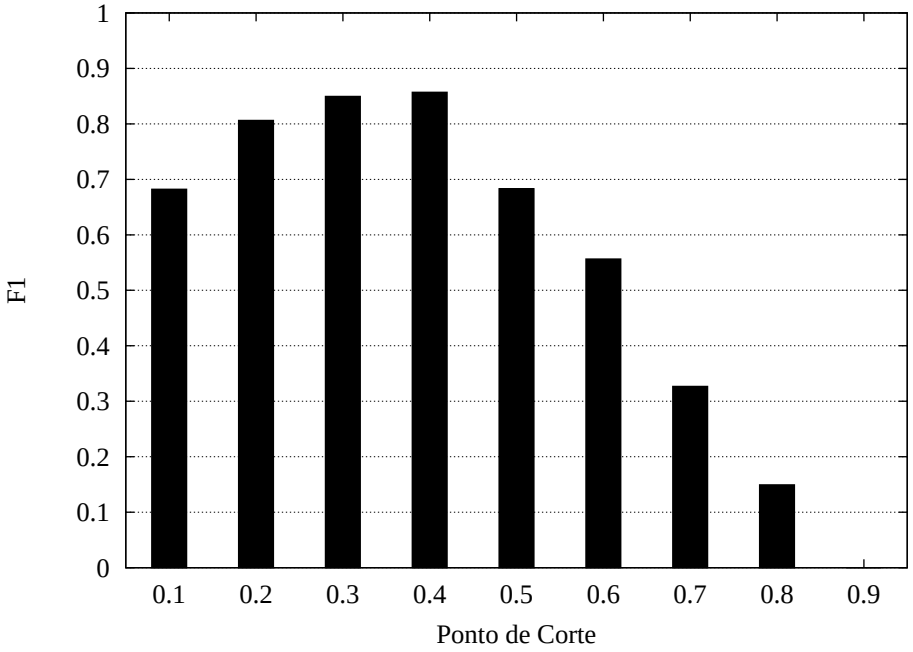

Figura 4.4. Níveis finais de F1 do algoritmo DREAM-LAC para cada ponto de corte analisado.

Para os experimentos com o algoritmo DREAM-LAC os dados das características precisaram ser discretizados. O método de discretização escolhido foi o proposto por Fayyad & Irani [1993], escolhido porque mostrou-se melhor no estudo realizado por Veloso [2009]. Os parâmetros utilizados nos experimentos com o algoritmo DREAM-LAC foram 1 para o suporte mínimo, 0,1% de confiança e tamanho máximo de regra igual à 7, como sugere o pacote [Veloso, 2009].

DREAM-RF – Aprendizado Utilizando Combinações de Árvores

Para o algoritmo DREAM utilizando um algoritmo de Combinações de Árvores (DREAM-RF) o melhor ponto de corte foi 0,6 como mostra o gráfico da Figura 4.5. Em geral o melhor ponto de corte deverá ser um valor mediano: não muito alto (muito restritivo) nem muito baixo (muito permissivo). Para o DREAM-RF, o menor nível

de F1 ocorreu com o ponto de corte 0,1 seguido pelo 0,9, exatamente por serem muito restritivo e muito permissivo, respectivamente.

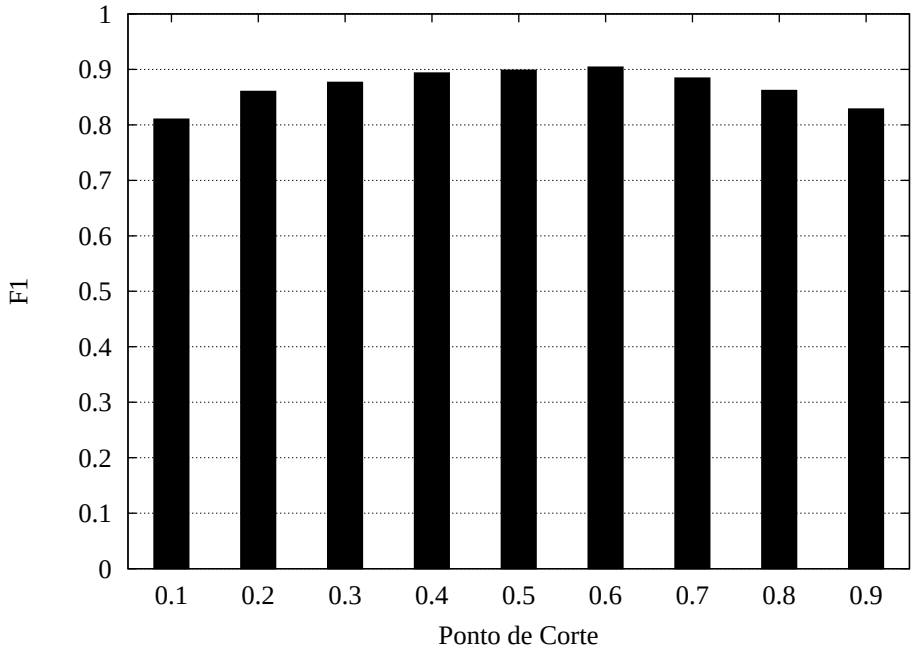

Figura 4.5. Níveis finais de F1 do algoritmo DREAM-RF para cada ponto de corte analisado (escalas ajustadas).

Nos experimentos realizados com o algoritmo DREAM-RF os parâmetros utilizados para o algoritmo de combinação de árvores foram: profundidade máxima ilimitada das árvores, número de árvores geradas igual a 10, número de atributos a ser usado na seleção aleatória igual a 0 e semente para números aleatórios igual a 1, como sugere o pacote [Markov & Russell, 2006].

DREAM-SVM – Aprendizado Utilizando Máquinas de Vetor de Suporte

O algoritmo DREAM também foi experimentado utilizando o classificador Máquinas de Vetor de Suporte (DREAM-SVM). Apesar de a versão do classificador SVM utilizada nos experimentos retornar valores de probabilidades entre zero e um, as probabilidades obtidas para os pares testados foram binárias, ou seja, zero ou um. A principal razão para esse fato é que, em geral, o SVM não trabalha bem quando poucas características do problema estão disponíveis. Assim, as sete características utilizadas nos experimentos não foram suficientes para criar um bom cenário para sua aplicação. A distribuição das probabilidades do DREAM-SVM é dada por uma sigmóide e, quando

utilizado com poucas características, o algoritmo tende a concentrar as probabilidades nos extremos dessa curva. Assim, como as probabilidades obtidas foram zero ou um e todos os pontos de corte estão entre esses valores, os níveis de F1 obtidos foram os mesmos para todos os pontos de corte, como mostrado na Figura 4.5.

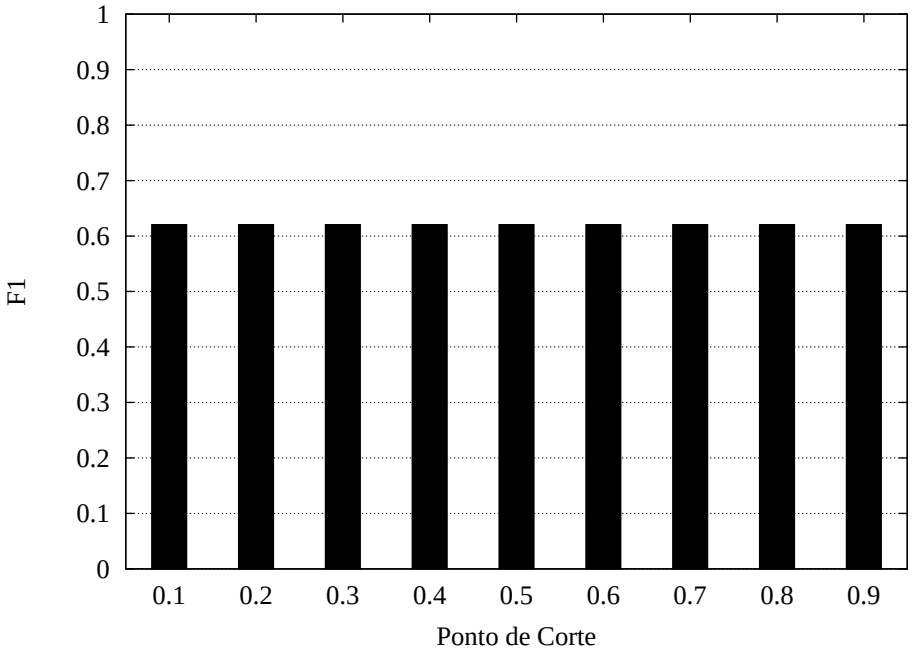

Figura 4.6. Níveis finais de F1 do algoritmo DREAM-SVM para cada ponto de corte analisado.

Contudo, os algoritmos DREAM-DT, DREAM-LAC e DREAM-RF também foram aplicados ao mesmo número de características e, mesmo assim, alcançaram resultados bem melhores. Assim, os resultados do DREAM-SVM não justificam a computação de mais características, visto que bons resultados podem ser obtidos por outros algoritmos baseados em aprendizado de máquina sem a necessidade de características adicionais.

Nos experimentos, os parâmetros utilizados para o algoritmo DREAM-SVM foram: parâmetro de complexidade C igual a 1, tipo de filtro igual a normalização dos dados de treino, função *kernel* PolyKernel com tamanho de *cache* igual a 250007 e expoente igual a 1, e semente para número aleatório igual a 1, como sugere o pacote [Markov & Russell, 2006].

4.6 Distribuições de Probabilidades

O gráfico na Figura 4.7 mostra a distribuição das probabilidades retornadas por cada algoritmo. Mais especificamente, cada algoritmo retorna, para cada par candidato, uma probabilidade desse par ser de fato uma réplica. Nós aplicamos um ajuste de forma a modelar cada distribuição por meio de uma função. A função que melhor ajustou a distribuição de probabilidade dos algoritmos DREAM-DT, DREAM-LAC e DREAM-RF foi um polinômio de quinto grau, ao passo que a função que melhor ajustou a distribuição de probabilidade para o algoritmo DREAM-SVM foi uma sigmóide: $\frac{1}{1+b\times exp(-a\times x)}$. Aparentemente, a distribuição de probabilidade tem uma grande correlação com a eficácia do algoritmo, no sentido de que algoritmos com distribuições mais suaves demonstram-se ser mais eficazes do que algoritmos com distribuições abruptas.

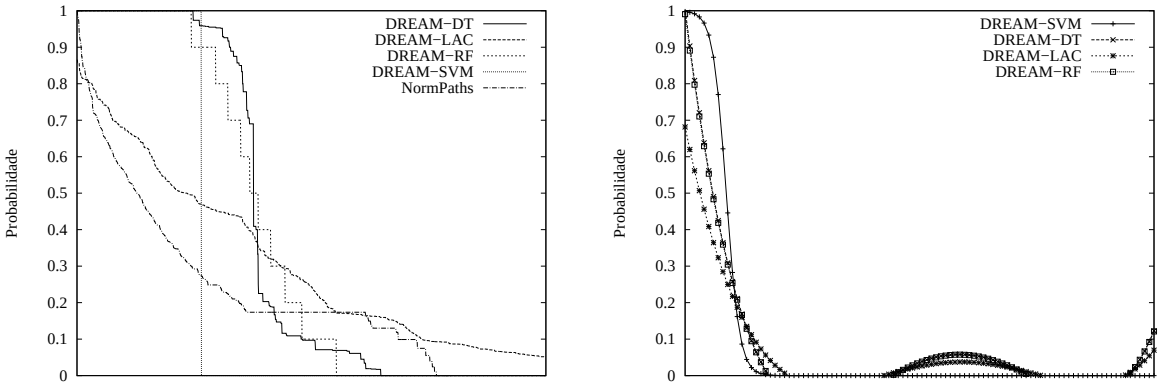

Figura 4.7. Distribuições de probabilidades obtidas pelos algoritmos DREAM-DT, DREAM-LAC, DREAM-RF, DREAM-SVM e NormPaths.

4.7 Comparação Entre os Algoritmos

Esta seção apresenta mais detalhes do comportamento dos algoritmos com relação aos pontos de corte escolhidos e apresenta também a comparação entre os resultados finais obtidos por cada um deles. Mais precisamente, são apresentadas as curvas de precisão, revocação e F1 em todos os pontos do *ranking* recuperado por cada algoritmo. A comparação entre eles é feita por meio de suas curvas de F1 e curvas de precisão por revocação.

A Figura 4.8 apresenta o desempenho do NormPaths, nosso *baseline*, em termos de precisão, revocação e F1. Com o ponto de corte 0,1, o *ranking* recuperado foi de 1027 pares de sítios candidatos a réplica. Como é possível observar na figura, o método é preciso no topo do *ranking* recuperado, porém, à medida que o restante do ranking

é analisado, a precisão do método cai consideravelmente até chegar ao valor final de 43,2%. Em contrapartida, a revocação sobe até alcançar os 76,2%, o que dá ao método 55,2% de F1.

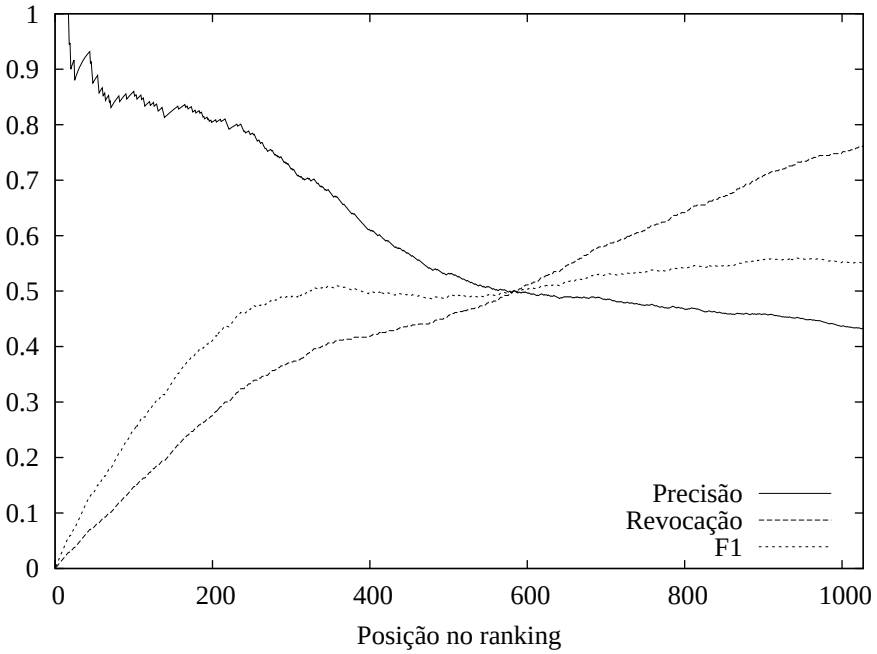

Figura 4.8. Curvas de precisão, revocação e F1 alcançados pelo NormPaths com ponto de corte em 0,1.

Como discutido na Seção 2.6, uma lista imprecisa de pares candidatos à réplica pode aumentar os custos da validação automática dos pares candidatos. Sendo assim, a precisão de 43,2% do NormPaths pode ser considerada baixa para os objetivos de sugerir pares de possíveis réplicas.

O algoritmo DREAM-DT recuperou um *ranking* de 577 pares de sítios candidatos a réplica para o ponto de corte 0,4. Como mostra o gráfico da Figura 4.9, o algoritmo alcançou resultados próximos a 90% nas três métricas. Mais precisamente, os resultados finais foram de 89,1% de precisão, 88,2% de revocação e 88,6% de F1. Esse último valor supera o do *baseline* em 33,4%. Vale ressaltar que até a 352ª posição no *ranking* a precisão do método se mantém em 100%, para a qual a revocação é de 60,4% e F1 de 75,3%.

O algoritmo DREAM-LAC também conseguiu superar o NormPaths nas três métricas. Com o ponto de corte ajustado em 0,4, foram recuperados 549 pares de sítios candidatos a réplica. Como apresentado na Figura 4.10, os resultados finais alcançados chegaram a 88,3% de precisão, 83,2% de revocação e 85,7% de F1, superando o *baseline* em 30,5%, com relação à métrica F1.

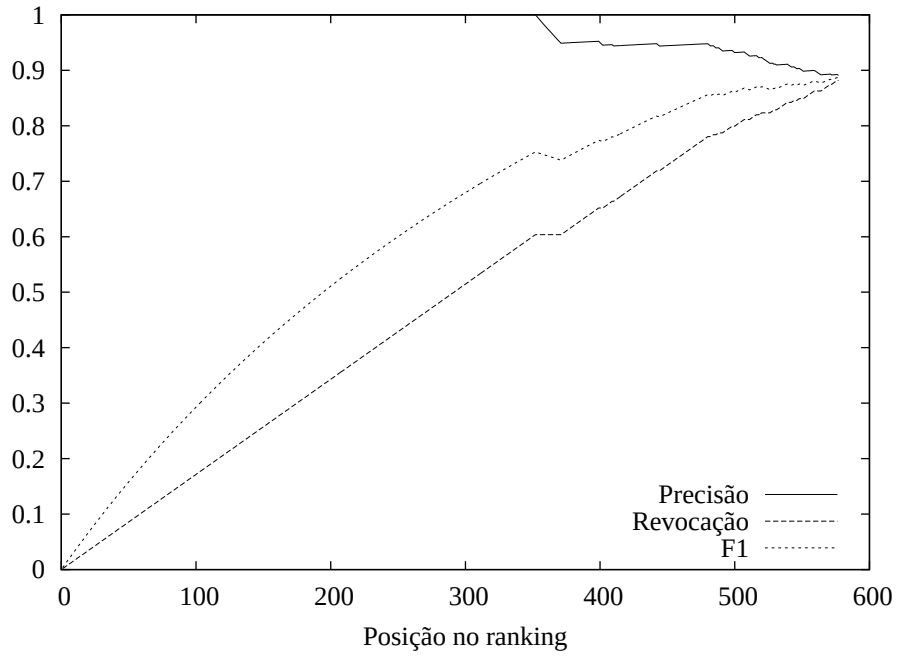

Figura 4.9. Níveis de precisão, revocação e F1 alcançados pelo algoritmo DREAM-DT.

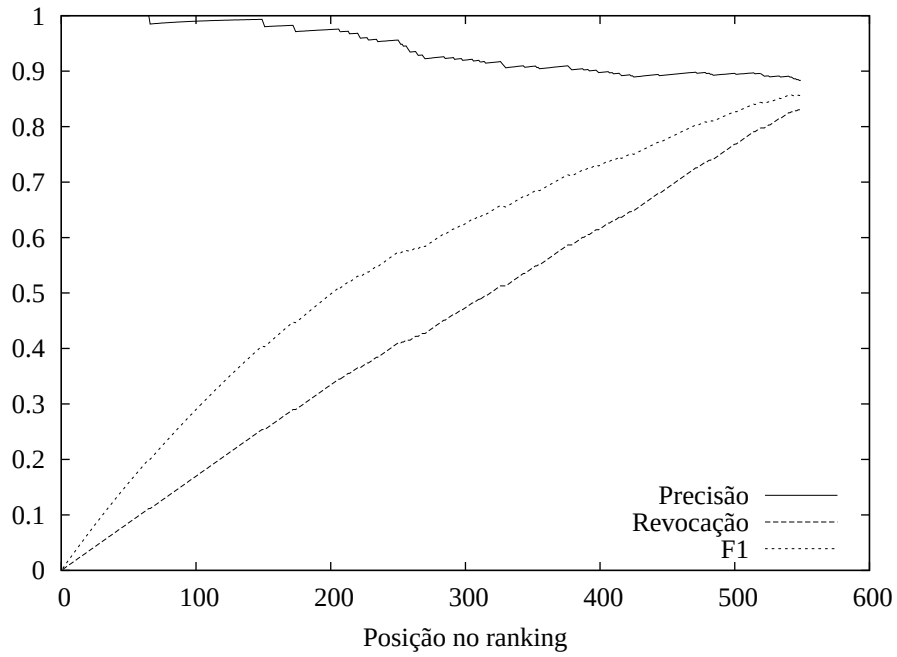

Figura 4.10. Níveis de precisão, revocação e F1 alcançados pelo algoritmo DREAM-LAC.

O algoritmo DREAM-RF alcançou os melhores resultados nos experimentos realizados, obtendo um ganho de 35,1% sobre o NormPaths na métrica F1. A Figura 4.11 mostra os níveis de precisão, revocação e F1, que chegaram a 92,9%, 87,9% e 90,3%,

respectivamente. O *ranking* recuperado foi de 552 pares de sítios candidatos a réplica. Vale ressaltar que a precisão se mantém em 100% até a 361ª posição do *ranking* e que até a posição 429 sua precisão se mantém superior a 99% (com revocação de 72,9% e F1 de 84,0%).

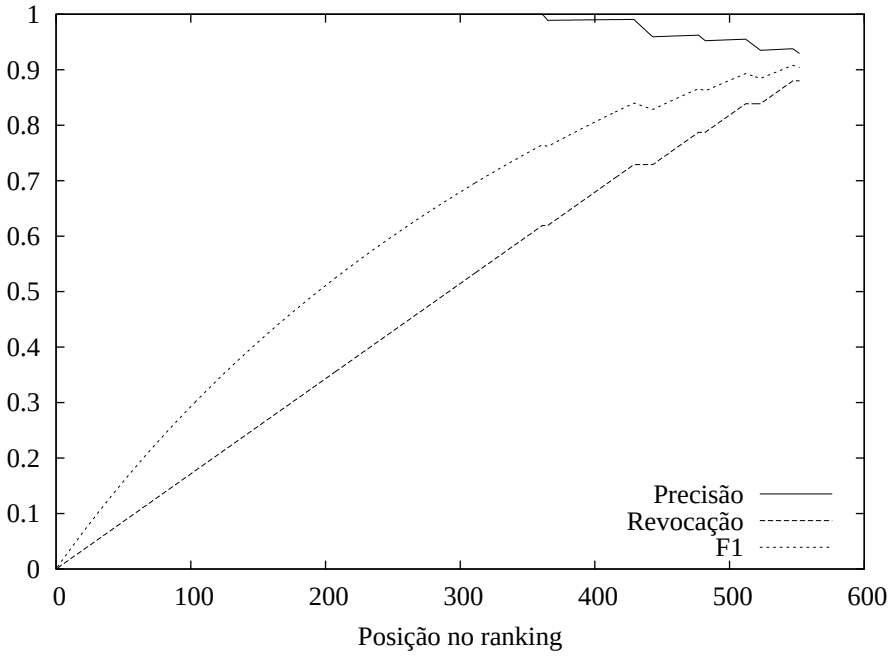

Figura 4.11. Níveis de precisão, revocação e F1 alcançados pelo algoritmo DREAM-RF.

O algoritmo DREAM-SVM obteve os menores ganhos com relação ao baseline. Mais especificamente, o ganho obtido foi de 6,8% na métrica F1, contra 35,1% do DREAM-RF. Vale lembrar que esse algoritmo não teve um cenário favorável nos experimentos, devido ao número de características utilizadas. Mesmo assim, ele superou o NormPaths nas métricas precisão e F1. O algoritmo DREAM-SVM recuperou 397 pares de sítios candidatos a réplicas e alcançou os valores finais de 76,6% de precisão, 52,1% revocação e 62,0% de F1.

Como discutido na seção anterior, as probabilidades retornadas pelo classificador SVM foram todas um (réplica) ou zero (não réplica). Como todos os pares recuperados como réplicas tiveram a mesma probabilidade (um) não é possível estabelecer uma ordem entre eles. Assim, a noção de *ranking* não faz sentindo na análise do SVM. Por isso, as curvas de precisão, revocação e F1 em cada posição do *ranking* do SVM não puderam ser traçadas.

Passamos agora a comparação entre os algoritmos experimentados. O gráfico da Figura 4.12 apresenta as curvas de F1 para cada posição dos rankings retornados

pelos algoritmos NormPaths, DREAM-DT, DREAM-LAC e DREAM-RF. Note que, como discutido anteriormente, como não é possível determinar a ordem entre os pares recuperados pelo DREAM-SVM, os valores de F1 para cada posição no *ranking* não puderam ser computados.

Como é possível observar no gráfico da Figura 4.12, as quatro curvas se sobrepõem nas primeiras posições do *ranking*. Porém, a partir da 25ª posição (aproximadamente) o NormPaths começa a se distanciar dos outros algoritmos e, por volta da 360ª posição, sua curva de F1 começa se estabilizar chegando a 55,2% no final do *ranking*. Vale ressaltar que a curva de F1 do algoritmo DREAM-RF é superior as dos outros algoritmos em todas as posições do *ranking*. Os níveis finais de F1 dos algoritmos DREAM-DT, DREAM-LAC e DREAM-RF são de 91,7%, 90.5% e 95,6%, respectivamente.

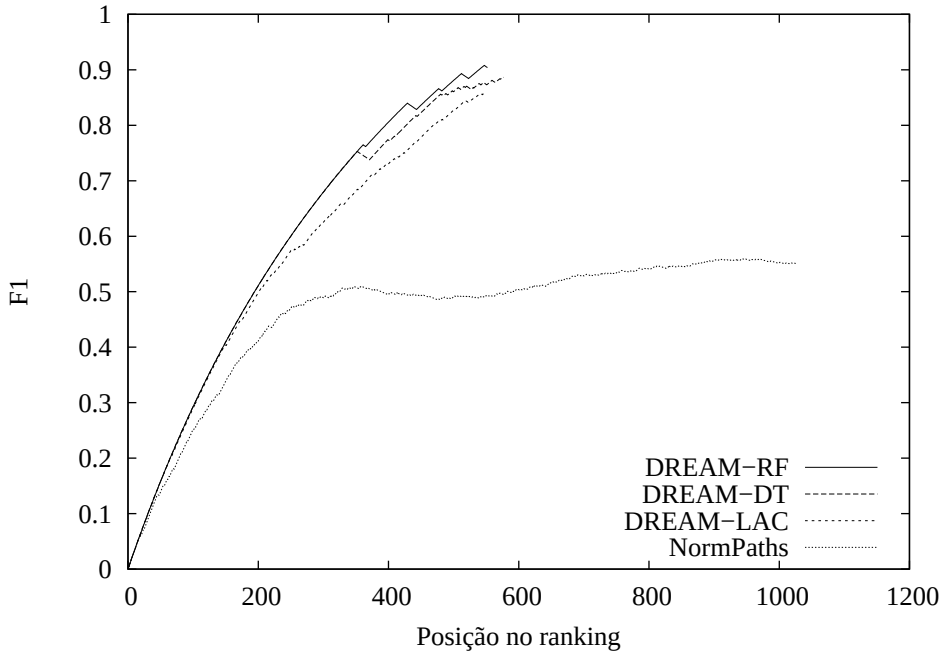

Figura 4.12. Comparação entre curvas de F1 dos algoritmos NormPaths, DREAM-DT, DREAM-LAC e DREAM-RF.

O gráfico da Figura 4.13 apresenta as curvas de precisão por revocação obtidas pelos algoritmos NormPaths, DREAM-DT, DREAM-LAC e DREAM-RF. Novamente, note que, como não é possível determinar a ordem entre os pares recuperados pelo DREAM-SVM, não foi possível traçar sua curva de precisão por revocação.

É possível notar no gráfico da Figura 4.13 que o DREAM-RF obteve resultados mais precisos que os outros algoritmos em todos os níveis de revocação. Além disso, também é possível observar que sua precisão só desce abaixo de 100% a partir dos 61,9% de revocação. A precisão do algoritmo DREAM-DT só desce abaixo dos 100% a partir

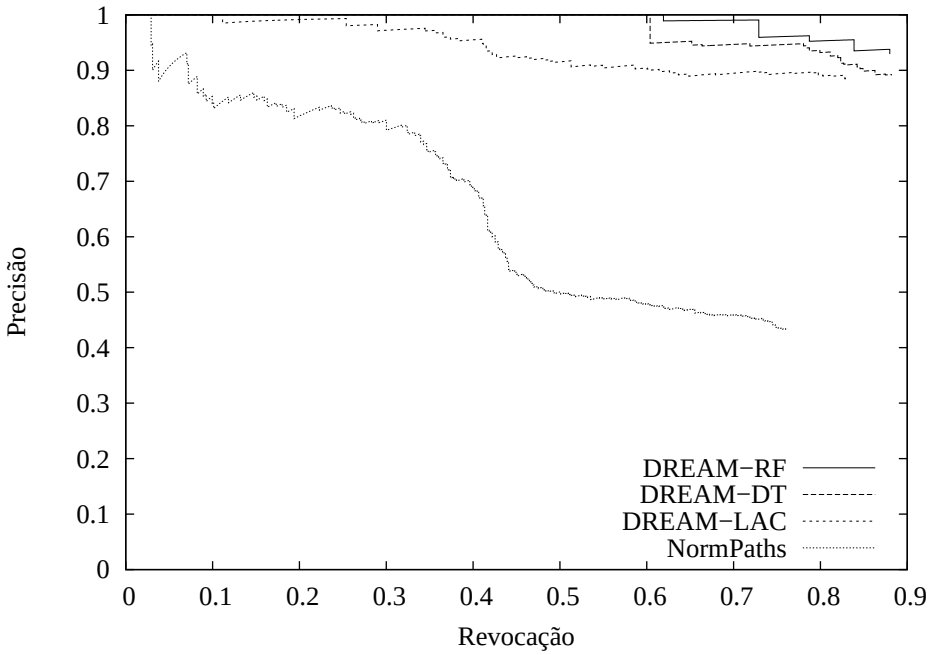

Figura 4.13. Curvas de precisão por revocação obtidas pelos algoritmos NormPaths, DREAM-DT, DREAM-LAC e DREAM-RF.

dos 60% de revocação, sendo que este é o algoritmo obteve a melhor revocação: 514 pares recuperados em 583. O algoritmo DREAM-LAC também obteve bons resultados, sendo mais preciso que o NormPaths em todos os níveis de revocação.

O desempenho alcançado pelo NormPaths neste trabalho, 43,2%, 76,2% e 55,2% de precisão, revocação e F1 respectivamente, foi menor do que o reportado em [da Costa Carvalho et al., 2007], onde algoritmo chegou a níveis de 54,3%, 71,7% e 61,81% de precisão, revocação e F1 respectivamente. Esse comportamento decorre das diferenças entre as coleções experimentais utilizadas. Apesar de o trabalho de [da Costa Carvalho et al., 2007] ter sido publicado em 2007, a coleção usada foi coletada em 2003, ou seja, há cerca de oito anos. Desde então a Web mudou muito, incorporando características com as quais as heurísticas fixas do NormPaths não estão preparadas para lidar.

Uma dessas mudanças é o fato de que a Web não é mais estática: é comum encontrar na Web atual sítios que possuem páginas com conteúdo volátil. Esse conteúdo volátil pode afetar o desempenho do NormPaths, pois a principal característica que o destaca entre os métodos anteriores é o uso (eficiente) de informações acerca do conteúdo das páginas dos sítios. O conteúdo volátil pode levar o NormPaths a não reconhecer como duplicatas páginas idênticas, coletadas de sítios replicados, o que pode acabar gerando falso-negativos.

Além disso, a precisão do NormPaths foi afetada consideravelmente por sítios que possuem páginas (padrão) em comum com conteúdo similar. Em muitos casos, esses sítios são gerados por ferramentas comerciais de confecção de sítios ou estão embasados em geradores de conteúdo que criam páginas padrão, com mesmo caminho e conteúdo muito similar, tais como páginas de contatos, ajuda e calendários, entre outros. Outro caso que induziu o NormPaths ao erro foram os sítios da classe Conteúdo Regional. Os pares de sítios dessa classe possuem o mesmo padrão, mesma estrutura (caminhos iguais) e mesma base de dados, mas não podem ser considerados réplicas.

A Tabela 4.6 resume os resultados finais obtidos nos experimentos apresentados neste capítulo. Nela é apresentado o melhor ponto de corte, a precisão, revocação e F1 final, e ganho sobre o *baseline* (NormPaths) em termos de F1.

Algoritmo	Melhor Ponto de Corte	Precisão	Revocação	F1	Ganho
NormPaths	0,1	43,2%	76,2%	55,2%	
DREAM-RF	0,6	92,9%	87,9%	90,3%	35,1%
DREAM-DT	0,4	89,1%	88,2%	88,6%	33,4%
DREAM-LAC	0,4	88,3%	83,2%	85,7%	30,5%
DREAM-SVM	-	76,6%	52,1%	62,0%	6,8%

Tabela 4.6. Resumo dos resultados obtidos por cada método experimentado.

A significância estatística dos resultados dos algoritmos experimentados foi validada por meio do *t-test* de *Student* [Jain, 1991]. Com 95% de confiança é possível afirmar que houve um empate estatístico entre os algoritmos DREAM-DT, DREAM-LAC e DREAM-RF. Também empataram estatisticamente o NormPaths e o DREAM-SVM.

4.8 Validação Estatística dos Resultados

É importante frisar que os resultados apresentados neste capítulo foram alcançados considerando o gabarito obtido por meio do validador automático modificado. Para validar estatisticamente os resultados alcançados pelos métodos bem como a precisão do validador automático, foi conduzido um experimento onde humanos avaliaram manualmente uma amostra de 370 pares dos 10.000 pares de sítios web do gabarito. O tamanho do conjunto avaliado manualmente foi escolhido de forma garantir uma confiança de 95% e no máximo 5% de erro nos resultados.

A Tabela 4.7 apresenta a precisão estimada alcançada pelo validador automático e por cada método, com base na amostra de 370 pares examinada manualmente. Os resultados são superiores aos obtidos baseados no gabarito. Isso ocorre porque humanos

são capazes de identificar réplicas que o validador automático não identifica. Note que, neste caso, as métricas revocação e F1 não puderam ser computadas uma vez que a validação manual se deu baseada em amostras e, assim, não foi possível determinar o conjunto de réplicas necessário para que a revocação fosse obtida.

Método	Precisão
Validador automático modificado	100,0%
DREAM-RF	95,6%
DREAM-DT	91,7%
DREAM-LAC	90,5%
DREAM-SVM	80,0%
NormPaths	45,8%

Tabela 4.7. Precisão obtida pelo validador automático e por cada método estudado, com relação a amostra de 370 pares verificados manualmente.

4.9 Comparação de Custos

A detecção de réplicas de sítios web em bases de máquinas de busca é uma tarefa que não precisa estar inserida no ciclo de coleta do coletor web. Como réplicas de sítios não são criadas com frequência, a detecção pode ser realizada periodicamente, de forma *offline*, em uma base congelada da máquina da busca. Dessa forma os custos da tarefa de detecção não interferem nos custos da coleta.

Por outro lado, quanto mais preciso o algoritmo de seleção de pares candidatos, menores os custos da tarefa de detecção de réplicas de sítios web. Isso ocorre porque a parte mais cara do processo é, de fato, a fase de validação automática, que exige uma coleta extra de páginas e a comparação entre os conteúdos textuais coletados.

Assim, mesmo que o custo de utilização de algoritmos de aprendizado de máquina seja mais alto que o custo de algoritmos não supervisionados, a detecção de réplicas utilizando o algoritmo DREAM é mais barata que utilizando o NormPaths. Isso porque o algoritmo DREAM é mais preciso do que o algoritmo NormPaths, isto é, o algoritmo DREAM fornece um conjunto de pares candidatos a réplica com um número menor de falso-positivos. Por exemplo, utilizando o NormPaths com as configurações adotadas nos experimentos da Seção 4.7, seria necessário validar 1027 pares para encontrar 444 pares de réplicas, ao passo que utilizando o DREAM-RF com as configurações adotadas nos experimentos da Seção 4.7, seria necessário validar apenas 552 pares para obter-se 511 pares de réplicas. Ou seja, a utilização o DREAM-RF requer 46% a menos de esforço de validação ao mesmo tempo que rende 24% a mais de eficiência que a utilização no NormPaths.

Capítulo 5

Conclusões e Trabalhos Futuros

Neste trabalho propomos um novo algoritmo para identificação de sítios web replicados em bases de máquinas de busca, chamado DREAM (Detecção de RÉplicas usando Aprendizado de Máquina). O algoritmo proposto faz uso de técnicas de aprendizado de máquina para aumentar a efetividade da tarefa de detecção réplicas. O principal objetivo deste trabalho foi determinar se é possível melhorar a detecção de réplicas com as mesmas características usadas na literatura por meio desse tipo de técnica.

Comparamos o algoritmo NormPaths, proposto em [da Costa Carvalho et al., 2007], nosso *baseline*, com quatro versões do algoritmo DREAM: utilizando Árvores de Decisão (DREAM-DT), utilizando Classificação Associativa (DREAM-LAC), utilizando Combinações de Árvores (DREAM-RF) e utilizando Máquinas de Vetor de Suporte (DREAM-SVM). Apenas o DREAM-SVM não superou o *baseline* nas três métricas utilizadas: precisão, revocação e F1. O DREAM-SVM, que não teve um cenário favorável devido ao número de características utilizadas nos experimentos, ficou abaixo do NormPaths apenas na métrica revocação. O algoritmo DREAM-RF obteve os melhores resultados, superando o NormPaths em mais de 35% de F1. O algoritmo DREAM-DT superou o NormPaths em 33,4% de F1 enquanto que para o DREAM-LAC esse valor foi de 30,5%. Já o DREAM-SVM, mesmo prejudicado pelo baixo número de características utilizadas nos experimentos, superou o NormPaths em 6,8% de F1.

Dentre os problemas que afetaram o desempenho do NormPaths podemos citar sítios com conteúdo volátil, sítios de conteúdo regional e páginas padronizadas. Devido a sua característica de se basear no conteúdo, réplicas com páginas voláteis podem levar o NormPaths a concluir que se trata de sítios diferentes, gerando falso-negativos. Por outro lado, sítios com páginas padronizadas (por exemplo, contatos, ajuda e calendários), levaram o método gerar falso-positivos. Sítios de conteúdo regional também

fizeram com que o NormPaths gerasse falso-positivos, devido ou alto número de páginas semelhantes compartilhadas por eles.

Se, por um lado, o desempenho obtido pelo NormPaths sofreu com as novas características da Web, por outro, os métodos baseados em aprendizado de máquina alcançaram bons resultados. Até mesmo o DREAM-SVM, que enfrentou problemas, conseguiu ganhos sobre o NormPaths em termos de F1. Assim, com base nos resultados apresentados, foi possível verificar nossa premissa de que abordagens baseadas em aprendizado de máquina podem melhorar os resultados da literatura em detecção de réplicas de sítios web.

Por combinar várias características de naturezas diferentes e utilizar uma coleção rotulada para aprender os padrões que caracterizam a nova Web, os algoritmos de aprendizado de máquina foram capazes de superar os problemas enfrentados pelo NormPaths e, por isso, alcançar melhores resultados. Concluímos que aprendizado de máquina mostrou-se uma abordagem vantajosa ao uso de heurísticas fixas para o problema de detectar réplicas de sítios em bases de máquinas de busca.

No futuro pretendemos analisar o algoritmo DREAM sobre uma coleção de pares de sítios web maior que a utilizada nesta dissertação e estudar outras características do problema, tais como a conectividade entre os sítios. Além disso, pretendemos estudar o impacto individual de cada característica no desempenho do processo de detecção de candidatos a réplica. Também pretendemos realizar um estudo mais completo sobre as classes de sítios com o objetivo de avaliar os métodos quanto ao desempenho em cada classe.

Referências Bibliográficas

Aizerman, A.; Braverman, E. M. & Rozoner, L. I. (1964). Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning. *Automation and Remote Control*, 25:821--837.

Baeza-Yates, R. & Ribeiro-Neto, B. (2011). *Modern Information Retrieval*. Addison Wesley, 2ª edição.

Bair, E. & Tibshirani, R. (2003). Machine learning methods applied to DNA microarray data can improve the diagnosis of cancer. *Special Interest Group on Knowledge Discovery and Data Mining Explorations Newsletter*, 5:48--55.

Bharat, K. & Broder, A. (1999). Mirror, mirror on the web: A study of host pairs with replicated content. *Computer Networks*, 31:1579--1590.

Bharat, K.; Broder, A.; Dean, J. & Henzinger, M. R. (2000). A comparison of techniques to find mirrored hosts on the www. *Journal of the American Society for Information Science*, 51:1114--1122.

Bharat, K. & Henzinger, M. R. (1998). Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 104--111.

Breiman, L. (2001). Random forests. *Mach. Learn.*, 45:5--32.

Brin, S. & Page, L. (1998). The anatomy of a large-scale hypertextual web aearch engine. *Computer Networks and ISDN System*, 30:107--117.

Broder, A. (1997). On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, pp. 21--29.

Broder, A. Z.; Glassman, S. C.; Manasse, M. S. & Zweig, G. (1997). Syntactic clustering of the web. *Computer Networks and ISDN System*, 29:1157--1166.

Chakrabarti, S.; Dom, B.; Raghavan, P.; Rajagopalan, S.; Gibson, D. & Kleinberg, J. (1998). Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks and ISDN System*, 30:65--74.

Cho, J.; Shivakumar, N. & Garcia-Molina, H. (2000). Finding replicated web collections. *SIGMOD Rec.*, 29:355--366.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20:273--297.

Croft, B.; Metzler, D. & Strohman, T. (2009). *Search Engines: Information Retrieval in Practice*. Addison Wesley, 1ª edição.

da Costa Carvalho, A. L.; de Moura, E. S.; da Silva, A. S.; Berlt, K. & Bezerra, A. (2007). A cost-effective method for detecting web site replicas on search engine databases. *Data & Knowledge Engineering*, 62:421--437.

Fayyad, U. & Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. pp. 1022--1027.

Fetterly, D.; Manasse, M. & Najork, M. (2003). On the evolution of clusters of near-duplicate web pages. In *Proceedings of the First Conference on Latin American Web Congress*, p. 37.

Hajishirzi, H.; Yih, W.-t. & Kolcz, A. (2010). Adaptive near-duplicate detection via similarity learning. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 419--426.

Henzinger, M. (2006). Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 284--291.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. Wiley, 1ª edição.

Kleinberg, J. M. (1998). Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 668--677.

Lacerda, A.; Cristo, M.; Gonçalves, M. A.; Fan, W.; Ziviani, N. & Ribeiro-Neto, B. (2006). Learning to advertise. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 549--556.

Levenshtein, V. I. (1965). Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8--17.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707--710.

Manku, G. S.; Jain, A. & Das Sarma, A. (2007). Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, pp. 141--150.

Markov, Z. & Russell, I. (2006). An introduction to the weka data mining system. *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 38:367--368.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers*, pp. 61--74.

Quinlan, J. R. (1986). Induction of decision trees. *Mach. Learn.*, 1:81--106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

Radlinski, F.; Bennett, P. N. & Yilmaz, E. (2011). Detecting duplicate web documents using clickthrough data. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pp. 147--156.

Robertson, S. E. & Jones, K. S. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129--146.

Robertson, S. E. & Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 232--241.

Salton, G. & Lesk, M. E. (1968). Computer evaluation of indexing and text processing. *Journal of the ACM*, 15:8--36.

Salton, G. & Yang, C. S. (1973). On the specification of term values in automatic indexing. *Journal of Documentation*, 29:351--372.

Schmid, H. (1994). Part-of-speech tagging with neural networks. In *Proceedings of the 15th Conference on Computational Linguistics*, pp. 172--176.

Veloso, A. A. (2009). *Classificação Associativa Sob Demanda*. PhD thesis, Departamento de Ciência da Computação da Universidade Federal de Minas Gerais, Belo Horizonte, Brasil.

Wartick, S. (1992). *Information Retrieval: Data Structures & Algorithms*, chapter Boolean Operations, pp. 264--292. Prentice-Hall.

Witten, I. H.; Bell, T. C. & Moffat, A. (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, 2ª edição.

Ziviani, N.; de Moura, E. S.; Navarro, G. & Baeza-Yates, R. (2000). Compression: A key for next-generation text retrieval systems. *Computer*, 33:37--44.